SUSTAINABLE COMPUTING WITH MOBILE CROWD COMPUTING

THESIS

Submitted in Partial Fulfilment of the Requirements
for the Degree of

Doctor of Philosophy (PhD)

PIJUSH KANTI DUTTA PRAMANIK

Reg. No.: NITD/PhD/CS/2017/00969

Under the Supervision
of

DR. PRASENJIT CHOUDHURY

Associate Professor, Department of Computer Science & Engineering
National Institute of Technology, Durgapur, India

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
National Institute of Technology Durgapur
West Bengal – 713209
India

January 2023

"So many of our dreams at first seems impossible, then they seem improbable, and then, when we summon the will, they become inevitable"
--- Christopher Reeve

To my parents

Smt. Jharna Dash

Sri Pranab Kumar Dutta Pramanik

ACKNOWLEDGEMENTS

This thesis is the outcome of more than seven years of relentless effort. I'd like to commend myself for being able to uphold the zeal, perseverance, tenacity and patience through this long period despite the odds and hurdles.

I thank my supervisor Dr. Prasenjit Choudhury for leading me to this exciting field of research and encouraging to move forward. I appreciate his kindness of patronaging and endorsing my capability and giving me the freedom to carry out the research unreservedly.

I gratefully acknowledge the help and support obtained from my friend and colleague Saurabh Pal who was always there with me be it with personal and moral support or assisting in the research works.

I also thankfully acknowledge the aids received from Dr. Tarun Biswas, Dept. of CSE, IIIT Ranchi, Dr. Bijoy Kumar Upadhyay, Dept. of ECE, TIT, Agartala, Sanjib Biswas, CBS, Kolkata, and Nilanjan Sinhababu, IIT Kharagpur in some of my research works. I sincerely thank Dr. Gautam Bandyopadhyay, Associate Professor, Dept. of Management Studies, NIT Durgapur, for his valuable comments and suggestions each time I asked for.

I thank my labmates Avick, Dhananjay, and Pradeep for lending their helping hands whenever needed.

I'd be at fault not admitting the continuous backing and encouragement received from my family members. I gratefully thank them for believing and having confidence on me.

I'm fortunate to have unconditional support and espousal from my wife Anindita, allowing me to focus on my research exclusively. Finally, I must say sorry to my adorable daughter Pajaswati for failing spending time with her even though see craved for. I missed her childhood.

Date: 30th January 2023

Pijush Kanti Dutta Pramanik
Registration No.: NITD/PhD/CS/2017/00969
Department of Computer Science & Engineering
National Institute of Technology Durgapur
West Bengal – 713209, India

NATIONAL INSTITUTE OF TECHNOLOGY DURGAPUR

CERTIFICATE

It is certified that the work contained in the thesis entitled "Sustainable Computing with Mobile Crowd Computing" has been carried out by me, Pijush Kanti Dutta Pramanik (Roll No.: 15/CA/1503, PT, PhD), under the guidance of Dr. Prasenjit Choudhury. The data reported herein is original and that this work has not been submitted elsewhere for any other Degree or Diploma.

(Signature of Candidate)
Pijush Kanti Dutta Pramanik
Place: Durgapur
Date: 30th January 2023

This is to certify that the above declaration is true.

(Signature of Supervisor)
Dr. Prasenjit Choudhury
Place: Durgapur
Date: 30th January 2023

DECLARATION/STATEMENT OF THESIS PREPARATION

Thesis title: Sustainable Computing with Mobile Crowd Computing

Degree for which the thesis is submitted: PhD

1. Thesis preparation guideline has been followed while preparing the thesis.
2. All specifications regarding thesis format etc. have been followed.
3. The contents of the thesis have been organized based on the guidelines.
4. The thesis has been prepared without resorting to plagiarism.
5. All sources used have been cited appropriately.
6. The thesis has not been submitted elsewhere for a degree.

(Signature of the student with date)

Name: Pijush Kanti Dutta Pramanik
Registration No.: NITD/PhD/CS/2017/00969
Department: Computer Science & Engineering

ABSTRACT

The embracement of information and computation technologies to an enormous extent has resulted in environment contamination drastically. Moreover, the introduction of the IoT and big data applications have garnered a massive amount of digital data. Processing and analysing these data demand vast computing resources, proportionately. The major downside of producing and using computing resources in such volumes is the deterioration of the Earth's environment. The production process of the electronic devices involves hazardous and toxic substances which not only harms human and other living being's health but also contaminate the water and soil. The production and operations of these computers in largescale also results in massive energy consumption and greenhouse gas generation. Moreover, the low use cycle of these devices churns out a huge amount of not-easy-to-decompose e-waste. In this outlook, instead of buying new devices, it is advisable to use the existing resources to their fullest, which will minimize the environmental penalties of production and e-waste.

On the other hand, the advancement of computing technology has miniaturized computers into the scale of few millimetres or centimetres. The new-age processors of the smart mobile devices (SMDs) such as smartphones and tablets require less power and dissipate less heat while offering significant computation capability. This brings to a new breed of revolutionary computing technology – the SMD computing. Evolution of SMDs has actually realized the miniaturization of computing devices with processing capability as par to a microcomputer.

Furthermore, a grid of such SMDs cumulatively can garner enough processing power to resolve complex computational jobs. The philosophy of combining computation power of numerous public-owned SMDs to escalate the computation power leads to the idea of mobile crowd computing (MCC). MCC utilizes the idle computing resources of public's SMDs, available voluntarily or in return of incentives, providing a feasible and cost-effective, flexible, and scalable high-performance computing solution.

In this thesis we advocate for adopting MCC to abate the use of traditional HPCs such as data centres and supercomputers. We aim to establish MCC as the most

feasible computing system solution to achieve sustainable computing. Towards this, we present a detailed comparison, between MCC and other HPC systems such as supercomputers and Grid and Cloud Computing, in terms of environmental effects (e.g., energy consumption, greenhouse gas generation, etc.), which confirms the advantages of MCC as a sustainable HPC system.

Though several empirical works have established the feasibility of mobile-based computing for various applications, there is a lack of comprehensive coverage on MCC. In this regard, we aim to explore the fundamentals and other nitty-gritty of the idea of MCC in a comprehensive manner. Starting with an explicit definition of MCC, we present the enabling backdrops and the detailed architectural layouts of different models of MCC, along with categorising different types of MCC based on infrastructure and application demands. MCC is extensively compared with other HPC systems (e.g., desktop grid, cloud, clusters and supercomputers) and other similar mobile computing systems (e.g., mobile grid, mobile cloud, ad-hoc mobile cloud, and mobile crowdsourcing). MCC being a complex system, various design requirements and considerations are extensively analysed. We meticulously mention the potential benefits of MCC, with special discussions on the ubiquity and sustainability of MCC. The issues and challenges of MCC are critically presented in the light of further research scopes. Importantly, several real-world applications of MCC are identified and propositioned.

To achieve satisfactory performance and QoS of an MCC system, selecting the most suitable resources (SMDs) is crucial. For this, the essential prerequisite is to profile and assess the resource parameters and their present status precisely. However, considering the heterogeneity and dynamicity of these resource parameters, profiling them, and assessing their fitment for different requirements is not trivial. As a result, selecting the most suitable SMDs as resource provider also becomes confounding. In this chapter, we present a methodological approach to profile the candidate SMDs for assessing their resources to be considered for job scheduling. For profiling, we considered various resource parameters, some of which are collected instantaneously, some are accumulated from logged data, and some parameters are derived by analysing the log data.

The selection of appropriate SMDs is generally made based on the computing

capability of an SMD, which is defined by its various fixed (e.g., CPU and GPU power, no. of cores, RAM, etc.) and variable (e.g., current CPU and GPU load, battery remaining, etc.) resource parameters. As the selection is made on different criteria of varying significance, the resource selection problem can be duly represented as an MCDM problem. However, for the real-time implementation of MCC and considering its dynamicity, the resource selection algorithm should be time-efficient. Considering that we aim to find out an MCDM method that would be most suitable to be used for resource selection in such a dynamic and time-constraint environment. For this, we present a comparative analysis of various MCDM methods under asymmetric conditions with varying selection criteria and alternative sets. In this comparative study, we considered the Entropy method to decide criteria weights and EDAS, ARAS, MABAC, MARCOS, and COPRAS methods for resource ranking. We considered four different sizes of decision matrices for evaluation. We executed each program with four datasets on a Windows-based laptop and also on an Android-based smartphone to evaluate the average runtime. Besides time complexity analysis, we perform sensitivity analysis and ranking order comparison to check the correctness, stability, and reliability of the rankings generated by each method.

Scheduling is an important aspect for MCC like any other distributed systems. The overall performance and the integrity of the MCC can be assessed by factors such as execution time, resource utilisation, load balancing, etc. An efficient task scheduler should conform to these requirements. Conversely, an inefficient scheduling method will have a negative impact on the QoS of MCC. Furthermore, considering the battery-powered constrained energy of the MCC resources, i.e., the SMDs, it is crucial to minimise the energy consumption to complete the scheduled task. This can be achieved to some extent by optimising the task scheduling to the appropriate SMDs. However, considering only energy efficiency might lead to a huge load imbalance among SMDs, i.e., the most energy-efficient SMDs would be overloaded most of the time. In a dynamic and heterogeneous system like MCC, it is nontrivial to realise an optimised scheduler, in view of the fact that scheduling in a heterogeneous distributed system is an NP-complete problem. To address this, we present two scheduling solutions. In the first one, we propose a heuristic algorithm for

resource-aware scheduling in MCC with the objectives of minimising makespan and maximising resource utilisation and load balancing. Before scheduling, the resource strength of each SMD is calculated by considering several static and dynamic resource parameters such as CPU clock speed, number of cores, its present load, available RAM and battery, and device temperature. The work is analysed and validated by extensive simulations with synthetic as well as collected datasets. Experimenting with diverse simulation scenarios confirms the consistency and reliability of the proposed algorithm. Our algorithm exhibits significant improvements compared to other popular metaheuristic algorithms such as PSO, GA, and a heuristic algorithm MCT in terms of the considered objectives. The statistical hypothesis tests viz. ANOVA and post hoc tests are carried out to demonstrate the effectiveness of the proposed work. In the second solution, we propose a PSO-based scheduling algorithm to minimise the overall energy consumption among a set of SMDs designated to execute a set of MCC tasks while maintaining a satisfactory load balance level. The proposed method is analysed and validated by extensive simulations with synthetic as well as collected datasets. The work is compared with popular heuristic (MCT, MinMin, MaxMin, and PPIA) and metaheuristic (GA) optimisation algorithms, displaying significant improvements over others in terms of the considered objectives. Here also, ANOVA is carried out to demonstrate the distinctiveness of the proposed PSO-based algorithm.

In a local MCC, where the SMDs are connected to the MCC coordinator through a local network such as WLAN, the availability of the SMDs become crucial for the attainment of MCC and maintaining its QoS. Schedule jobs to the fleeting SMDs would result in frequent job offloading and, in the worst case, job loss, which would affect the overall performance and the QoS of MCC. In a Local MCC, generally, a set of users are available for a certain period regularly. Based on this information, the chances of a user being available for a certain duration from a given point of time can be predicted. In this chapter, we provide an effective model to predict the availability of the users (i.e., their SMDs) in such an MCC environment. We argue that before submitting a job to an SMD, the stability of it is to be assessed for the duration of execution of the job to be assigned. If the predicted availability period is greater than the job size, then only the job should be assigned to the SMD. An

accurate prediction will minimize the unnecessary job offloading or job loss due to the early departure of the designated SMD. Along with experimenting with a readymade API available in Keras named ConvLSTM, we propose an advanced convolutional feature extraction mechanism that is applied to LSTM and GRU-based time-series prediction models for predicting SMD availability. To collect user mobility data, we considered a research lab scenario, where real mobility traces were recorded with respect to a Wi-Fi AP. We compared the prediction performances of convolutional LSTM and GRU with the basic LSTM and GRU and ARIMA in terms of MAE, RMSE, R^2, accuracy, and perplexity. In all the measurements, the proposed convolutional LSTM exhibited considerably better prediction performance.

Besides a centralised system, MCC can also be implemented as a P2P system where the participating SMDs may borrow resources from each other, when needed. Here, a resource-deficient SMD user would be able to seeks it neighbouring resource-rich SMDs' help to carry out a resource-wanting job. It would relieve users from going to the cloud. However, user mobility imposes a serious challenge in P2P MCC (PMCC). User's unpredicted mobility makes the PMCC unstable. To address this, we aim to find a stable cluster of SMD users who would likely to be ideal candidates to form an ad-hoc PMCC. We propose a novel mobility prediction method to assess the probability of a group of SMD users to be relatively static with respect to each other. We submit an algorithm to estimate the relative stability of an SMD with respect to its neighbourhood over a period of time, irrespective of its geographical location. We assess the short-term relative stability between a group of SMD users as well as the long-term mobility pattern between them. For both the experiments, we have used the UCSD dataset that comprises real-traces of 235 mobile device users for 78 days across 402 APs. Furthermore, sometimes it may happen that the required service is not available within the immediate network of the service requester. In this case, we suggest to have a carrier that would carry the request to a service provider in another network, get the service from it and handover to the requester. Here also, considering the mobility of the service consumer, provider, and the carrier should be crucial for proper and timely service exchange. Considering this, we further present a service provisioning model in PMCC based on the

mobility patterns of the above-mentioned three entities. In the second experiment, we specifically focussed to get the information - a) average time gap after a user connects to an AP, b) average duration he/she remains connected to an AP, and c) a set of four users who remains connected to a particular AP simultaneously. Knowing the mobility patterns of the service consumer, provider, and the carrier, in terms of the above-mentioned information, is helpful to bind them in particular time frames. This allows avoiding the liveness problem (consumer waits for the service indefinitely) and availability problem (carrier returns with the service but cannot find the consumer). The estimated results of both experiments are analysed and validated using different metrics.

We aspire to establish a proof-of-concept for the feasibility and use of MCC as a sustainable edge computing solution (MCC-edge). The widespread adoption of utility-based real-time applications has placed the necessity of widescale deployment of edge computing infrastructure. Crowdsourced edge computing is deemed a suitable way out. For the experiment case, we consider a typical smart HVAC system of an office building. We aim to process the HVAC data in real-time using the MCC-edge setup within the building for auto adjustment of the AC controller and error notifications. To maintain the ideal comfort level of the occupants, we present an extensive calculation using the dew point and heat index of the room. Along with a general framework of MCC-edge, a high-level layered architecture of the MCC-edge for HVAC is presented. We report the module-wise design and implementation procedures with exhaustive details. The performance of MCC-edge is statistically compared with the commercial edge and cloud computing solutions in terms of cost, energy consumption, and latency, showing a significant advantage over the two.

Finally, to carry forward the accomplishment of the MCC vision, the future prospects are briefly elucidated. In this thesis, we tried to cover every aspect, in general, that is required to know to understand MCC. We position this work as a preliminary reference for the interested researchers, both novice and experienced, who are keen to work on MCC, as well as other stakeholders willing to explore the benefits of MCC.

Keywords

Sustainable computing
High-performance computing
Smartphone computing
Mobile computing
Mobile grid computing
Volunteer computing
Opportunistic computing
Crowdsourced system
Crowd computing
Mobile crowd computing
Service-oriented computing
Mobile cloud
P2P cloud
Ad-hoc mobile cloud
Edge computing
IoT
HVAC
Resource profiling
Resource selection
Task scheduling
Resource-aware scheduling
Load balance
Availability prediction
Mobility tracing
Mobility prediction
Relative mobility
Benchmarking
MCDM
Entropy
EDAS
ARAS
MABAC
MARCOS
COPRAS
Multicriteria optimisation
Metaheuristic
Particle swarm optimisation
Deep learning
Convolutional feature extraction
CNN
RNN
LSTM
GRU
Dew point
Heat index
Energy consumption
Energy-efficiency
Environmental hazards

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

Acronyms

AC	Air Conditioner
AC	Alternative Current
ACO	Ant Colony Optimization
ACPI	Advanced Configuration and Power Interface
ACT	Average Connection Time
Ag	Silver
AHP	Analytic Hierarchy Process
AI	Artificial Intelligence
Al	Aluminium
ALICE	A Large Ion Collider Experiment
AMOLED	Active-Matrix Organic Light-Emitting Diode
ANP	Analytic Network Process
AP	Access Point
API	Application Programming Interface
AR	Augmented Reality
ARAS	Additive Ratio Assessment
ARIMA	Autoregressive Integrated Moving Average
ARM	Advanced RISC Machines
ASCI	Accelerated Strategic Computing Initiative
ASR	Average Stability Ratio
AST	Abstract Syntax Tree
ASTM	American Society for Testing and Material
Au	Gold
AWS	Amazon Web Services
BAA	Border Approximation Area
Be	Beryllium
BFR	Brominated Flame Retardant
BLE	Bluetooth Low Energy
BMU	Battery Monitoring Unit
BOINC	Berkeley Open Infrastructure for Network Computing
BWM	Best Worst Method
BYOD	Bring Your Own Device
CCTV	Closed Circuit Television
Cd	Cadmium
CFC	Chlorofluorocarbon
Cl	Chlorine
CO_2	Carbon di-oxide
CoCoSo	Combined Compromise Solution
COMET	Characteristic Objects METhod
COPD	Chronic Obstructive Pulmonary Disease
COPRAS	COmplex PRoportional ASsessment
CPU	Central Processing Unit
Cr(VI)	Hexavalent Chromium
CRT	Cathode-Ray Tube
CSA	Cuckoo Search Algorithm
Cu	Copper
CV	Coefficient of Variation
D2D	Device-to-Device
DDoS	Distributed DoS
DEA	Data Envelopment Analysis
DES	Data Encryption Standard

DF	Degree of Freedom
DHT	Digital temperature and humidity
DIY	Do it yourself
DMU	Decision Making Unit
DNN	Deep Neural Network
DoS	Denial-of-Service
DP	Dew point
DRX	Discontinuous Reception
DSL	Deep Supervised Learning
DSP	Digital Signal Processor
DTM	Dynamic Tone Mapping
DuT	Device under Test
DVFS	Dynamic Voltage and Frequency Scaling
DVS	Dynamic Voltage Scaling
E_3	Energy Efficient Engine
EC2	Elastic compute cloud
EDAS	Evaluation based on Distance from Average Solution
ELECTRE	ELimination Et Choix Traduisant la REalité
EMBB	Enhanced Mobile Broadband
eMMC	embedded Multi-Media Controller
ERP	Enterprise Resource Planning
ESM	Even Swaps Method
FIFO	First-In-First-Out
FLOPS	Floating-Point Operations Per Second
FSM	Finite State Machine
FTP	File Transfer Protocol
GDSS	Group Decision Support System
GFLOPS	Giga-FLOPS
GHz	GigaHertz
GLCM	Grey Level Cooccurrence Matrix
GPGPU	General-Purpose GPU
GPRS	General Packet Radio Service
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRA	Grey Relational Analysis
GSMA	Global System for Mobile communications Association
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
HDMI	High-Definition Multimedia Interface
Hg	Mercury
HI	Heat Index
HL	Hosmer and Lemeshow
HMD	Head-Mounted Display
HMM	Hidden Markov Model
HMP	Heterogeneous Multiprocessing
HPC	High-Performance Computing
HTTP	Hypertext Transfer Protocol
HVAC	Heating, Ventilation and Air Conditioning
I/O	Input/Output
I^2C	Inter-Integrated Circuit
IaaS	Infrastructure as a Service
IaaS	Infrastructure as a Service

IC	Integrated Circuit
ICE	Internal Combustion Engine
ICT	Information and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
IHV	Independent Hardware Vendor
IIT	Indian Institute of Technology
ILP	Integer Linear Programming
IMEI	International Mobile Equipment Identity
IoE	Internet of Everything
IoT	Internet of Things
IPsec	IP Security
ISP	Image Signal Processor
ISV	Independent Software Vendor
IT	Information Technology
ITU-R	International Telecommunication Union Radiocommunication Sector
JSON	JavaScript Object Notation
kWh	Kilowatt hours
LAA	Lower Approximation Area
LAN	Local Area Network
LASSO	Least Absolute Shrinkage and Selection Operator
LC	Local Coordinator
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LHC	Large Hadron Collider
Li	Lithium
Li^+	Lithium ions
LiAB	Lithium-air Battery
LiB	Lithium-ion Battery
LiBOB	Lithium Bis(Oxalato)Borate
Li-Poly	Lithium Polymer
LiSB	Lithium-Sulphur Battery
LMB	Lithium Metal Battery
LPDDR	Low-Power Double Data Rate
LPDDR SDRAM	Low-Power Double Data Rate Synchronous Dynamic RAM
LTE	Long Term Evolution
M2M	Machine-to-Machine
MaaS	MCC-as-a-Service
MABAC	Multi-Attributive Border Approximation Area Comparison
MAC	Media Access Control
MAC-BETH	Measuring Attractiveness by a Categorical Based Evaluation Technique
MAE	Mean Absolute Error
mAH	Milliampere Hour
MARCOS	Measurement of Alternatives and Ranking according to COmpromise Solution
MARE	Multi-Attribute Range Evaluations
MAUT	Multi-Attribute Utility Theory
MC	MCC Coordinator
MCC	Mobile Crowd Computing
MCDM	Multi Criteria Decision Making
MCU	Micro Controller Unit

MD	Message Digest
MEC	Mobile Edge Computing
MEW	Multiplicative Exponential Weighting
MEXP	Mobile Exchange eXperiment Protocol
MIT	Massachusetts Institute of Technology
ML	Machine Learning
MLP	Multi-Layer Perceptron
MMTC	Machine Machine-type Communication
MMTC	Machine-Machine Type Communication
MOORA	Multi-Objective Optimization on the basis of Ratio Analysis
MoS_2	Molybdenum Disulfide
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MPI	Message Passing Interface
MSE	Mean Square Error
Mt	Metric Ton
MULTI-MOORA	Multiplicative MOORA
mW	MegaWatt
NA	Not Applicable
NFC	Near Field Communication
NiB	Sodium-ion Battery
Ni-Cd	Nickel-Cadmium
Ni-MH	Nickel-Metal Hydride
N_2O	Nitrous Oxide
OCR	Optical Character Recognition
OEM	Original Equipment Manufacturer
OLED	Organic Light-Emitting Diodes
OLS	Ordinary Least Squares
OS	Operating Systems
OSPM	OS Power Management
P2P	Peer-to-Peer
PaaS	Platform as a Service
PAPRIKA	Potentially All Pairwise RanKings of all possible Alternatives
Pb	Lead
PC	Personal Computer
PCA	Principal Component Analysis
PCB	PolyChlorinated Biphenyl
Pd	Palladium
PDA	Personal Digital Assistant
PIPRECIA	PIvot Pairwise RElative Criteria Importance Assessment
PIR	Passive Infra-Red
PLS	Partial Least Squares
PMIC	Power Management ICs
POEM	Portable Open-source Energy Monitor
PPI	Pixels Per Inch
PROMETHEE	Preference Ranking Organization METHod for Enrichment Evaluation
PSO	Particle Swarm Optimization
PTC	Positive Temperature Coefficient
PUE	Power Usage Effectiveness
PVC	PolyVinyl Chloride
QoE	Quality of Experience

QoS	Quality of Service
RAFSI	Ranking of Alternatives through Functional mapping of criterion sub-intervals into a Single Interval
RAM	Random Access Memory
RCAS	Railway Collision Avoidance System
RDD	Resilient Distributed Dataset
REMBRANDT	Ratio Estimations in Magnitudes or deci-Bells to Rate Alternatives which are Non-DominaTed
REST	REpresentational State Transfer
RFID	Radio frequency identification
RGB	Red, Green, and Blue
RH	Relative Humidity
RMSE	Root Mean Square Error
RN	Reference Node
RNL	Receptaculum Nelumbinis-Like
ROM	Read Only Memory
ROM	Read Only Memory
RS	Relative Stability
SaaS	Software as a Service
SAW	Simple Additive Weighting
Sb	Antimony
SBC	Single-Board Computer
SC	Service Carrier
SD	Secure Digital
SD-UDN	Software-Defined Ultra-Dense Network
SEI	Solid Electrolytic Interface
SEMO	Smart Energy Monitoring System
SETI	Search For Extraterrestrial Intelligence
SHA	Secure Hash Algorithm
SLA	Service Level Agreement
SMART	Simple Multi-Attribute Rating Technique
SMB	Sodium Metal Battery
SMD	Smart Mobile Device
SMP	Symmetrical Multi-Processing
SoC	System-on-Chip
SP	Service Provider
SPI	Serial peripheral interface
SQL	Structured query language
SS	Service Seeker
SSD	Solid state drives
SSH	Secure Shell
SSIM	Structural-Similarity-index
SSL	Secure Sockets Layer
SVM	Support-Vector Machine
SWARA	Stepwise Weight Assessment Ratio Analysis
Ta	Tantalum
TCAS	Train Collision Avoidance System
TCO	Task Caching and Offloading
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
TWh	Terawatt-hour
UAA	Upper Approximation Area

UART	Universal asynchronous receiver-transmitter
UAV	Unmanned Aerial Vehicle
UDFS	User-Driven Frequency Scaling
UDP	User Datagram Protocol
UE2	User Equipment 2
UI	User Interface
UPS	Uninterruptible Power Supply
URLLC	Ultra-Reliable Low Latency Communication
URLLC	Ultra-Reliable Low Latency Communication
USB	Universal Serial Bus
USB OTG	USB On-The-Go
VC	Vapour Compression
VIKOR	Više Kriterijumska optimizacija i Kompromisno Rešenje
VoIP	Voice over Internet Protocol
VoLTE	Voice over Long-Term Evolution
VPN	Virtual Private Network
VR	Virtual Reality
WAN	Wide Area Network
WASPAS	Weighted Aggregated Sum Product Assessment
WCG	World Community Grid
WebRTC	Web Real-Time Communication
WEP	Wired Equivalent Privacy
Wh	Watt-hour
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
WPM	Weighted Product Method
WSM	Weighted Sum Model
WTD	Wireless Topology Discovery
WTLS	Wireless Transport Layer Security
XML	eXtensible Markup Language
YoY	Year-Over-Year
ZAB	Zinc-Air Battery

PUBLICATIONS

The following publications are the outcomes of the parts and extended works of this thesis. The chapter number indicates that the corresponding paper partially or fully constituted that chapter. The journal metrics are as of the date of acceptance/initial communication (as applicable) of the paper.

Scopus/WoS Indexed Journals

1. PKD Pramanik, S Pal, P Choudhury, Mobile Crowd Computing: Potential, Architecture, Requirements, Challenges, and Applications, ***The Journal of Supercomputing***, Springer. [Indexed in **SCIE**] [**IF 2.557**, SJR .73, **Q2**] [Under review] [Chapter 3]
2. PKD Pramanik, S Pal, M Mukhopadhyay, P Choudhury, Sustainable Edge Computing with Mobile Crowd Computing for Smart HVAC System: A Proof of Concept, ***Sustainable Computing: Informatics and Systems***, Elsevier. [Indexed in **SCIE**, **Scopus**] [IF 4.923, SJR 1.07, Q1] [Under review] [Chapter 9]
3. PKD Pramanik, T Biswas, P Choudhury, Load Balance-Aware Energy-Efficient Scheduling for Mobile Crowd Computing: A PSO-based Solution, ***Sustainable Computing: Informatics and Systems***, Elsevier. [Indexed in **SCIE**, **Scopus**] [IF 4.923, SJR 1.07, Q1] [Under review] [Chapter 6]
4. PKD Pramanik, T Biswas, P Choudhury, Multicriteria-based Resource-aware Scheduling in Mobile Crowd Computing: A Heuristic Approach, ***Journal of Grid Computing***, Springer, 2023. [Indexed in **SCIE**] [**IF 4.674**, SJR 1.18, **Q1**] [Chapter 6]
5. PKD Pramanik, S Pal, P Choudhury, Resource Profiling and Selection for Local Mobile Crowd Computing, ***Wireless Networks***, Springer. [Indexed in **SCIE**, **Scopus**] [IF 2.701, SJR 0.42, Q2] [Under review] [Chapter 4]
6. PKD Pramanik, S Biswas, S Pal, D Marinković, P Choudhury, A Comparative Analysis of Multi-Criteria Decision-Making Methods for Resource Selection in Mobile Crowd Computing, ***Symmetry***, 2021. [Indexed in **SCIE**, **Scopus**] [IF 2.713, SJR 0.39, Q2] [Chapter 5]
7. PKD Pramanik, N Sinhababu, KS Kwak, P Choudhury, Deep Learning-based Resource Availability Prediction for Local Mobile Crowd Computing, ***IEEE Access***, 2021. [Indexed in **SCIE**, **Scopus**] [IF 3. 367, SJR 0.59, Q1] [Chapter 7]
8. PKD Pramanik, N Sinhababu, A Nayyar, M Mashud, P Choudhury, Predicting Resource Availability in Local Mobile Crowd Computing Using Convolutional GRU, ***Computers, Materials and Continua***, vol. 70(3), pp. 5199-5212, 2021. [Indexed in **SCI**, **Scopus**] [IF 4.89, SJR 1.53, Q1] [Chapter 7]
9. PKD Pramanik, P Choudhury, Mobility-Aware Service Provisioning for Delay

Tolerant Applications in a Mobile Crowd Computing Environment, ***SN Applied Sciences***, vol. 2(3), article no. 403, 2020. [Indexed in **ESCI**, **Scopus**] [Chapter 8]

10. PKD Pramanik, G Bandyopadhyay, P Choudhury, Predicting Relative Topological Stability of Mobile Users in a P2P Mobile Cloud, ***SN Applied Sciences***, vol. 2(11), article no. 1827, 2020. [Indexed in **ESCI**, **Scopus**] [Chapter 8]
11. PKD Pramanik, S Pal, P Choudhury, Green and Sustainable High-Performance Computing with Smartphone Crowd Computing: Benefits, Enablers, and Challenges, ***Scalable Computing: Practice and Experience***, vol. 20(2), pp. 259-283, 2019. [Indexed in **ESCI**, **Scopus**] [SJR 0.18, Q3] [Chapter 1]
12. PKD Pramanik, N Sinhababu, B Mukherjee, S Padmanaban, A Maity, BK Upadhyaya, JB Holm-Nielsen, P Choudhury, Power Consumption Analysis, Measurement, Management, and Issues: A State-of-the-art Review on Smartphone Battery and Energy Usage, ***IEEE Access***, vol 7(1), pp. 182113-182172, 2019. [Indexed in **SCIE**, **Scopus**] [IF 4.098, SJR 0.61, Q1] [Chapter 3]

International Conferences

13. PKD Pramanik, N Sinhababu, A Nayyar, P Choudhury, Predicting Device Availability in Mobile Crowd Computing using ConvLSTM, *7th International Conference on Optimization and Applications (ICOA 2021)*, May 2021, Wolfenbüttel, Germany, **IEEE**. [Indexed in **Scopus**] [Chapter 7]
14. PKD Pramanik, P Choudhury, A Saha, Economical Supercomputing thru Smartphone Crowd Computing: An Assessment of Opportunities, Benefits, Deterrents, and Applications from India's Perspective, *4th International Conference on Advanced Computing and Communication Systems (ICACCS-2017)*, January 2017, pp. 1-7, Coimbatore, India, **IEEE**. [Indexed in **Scopus**] [**Best paper awarded**] [Chapter 3]

Book Chapters

15. PKD Pramanik, S Pal, P Choudhury, Smartphone Crowd Computing: A Rational Approach for Sustainable Computing by Curbing the Environmental Externalities of the Growing Computing Demands, in *Emerging Trends in Disruptive Technology Management for Sustainable Development,* [eds.] R Das, M Banerjee, S De, pp. 45-80, **Chapman and Hall/CRC**, New York, 2019. [Chapter 1]
16. PKD Pramanik, S Pal, G Pareek, S Dutta, P Choudhury, Crowd Computing: The Computing Revolution, in *Crowdsourcing and Knowledge Management in Contemporary Business Environments*, [ed.] R Lenart-Gansiniec, pp. 166-198, **IGI Global**, 2018. [Chapter 3]

17. PKD Pramanik, P Choudhury, IoT Data Processing: The Different Archetypes and their Security & Privacy Assessments, in *Internet of Things (IoT) Security: Fundamentals, Techniques and Applications*, [eds.] SK Shandilya, SA Chun, S Shandilya, E Weippl, pp. 37-54, **River Publishers**, Gistrup, Denmark, 2018. [Indexed in **WoS BkCI**, **Scopus**] [Chapter 9]
18. PKD Pramanik, S Pal, A Brahmachari, P Choudhury, Processing IoT Data: From Cloud to Fog. It's Time to be Down-to-Earth, in *Research Anthology on Architectures, Frameworks, and Integration Strategies for Distributed and Cloud Computing*, [ed.] Information Resources Management Association, pp. 920-943, **IGI Global**, 2021. [Chapter 9]
19. PKD Pramanik, S Pal, P Choudhury, Beyond Automation: The Cognitive IoT. Artificial Intelligence Brings Sense to the Internet of Things, in *Cognitive Computing for Big Data Systems Over IoT: Frameworks, Tools and Applications*, [eds.] AK Sangaiah, A Thangavelu, VM Sundaram, Lecture Notes on Data Engineering and Communications Technologies, vol. 14, pp. 1-37, **Springer**, 2018. [Indexed in **WoS**, **dblp**] [Chapter 9]

1

Introduction

"There must be a better way to make the things we want, a way that doesn't spoil the sky, or the rain or the land." --- Paul McCartney

1.1 Escalating Environmental Perils

In recent years, the impact of the changing environment and climate has been experienced worldwide and, on many occasions, severely. The effect ranges from Antarctic glacier melting to the expansion of the Sahara Desert. For instance, in the year 2017, the U.S. was hit by several devastating natural disasters that range from floods and hurricanes to droughts and wildfires. Let us take a look at some of them, which caused major damage and indicate the operation of the environment [1]:

- Hurricanes Irma, Harvey, and Maria: The U.S. was hit by three category-4 hurricanes during 2017. The storm and the accompanying rainfall led to an unprecedented rise in sea level, resulting in a devastating flood. Hurricane Harvey was fuelled by record-breaking rainfall (reportedly, 1-in-25,000-year rain).
- Atmospheric river storms in California: Throughout the winter season of 2017, different parts of California were hit by several back-to-back extreme atmospheric river storms that produced record rainfall and flooding in the state.
- Spring snow/rainfall and floods: In the mid-March, the Northeast U.S. received snowfall at an astonishing rate of 7 inches per hour totalling to 42 inches, thanks to the winter storm Stella. In the late-April, the Midwest U.S. was devastatingly flooded due to heavy rainfall of up to 15 inches. The percentage of heaviest 1% of rainy and snowy days has been increased from by 53 and 92 in the Midwest and the Northeast U.S. respectively during the years 1958 through 2016.
- High plains flash drought: In the July, flash drought gripped the Dakotas and Montana that led to one of Montana's worst wildfires causing agricultural losses of $2.5 billion.

- California experienced a record heat: On September 1, the temperature in San Francisco reached 106°F breaking its all-time heat record. Actually, from late August through early September, the whole California experienced the worst state-wide heat wave ever recorded, and many parts of it broke daily, monthly, and all-time temperature records. In fact, heat waves have become more frequent across the U.S.
- Plains on fire: A few weeks after the record heat wave in September, California was hit by the deadliest and most destructive fires in state history, killing 40 people and destroying a total of 8,323 structures. Earlier, around in March, places like Kansas, Oklahoma, Colorado, and Texas were blazed by major fires. The fire of Oklahoma was the largest wildfire on record in the state breaking the previous record just set one year prior. These fires are not one-off incidents. The wildfire incidents are consistently increasing over the years in the western U.S. grasslands or the Great Plains region. Since the 1970s, in every decade, more than 100,000 acres of extra grass and shrubland caught fire than the previous decade. In a study covering over a three-decade period (1984-2014), it is observed that the total area burned by large wildfires in the Great Plains rose by 400%.
- Summer in winter: Generally, if the ratio of days that record the highest temperature and the days that of lowest temperature is evenly balanced, the climate is supposed to be stable. But for the past three decades, the balance has been disrupted. Due to the consistently warm climate, the number of record high-temperature days have begun to outpace the number of record low-temperature days. In the month of February, there were 34 instances of heat-record breaking for every cold-record breaking in different parts of the U.S. As a result of this growing imbalance, the February was one of Chicago's warmest on record.
- Reduced snowing in winter: Since 1884, Chicago had the least snow cover during the months of January and February. This is attributed mainly to the unseasonably and atypical warm and rainy weather. The climate change caused a significant reduction in snow cover extent over high northern latitudes during the last 100 years.

The worldwide impact of environmental idiosyncrasies has forced us to deliberate on to figure out the reasons for these abnormal behaviours of nature. Is there any common and explainable reason behind this? Yes, the main and explainable reason behind this is suspected to be pollution and global warming.

The phenomenon of global warming and climate change is largely man-made and has been accelerating at a rapid and unprecedented way since the Industrial Revolution began in the late 1700s [2]. According to NASA, since 1880, the average temperature of the Earth has risen 0.8 °C which is projected to increase further, according to the U.S. Environmental Protection Agency (EPA), between 1.13 and 6.42 °C over the next 100 years.

The main reason for the warming of the Earth is the greenhouse gas effect that obstructs the infrared and heat radiation to escape from Earth toward space. Gases that contribute most to the greenhouse effect include water vapour, CO_2, Methane, N_2O, CFCs, etc. Most of these gases are human-produced and are responsible for increasing the Earth's temperature over the past 50 years. Among these, CO_2 is the most common greenhouse gas in the atmosphere. For instance, in 2012, CO_2 reportedly accounted for nearly 82% of all greenhouse gas emissions in the U.S. In the last 150 years, mainly due to the industrial activities, the atmospheric CO_2 levels are raised from 280 ppm (parts per million) to 410 ppm which is further expected to be degraded to 450 ppm by 2035 unless greenhouse gas emissions are controlled strictly [3].

Electricity generation is one of the major sources of carbon pollution, because in most of the countries, still today, the majority of the electricity is generated by burning fossil fuels. For example, even a developed nation like Australia gets 73% and 13% of its electricity by burning coal and gas, respectively [4].

1.2 Environmental Impact of ICT

Besides other industries, the rapid advancement of the ICT industry (that includes computers and peripherals, computer and telecommunication networks and associated equipment, and the data centres) has soared the energy consumption like never before. At present, globally, nearly 10% of the total energy is consumed by

the ICT industry. Factually, the total global energy demand is estimated at 20,000 TWh, whereas ICT is accountable for using 2,000 TWh [5]. This huge energy consumption produces roughly 1.7% (530 Mt) of the total CO_2 emissions [6] ICT's carbon footprint is roughly equal to the carbon emission from the aviation industry's fuel burning. Experiencing the ultra-penetration of ICT into every sphere of human life that results in increased energy consumption rate by 20% per year, it is expected that the world's energy consumption of ICT will be double by 2030. Out of total energy consumption by ICT, two-thirds are attributed to the devices, data centres while the rest goes for the telecommunication networks.

The production and use of the ICT commodities have triggered several negative impacts on the environment. The major concerns are discussed in the following.

Use of natural resources: Use of natural resources in the production of the ICT products has a reason for natural resource depletion from the Earth; thus, unbalancing the natural diversity. To back this argument, let us check out the following statistics [7]:

- The amount (in terms of weight) of fossil fuel and chemicals required in manufacturing an average desktop computer is at least 10 times of its own weight. This ratio is much more than in the case of an automobile or refrigerator, which require fossil fuels by 1-2 times their weight.
- To make a microchip, on average, 16000 litres of water, 1.6 kg of fossil fuel, 0.7 kg of chemicals are used [8] [9] while making a computer with a 17-inch CRT monitor it accounts for 1500 litres of water, 240 kg of fuel, 22 kg of chemicals which costs a total material of 1.8 tons [10] [11].

Energy consumption: Device production and operations consume huge energy. For example, nearly 30,000 megajoules of energy is used in the manufacturing of an average computer. The energy consumption demands more energy production, which increases the carbon footprint [12].

Effects of the manufacturing process: The production of computer hardware causes havoc pollution. The different parts of a computer and its peripherals contain several harmful heavy metals. Along with the environment, these toxic heavy

metals are really dangerous to human and animal health. Long-term exposure to these elements may be fatal to the workers and their families and also the neighbouring communities. Some of the most hazardous metals that damagingly effect hominoid health are:

- Antimony (Sb): Immediate contact to antimony may cause aggravated irritation of the eyes, skin, and lungs. Long-term exposure to this toxic metal can impend stomach pain, diarrhoea, vomiting, stomach ulcers, pulmonary edema (swelling due to the accumulation of interstitial fluid in an organ or any area of the body), chronic bronchitis, chronic obstructive pulmonary disease (COPD, includes both chronic bronchitis and emphysema), pneumoconiosis, altered electrocardiograms, spontaneous abortion, and menstrual irregularities.
- Arsenic (As): Arsenic is one of the most toxic metals found in the Earth ground. It has severe impacts on human health. Long-term exposure to high levels of arsenic is highly cancerous and is one of the main reasons for skin, bladder, and lung cancer. Arsenic is also associated with heart disease. Small amounts (<5 mg) of arsenic ingestion (through water or pesticides/ insecticides) cause nausea, vomiting, abdominal pain, and diarrhoea. Acute poisoning due to a lethal dose of arsenic (100 mg to 300 mg) may lead to death.
- Beryllium (Be): Exposure to beryllium fumes and particles causes chronic beryllium disease (a fatal respiratory disease). Beryllium also has the potential to harm different organs like the liver, kidneys, heart, and nervous system. This carcinogen metal may cause lung cancer also.
- Cadmium (Cd): Cadmium is a highly toxic element, and if inhaled in excessive level can cause death. Long-time exposer cadmium can damage kidneys and bones. Excessive exposure may harm lung functions and increase the risk of lung cancer.
- Chromium (Cr): Chromium compounds affect the respiratory tract badly resulting in diseases like asthma, chronic bronchitis, chronic irritation, chronic pharyngitis, chronic rhinitis, congestion and hyperaemia, polyps of the upper respiratory tract, tracheobronchitis, etc. High dose of chromium exposure may

even lead to lung, nasal, or sinus cancers. Cases of sperm damage and the male reproductive system also been observed as a result of chromium exposure.

- Cobalt (Co): Though cobalt is beneficial for humans because it is a metal constituent of vitamin B12, high concentrations of cobalt may promote various adverse health effects. High concentrations of cobalt may affect human health, causing vomiting and nausea, vision problems, heart problems, and thyroid damage. As per clinical experiments, cobalt has also been classified to be carcinogenic.
- Lead (Pb): Lead affects the kidneys and reproductive systems. Even low levels of lead can be harmful to a child's nervous system and mental development.
- Mercury (Hg): Mercury is linked to brain and kidney damage. It also affects the nervous, digestive, and immune systems. Mercury is seriously harmful to the developing foetus and young children, affecting the nervous and cognitive system.
- Selenium (Se): Selenium is known to have many benefits (mainly due to its antioxidant properties) to human health if it is consumed in moderate level. But a high dose of it has several adverse health effects. Overexposure of selenium may cause an accumulation of fluid in the lungs. Selenium is also attributed to health menaces like bad breath, bronchitis, bronchial asthma, shortness of breath, nausea, vomiting, abdominal pain, diarrhoea, enlarged liver, conjunctivitis, and pneumonitis. High concentrations of selenium are associated with skin cancer, prostate cancer, and diabetes. High enough levels of selenium can be the cause of death.

In addition to the above-mentioned, other metals such as aluminium (Al), barium (Ba), copper (Cu), gallium (Ga), gold (Au), iron (Fe), manganese (Mn), palladium (Pd), platinum (Pt), silver (Ag), and zinc (Zn) are also used in manufacturing a PC. Exposure to these metals in considerable amount is harmful to organisms.

The chemicals involved in the production of computers also damage the environment and the health of living beings. For example, nitrogen trifluoride (NF_3), used in LCD, thin-film photovoltaic cells and microcircuit manufacturing, has 17,000 times greater potential to cause global warming as compared to CO_2 [13]. BFR,

another important substance used in computer production, may lead to thyroid damage and undeveloped foetus. The oil-based paints that are used for the finished products are also extremely toxic in nature. All of these metals and chemicals and toxic materials causes water contamination and air pollution damaging the global environment.

Burden of hazardous e-waste: We are experiencing an e-waste tsunami. E-waste, one of the fastest-growing types of waste worldwide, has become a serious threat to the Earth. Globally, in 2014, the per inhabitant e-waste generation was recorded as 5.8 kg, which had been increased to 6.3 kg in 2017 and is expected to reach 7.0 kg by 2022 [14] [15]. Worldwide, 20 to 50 million tons of e-waste are generated every year [16]. The increase in production and buying of computing devices, along the changing technology has seriously contributed to increasing electronic waste. As a matter of fact, approximately 90% of the discarded computer accessories are not recycled but dumped openly.

Among the total solid waste deposited in landfills, 70% of the hazardous waste is accounted to e-waste [16]. This huge amount of e-waste releases a substantial amount of toxic materials, volatile organic chemicals, and heavy metals which not only exhaust resources but causes environmental pollution and global climate change. The toxic elements due to improper waste disposal pollute the soil, making them infertile which become impotent to support crops and other plant life [17]. This deters the production of foods, which eventually leads to malnourishment of the natives and the nationals. Furthermore, the contaminated food farmed on the polluted soil may be the source of serious illness.

Many often, the e-wastes are sent to the developing countries to be dumped in the landfills. People extract valuable materials such as gold, silver, and copper from the discarded electronics by burning the substances. This produces hazardous gas and smoke (due to the presence of other toxic materials) by which not only the air but water also gets polluted.

Industrial discharge: Untreated industrial discharges like oil, toxic chemicals, and sewage contaminate the water bodies like rivers and lakes. The polluted water

is dangerous for the aquatic creatures. For instance, over 8,000 marine lives were reported dead six months after the disastrous Deepwater Horizon oil spill in 2010 that affected 16,000 miles of U.S. coastline [17]. Also, consuming the fish and seafood from the contaminated water can have serious health effects, especially to children and pregnant women. Besides, chemical fumes, smoke, and other industrial emission pollute the air. Moreover, the solid discharge from industry is huge, and most are nondegradable.

The environmental impact of the production of computers and mobile devices (e.g., tablets and smartphones) is so immense that, to equalise it, we would have to use each device for between 33 and 89 years [18] [19].

In line with the obligations for complex problems and applications, along with the massive increase in big data generated from innumerable sources, the need for high-performance computing (HPC) has increased enormously. And more requirement for computers leads to more productions and more uses of computers which means more environmental hazards and pollution.

If we do not reconsider our device consumption model and carry on at the current pace, it is supposed that by 2050 we might need 8.5 planets to absorb the carbon monoxide and 6, 3.5, and 3.5 planets to meet demands for steel, cement, and wood, respectively [20].

1.3 Sustainable Computing

Whatsoever the negative impacts of computers have on the environment; we cannot head them off from our livings. We need them in every step of our daily life. Actually, we need more and more powerful computers day by day for various purposes. In view of that, we need to consider seriously to minimize the environmental impacts of producing and using computers. Altogether, to mitigate the environmental hazards due to computing devices, we need to concentrate on green and sustainable computing.

1.3.1 Defining Sustainable Computing

Sustainable computing is a methodology that embraces a range of policies,

procedures, programs, and attitude for using information technology (IT). It is a holistic approach that includes power control and management, wastage management and education concerning the deployment of IT. The concept of sustainable computing considers the total ownership cost, environmental impact and the benefit of the technology. It should consider minimizing the use of hazardous materials, maximizes energy efficiency during the product's lifetime, and recyclability of the product and the factory waste. Sustainable computing is important for all classes of systems, ranging from handheld systems to largescale data centres.

1.3.2 Elements of Sustainable Computing

The four aspects, as shown in Fig. 1.1, are considered as the core elements of sustainable computing. In the following, we discuss them in brief.

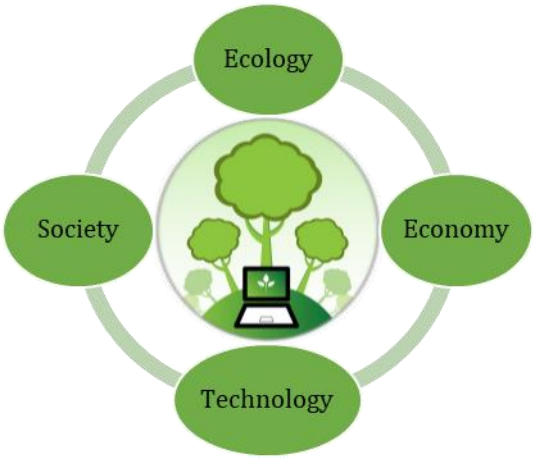


Fig. 1.1. Elements of sustainable computing

Society: The society is one of the important key elements of sustainable computing. The ever-increasing demand for computer leads to manufacturing and purchase puts a lot of negative effects on the environment. But it is the people of society whose careful selection of computing devices and judicious use and management may minimize the negative impact on the environment. People wisdom and awareness could possibly reduce the carbon footprint and conserve energy.

Economy: The environment pays the price of a rising economy. Today's world economy is changing rapidly. These changing economies force the use of ICT hugely at all levels of business processing. Meeting the big market demand, industries are also bringing new technologies every other day. This fast-changing

economy, thus, puts a negative impact on the environment. Stringent business policy, supporting green computing model would enable to reduce the negative impact. Perhaps, well-calculated and measured policies would able to restrain the negative effect of the economy on the environment.

Ecology: As already discussed in Section 1.2, excessive use of computers over the last two decades has resulted in millions of tons of e-waste. This e-waste is continually damaging the environment by contaminating the soil and water. Besides, computer manufacturing has increased air and water pollution. Further, extensive usages of computers ensue enormous power consumption resulting in more greenhouse gas emission. The overall negative impact on the environment is disrupting the ecological balance, thereby changing the marine and land life, vegetation and climate. The water and air pollution profoundly impacted the land and water species, including the endangered species. The toxic in water or soil may pass through food may cause genetic and neurological changes which may pass on to generations in larger animals, including human beings. Governments, civic bodies, and industries have important roles to play in tackling the endangering e-waste management problem. Strict government and administration guidelines needed to check the abundant use of toxic materials and dumping and releasing e-wastes in the open lands. Some general responsibilities of governments and industries are listed in Fig. 1.2 and Fig. 1.3, respectively.

Technology: As we progress, the environmental condition continues to deteriorate due to the adverse consequences of the industrial development on the environment. The technological advancement in every sector has made it worse due to more demands from people. In other words, technology is one of the culprits behind the ill environmental health. But considering the stage at which we are now, the only hope is also the advanced technologies towards sustainable developments. Sustainable technology and sustainable computing are important components of that. Fig. 1.4 shows the goals for sustainable technologies and sustainable development.

Fig. 1.2. Government's responsibilities in e-waste management [21]

1.4 Computational Measures Adopted for Sustainable Computing

To attain sustainable computing, the computing fraternity mainly focussed on energy efficiency. The improvement of energy efficiency of any given computer system without affecting the reliability factor is a major challenge to overcome in almost all the computing domains, be it a low power embedded device or a large-scale server. Here, the key concern is the measures regarding how to reduce the power consumption where the fault tolerance technique needs computation and state redundancy, increasing the power consumption and a balanced trade-off between them. The trade-off can be managed by combining the techniques that comprise of both hardware as well as the software where it is literary impractical to concentrate over a single component or a level of the system on attaining adequate power consumption and as well as reliability. In the following, we discuss a few

such computational measures, aiming to minimise the energy consumption of the computers and data communication.

Fig. 1.3. Role of industries and corporates in e-waste management

Low-power processors: For sustainable computing, it is essential that processors consume low power. Processors in computing devices consume a considerable amount of energy. This energy consumption varies linearly with the processor clock speed. To process, the increasing jobs demand clock speed is increased, resulting in high energy consumption. But not all the jobs require a powerful processor. Submitting the low-end jobs to the low-end processor will save significant energy. Even when the processor is idle, it consumes energy. In that case, also, a low power processor will waste less power. The development of low-power processors allows them to use very less battery power. Due to their low power consumption characteristics, they are suitable for mobile-based computing for a longer duration. It is an ongoing challenge to best fit the performance with power consumption. The chip designers are struggling to attain the most appropriate power-performance balance. The processor circuit is reduced, and the distance between the

interfacing circuits are minimized to reduce energy consumption. Often the memory and the input and output port are onboard fabricated to reduce the power consumption. Manufacturers like Intel and AMD are putting forward new processor technologies where a single processor can do the job of multiple processors consuming an equal amount of energy. The multi-core processor technology (dual, quad, or octa-core processors) enhances the computing performance significantly by enabling parallel computing capability in a single processor package. The multi-core processor reflects as multiple processors working together with performance very higher than a single processor at lower clock speeds. The voltage consumption per core is less and thus, typically consuming less power [22].

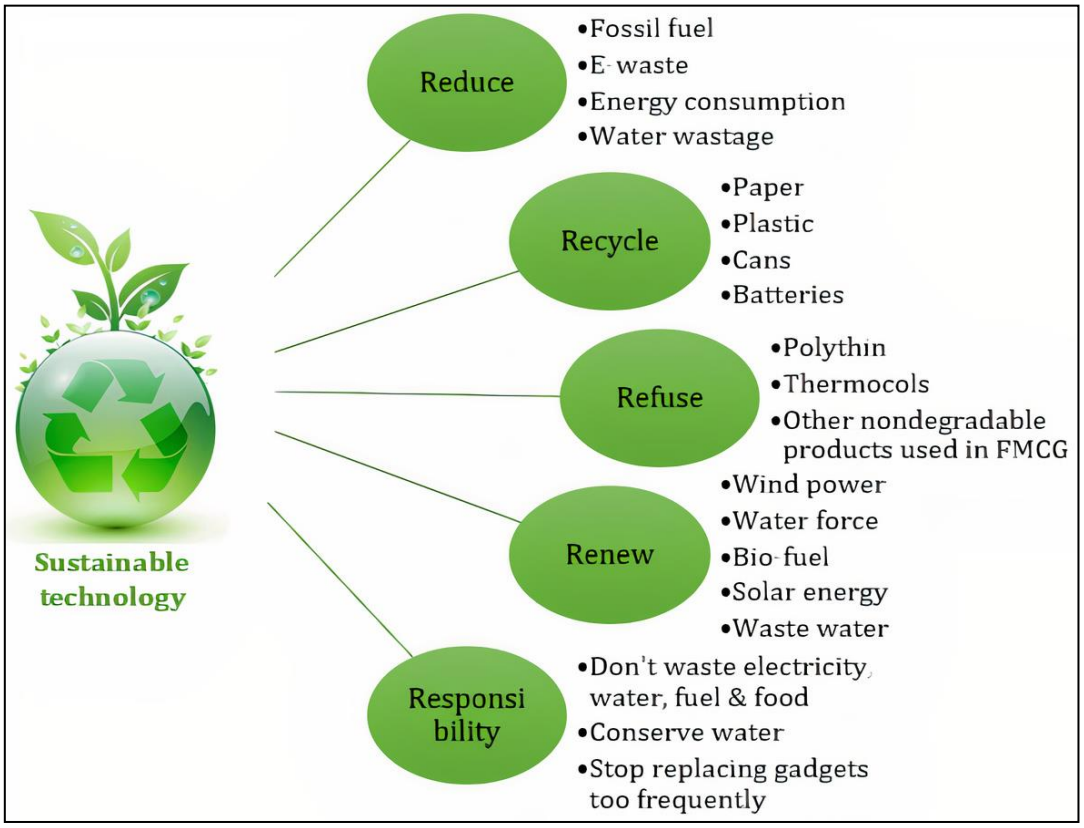


Fig. 1.4. Goals for sustainable development [23]

Energy-efficient storage: Secondary storage devices (e.g., hard drive) are electro-mechanical devices which consume huge energy while accessing data from the magnetic disc. The magnetic plate sizes, the speed of data accession, read-write head movement, and data transfer is some of the factors which affect energy consumption. The development of a new storage device like SAS (Serial Attached SCSI) with the advantage of a 2.5-inch plate size model provides high performance with

less energy consumption in comparison to a traditional 3.5-inch model. Similarly, for less I/O-intensive applications, SATA (Serial ATA) provide high yield with less power consumption [22].

Algorithmic efficiency: The efficiency of an algorithm counts on the computational resources used by the algorithm. The increasingly complex algorithms need more space and time, which increases processor cycles consumption and thus the power. To attain maximum energy efficiency of an algorithm, its resource utilization needs to be minimized. For attaining sustainable computing, it is necessary that the algorithms used in computation job are energy efficient or could be said to have the requirement of less hardware. For example, the time complexity of a hashing-based search is very less as compared to a linear search. This ensures hashing based search uses less processor cycle and hence consumes less energy. In a study at Harvard, it is found that 7 grams of CO_2 are produced for an average Google search, which Google doubts and claim it to be 0.2 grams. Irrespective of the claims, it is clear that an inefficient algorithm in terms of resource complexity could lead to the consumption of huge energy. Thus, for having sustainable computing, the energy efficiency of the algorithm should be considered as one critical parameter. Switching to an efficient algorithm would be a sustainable solution for energy-efficient computing.

Efficient resource allocation: Processing job requires various computational resources like processor, memory (internal and external), I/O devices, and other devices. For maximizing computational productivity, it demands an efficient strategy of resource allocation. The processes executing in parallel often may require resources which may be shared and held by other processes. A process holding a resource while not using it consumes a lot of resource energy. Efficient and intelligent resource allocation may help to solve problems like starvation and deadlock situation. The optimal resource allocation strategy ensures resources are properly allocated on time and requirement basis to processes and are properly released, thus saving extra power consumptions.

Energy-efficient routing: Routing is an optimization task of selecting an efficient and reliable network path for routing data packets. The various criteria for optimal

path selection depend upon the distance between source and target, network bandwidth, shortest delay, and constraints like limited node power, restricted wireless link capacity, etc. It is seen as the number of hops increases, the network path selection and transmission of data packets through different hops makes routing energy consuming. In sustainable computing, it is crucial that energy efficient routing protocols are used which uses fewer hops for delivering the packets.

Energy-efficient display: In a computer system, in comparison to the other components and peripherals, the display device (monitor) consumes the most energy. Even when the computer is idle, the display device continually keeps consuming energy. For sustainable computing, display devices should be energy efficient. There are two ways seen for reducing energy consumption by display devices. One is integrating low power consumption technology for display, and other is efficient power management, which makes sure the display device hibernates when it is in an idle state. Earlier, the use of CRT technology consumed a lot of energy, but their replacements by LCD and subsequently, LED technologies have reduced the power consumption considerably. Further, in comparison to LCD monitors which typically use a cold-cathode fluorescent bulb to provide light for the display, the LED monitors use an array of LEDs. Thus, LED reduces the amount of electricity used for display; moreover, LEDs are mercury-free and nontoxic as compared to LCDs.

Operating system support: The importance of designing an energy efficient system has gained attention with the proliferation of portable and battery-operated devices, e.g., laptops, PDAs, mobile phones, etc. Various hardware solutions have been proposed as a method to minimize energy consumption where the energy-efficiency in terms of software solution is comparatively unexplored yet. As software is the driving force behind given hardware, the decisions undertaken during software designs generally have a major impact on the overall system energy consumption. OS as system software manages the different components and resources of a computing device. From the research perspective, apart from the memory management in the OS, the remaining areas were never given focus in respect to energy efficiency. One of the functionalities of the OS is resource accession and scheduling them for use. Most of the time, when the computing system is idle or

not in use, the OS continually keeps accessing the different resources, and this makes the computer to consume energy continually. In sustainable computing, the OS must be energy efficient where the computing resources must be cleverly and efficiently used to avoid unnecessary energy consumption.

Efficient power management: Hardware stuffs in a computer, consume huge energy even when they are not in use (but kept on). For sustainable computing, it is absolutely necessary that the computer conserves energy. The criteria of power management for devices like CPU, GPU, and computer peripherals, e.g., monitor, printer, etc. that they are able to manage the power efficiently by turning off or switching to a low-power state when non-active. Several efficient power management techniques available that make computers [24], HPC systems [25], data centres [26], and mobile devices [27] [28] [29] energy-efficient. For efficient power management, the computer hardware devices abide the ACPI, an open standard, which allows the operating system to control and manage the device power directly, and hence when not required are set to off. CPU generally consumes high power with an increase in job processing and also cause heating and thus extra power is required for cooling. New power management programs called 'undervolting' allows setting the CPU power manually. There are automatic undervolt programs available which automatically increases the CPU power on demand like "SpeedStep" on Intel processors, "PowerNow!" or "Cool'n'Quiet" on AMD chips, LongHaul on VIA CPUs, and LongRun with Transmeta processors. Currently, most servers consume approximately 70% of the maximum power even when they are idle and consumes 80% of their maximum when they are working at 20% of their peak utilization. In the server, power management is disabled to keep up the response time and performance. But, enabling the processor power management may allow the server to save energy consumption up to 50% in the idle state [22]. Highly efficient and properly designed power supplies reduce the power loss within a server, which results in significantly less energy consumption and heat generation while in operation.

1.5 Sustainable Computing Paradigms

In the previous section, we checked out some computing measures that are

discretely applied to attain energy efficiency. In this section, we discuss the computing system paradigms or approaches that as a whole would facilitate to attain sustainable computing. One of the strategies to attain sustainable computing is to utilise the existing resources optimally and fully, minimising the requirement of new devices, which ultimately would reduce the environmental impact caused by the production process and the e-waste. In the following, we briefly discuss these approaches while their environmental advantages and the associated issues are summarised in Table 1.1.

Grid computing: Grid computing is a distributed system that allows seamless access to a computing grid made of a collection of computing resources connected through a network. Grid computing offers supercomputing like computing power utilizing intra- and/or inter-organizational computing resources such as desktops, clusters, RAIDs, etc. It can impressively save organizations' IT budget. Instead of spending on third-party computing resources (e.g., the cloud) organizations can make use of their existing IT infrastructure. In-house computing resources can be utilized to form a grid or pool of resources. For sustainable computing, grid computing is a suitable approach. The flexibility to adopt different computational devices supports in reusing the idle heterogeneous devices for computing by the process called CPU cycle stealing. This is an excellent feature which makes use of existing unused active computing devices (desktop computers, clusters and supercomputers) which otherwise in their idle state wastes enormous processing cycles as well as energy. The Grid, on the basis of the job requirement, scales up its computational power from the available connected devices. This makes sure that only the required number of computational devices are used without keeping the entire resources on hold. This, in comparison to other HPC facility (supercomputers), makes sure that the cost and energy are saved and would allow reusing the existing computing infrastructure at their best.

Cloud computing: The concept of cloud computing may be stated as a shared pool of configurable computing resources along with quality services which can be rapidly provided on-demand basis with limited effort [30]. The cloud computing services (hardware and software) can be scaled to any number of computer

requests and thus eliminate the need for private data centres. For a business organization, the cost incurred for subscribing cloud service is comparatively very less than maintaining private data centres. The cloud technology, by its resource sharing approach, has actually discouraged business enterprises for opting private data centres. Thus, reducing the number of data centres has significantly contributed to energy saving. In this view, recently, Google has claimed that the use of cloud technology has reduced existing data centres power consumption by 50%.

Serverless computing: The concept of cloud computing has brought the serverless model, which allows dynamic handling and allocation of machine resources (hardware and services) on-demand basis. This eliminates the cost for purchasing and maintenance of privately-owned servers. The cloud technology allows resource sharing, which makes optimum use of cloud resources in parallel for multiple purposes on a large scale. Serverless computing adds another layer of abstraction atop cloud infrastructure. It can be assumed as the more exclusive version of PaaS in cloud. In PaaS, a minimum set of resources must be maintained at the client's end, whereas in serverless computing everything is deported to the remote server. The developers are freed from worrying about anything but their runnable code and functions which should be run and tested in the cloud server [31]. As per the on-demand service provisioning principle of cloud, serverless computing is also able to scale up quickly by spawning new instances of resources as they are requested. Moreover, it also scales down quickly by shutting resources down when they are not required or if their use period is exhausted. This saves a lot of energy consumption. Serverless codes need not be run in any specific server; rather, they can run anywhere through the Internet. This means the serverless applications can be deployed in the edge of the network that is close to the end users [32]. This will not only reduce the latency but also saves a significant amount of energy by eliminating the need for unnecessary data transmission [30]. Individual private servers consume huge energy. The ratio of job processing to energy consumption is disproportionate, with immense energy is consumed while the server remains underused. Serverless computing saves energy consumption in running those servers. Therefore, this model is considered a key approach to sustainable computing.

Using terminal servers: The concept of terminal servers contributes to green computing. The use of terminal server along with the thin clients gives users the impression that the computation is carried in the very same terminal, while the actual computation takes place in the terminal server. The thin client uses up to the 1/8th amount of energy in comparison to workstations and thus considerably reduces energy consumption. There has been an increase in the use of terminal servers and thin clients to create virtual labs. The terminal server software include Terminal Services (now Remote Desktop Services) for Windows platform and Linux Terminal Server Project (LTSP) for the Linux platform while Windows Remote Desktop and RealVNC can provide a thin client.

Virtualization: Provision of assigning workloads to the servers on one-to-one basis may cause resource underutilization. This can be avoided by virtualization. Virtualization is the process of logically parting a server into multiple virtual servers or server instances sharing the same hardware resources and allows processing multiple applications or jobs on different virtual servers. Furthermore, virtualization supports the distribution of works among virtual server instance, ensuring the server resources are used effectively. Dedicated application server consumes more resources than is justified by their workload. By virtualization, a physical server acts as multiple server instances, consuming less energy in comparison to separate dedicated servers. It is seen with this process of server consolidation up to 25% of power can be saved [22]. For performing virtualization requires better hardware resources as a high-end processor, good memory and storage, ensuring performance effectiveness of virtual servers. The concept of virtualization was first conceived by IBM in the 1960s for mainframe computers, and later on, the concept was implemented for x86 computers in 1990s. Many software companies have come up with software solutions for virtualization; this includes Linux container which effectively uses resources to reduce energy consumptions. Microprocessor manufacturers like Intel and AMD have incorporated virtualization enhancements to the x86 processor for supporting virtual computing.

Among these five approaches discussed above, grid and cloud computing are the most prominent initiatives which have minimised the requirement of owning

personal computer systems considerably. They have also replaced the need for centralised HPC systems such as supercomputers and mainframes to some extent.

Though grid computing intends to fully utilise the existing resources, cloud computing does not intend to do so. Cloud computing needs additional resources to provide cloud services. The centralised resources such as data centres, at the cloud service provider's end, consumes massive energy leading to greenhouse gas emission substantially. In fact, data centres capture one-third share of the total energy consumption of ICT. A well-known report [33] of the year 2010 stated that the electricity consumption by data centres had risen by 56% in five years; whereas, during the same period, the overall increase in U.S. electricity usage was only 36%. This statistic reflects the seriousness of the energy requirements of the data centres, which has become more severe in recent years. If the right measures are not taken in due course, data centres are sure to become a grave threat to the environment.

Table 1.1. Environmental advantages and issues of the sustainable computing approaches

Computing approaches	Environmental advantages	Issues
Grid computing	Utilising existing idle resources facilitates less energy consumption and minimise environmental hazards due to the manufacturing and operation of the computing systems required otherwise.	Needs fast and reliable LAN and WAN connections. In the volunteered grid, it is difficult to motivate the resource owner to lend their resources. In the case of the non-volunteered grid (e.g., commercial grids), the services might be costly. Not only setting up and managing the grid resources but also accessing them often require expertise.
Cloud computing	It eliminates the need for private computers, servers, and data centres which has significantly contributed to energy saving.	The data centres, comprising the large servers and computers, and associated cooling systems consume massive power. Accessing the cloud service depends on the internet connection. The instability/unreliability and unavailability of connection hold back accessing the cloud. Involves security and privacy issues. It is true that cloud computing eradicates the big upfront investment on computing resources but accessing the right service is not that cheap either. Involves data transfer cost. Since cloud services are typically generic, they lack flexibility. The user/client has minimal control of

Computing approaches	Environmental advantages	Issues
		their own applications as the cloud service infrastructure is entirely owned, managed, and monitored by the service provider. Switching or migrating to a different cloud service provider is often complex or infeasible.
Serverless computing	It saves the energy requirement for running privately-owned servers. It also scales down quickly by shutting resources down when they are not in use. This saves a lot of energy consumption. Serverless codes not necessarily be run on any specific server. Hence, the serverless applications can be deployed at the edge of the network, i.e., close to the end-users [32]. This will not only reduce the latency but also saves a significant amount of energy by eliminating the need for unnecessary data transmission [30].	Not suitable for real-time applications which require low latencies. Also, not suitable for applications which need long execution times. Everything operates in stateless fashion; hence, handling state using stateless functions is a real issue. After some time of being idle, the function will require to go through a cold start which not only takes up to a few seconds but also consumes energy.
Using terminal servers	The processing and storage requirements for client machines are minimal because a terminal server hosts all the application logic which also runs on the server. The thin client uses up to $1/8^{th}$ amount of energy in comparison to workstations and thus considerably reduces energy consumption.	Running applications on a remote server always involve performance issues. There are chances of terminal servers getting bottlenecked with overloaded requests. Hence, the terminal server needs to be powerful enough to be able to handle all connections. If the terminal server is not backed up, there is a high risk of downtime due to a single point of failure. If the communication network is not reliable, the system will be affected harshly.
Virtualization	Effective resource utilisation leads to fewer production and less e-wastage. A single physical server acting as multiple server instances consumes considerably less energy in comparison to separate dedicated servers. It requires quantitively less hardware to run similar applications than dedicated systems, which leads to fewer device production and less e-wastage. The absence of the usage of the local hardware or software cuts the overall energy consumption.	The required hardware specification (e.g., memory, processor, etc.) is much higher for the same task executed in a native computer. Involves complex troubleshooting, in case of failure. Degraded performance than a physical server. Suffers from availability issue which discourages using virtual servers for mission-critical applications. Has major security issues.

1.6 MCC as Sustainable Computing

The computational measures and approaches for sustainable computing discussed in Section 1.4 and 1.5 are not sufficient for realising sustainable computing absolutely. In this section we introduce the concept of MCC and its sustainable benefits.

1.6.1 Mobile Crowd Computing

Object-oriented programming brought the revolution in software development by introducing the concepts of reusability and polymorphism, which allow software modules to be used multiple times for multiple purposes. This saves a significant amount of manhours and cost. We envisage the same role of MCC in sustainable computing.

In the previous section, we understood that grid computing has been successful in utilising the existing devices. But the problem with grid computing is that desktops are losing popularity; in fact, the same for laptops. On the other side, SMDs such as smartphones, phablets, and tablets are gaining huge acceptance as the new computer with the computing power they offer thanks to the power-packed hardware. The technological progress of SMDs, such as powerful SoCs with multicore CPUs and GPUs, has made them favourable as the primary computing device to many people. Industries are also showing interest in this direction. Initiatives such as Microsoft's Continuum[1] and Samsung's DeX[2] are striving to bring the desktop experience on the SMDs. Microsoft has endeavoured to run its full version of Windows 10 on the ARM chipsets, the most popular chipset for SMDs. And the great thing about SMDs is that they have become indispensable to our lifestyle. It is not feasible to restrain ourselves from using them. So, why don't we use these devices of their optimal potential, i.e., for computing purposes as well?

Furthermore, A number of such powerful SMDs, collectively, can offer huge computing capability. A satisfactory HPC may be achieved by making a grid of SMDs [34]. A typical MCC system can be perceived as a distributive computing

[1] https://www.microsoft.com/en-in/windows/continuum
[2] https://www.samsung.com/global/galaxy/apps/samsung-dex/

framework where a large job is divided and distributed to the people's SMDs to be executed. The philosophy of MCC is to combine computation power of numerous dispersed SMDs to escalate the overall computation power. The cumulative computing power achieved by such grids of SMDs can tail off the dependency on the data centres and low-end supercomputers as well.

Since in this proposed computing environment, the public-owned mobile devices are targeted to be utilised, this particular computing system is named as MCC. The users can share their SMD resources in a voluntary or incentive basis. The details discussion on MCC is presented in Chapter 3.

1.6.2 Sustainability of MCC

The concept of sustainable computing considers the total ownership cost, energy efficiency, environmental impact, and the benefit of the technology. Let us assess if MCC meets these requirements as a feasible sustainable computing option.

Economic sustainability: Setting up in-house computing infrastructure requires an upfront investment. It also involves regular expenditure for operational and maintenance costs. On the other hand, in a dynamic pricing model, using cloud services in peak hours will be considerably high-priced. Whereas setting up an MCC would require almost no cost. In an organisational MCC, the available SMDs on the premises can be effectively utilized for this. Organisations can cut costs significantly by adopting the bring your own device (BYOD) policy, obliging the employees to contribute their devices to MCC.

Energy efficiency: Running an SMD requires less power as compared to computers. Moreover, the CPUs of the contemporary SMDs are substantially more power-efficient, with merely 1 to 2 Watts of power consumed at their highest utilization with the peak load [35]. Therefore, they consume much less energy than other computing systems to perform the same operation. Statistically, the energy consumption of a standard SMD ranges from a few to 10 kWh per year. Therefore, total energy consumption is 10 TWh per year considering one billion SMDs are in operation worldwide. This is only 1% of the total energy consumed by ICT which is typically on the order of 1,000 TWh per year [6]. Furthermore, using MCC will not

incur any additional energy consumption as compared to dedicated computing infrastructure. Also, the availability of battery power eliminates the need for generator power sources. In effect, the need for electricity generation would be curtailed to a great extent. This leads to less consumption of fossil fuels, having a positive impact on the environment. Also, the heat dissipation of the SMDs is marginal compared to other computing devices; therefore, MCC would allow shunning the use of a cooling systems as required in traditional and dedicated computing resources, which thus would cut off the energy wastage in cooling. Considering the above-mentioned aspects, it can be reckoned that MCC would reduce the carbon footprint considerably.

Environment friendly: MCC supports the reusability approach, i.e., optimal and multipurpose use of existing devices without going for exclusive and specific purpose ones. This will curtail the production and use of new devices, minimising the environmental hazards of device production and e-waste significantly [36]. Furthermore, the small size of SMDs also aids in reducing the negative effects since they require less material in manufacturing and thus produce less manufacturing waste materials. Further, the contribution of e-waste of discarded phones also is considerably lesser in amount. A proper policy and implementation along with disciplined and responsible users can help in controlling and managing e-waste effectively. The environmental benefits of MCC are summarised in Fig. 1.5.

Utility: Besides the economic and environmental benefits, the MCC offers several other benefits, such as flexibility and scalability. An MCC can be set up anywhere in an ad-hoc manner. In an organisational MCC, considering the abundance of available SMDs, a practically zero-cost HPC can be achieved. Due to close proximity, MCC provides not only low and predictable latency but also configurable latency, making it suitable for time-constrained applications.

No additional manufacturing	•No need for explicit computing device production as people would anyway use SMDs.
Less manu-facturing hazards	•Production of smartphones is much environment-friendly compared to large computers.
Less e-waste	•Due to the small size, the e-waste will be lesser and can be managed more efficiently.
Do not require cooling	•No dedicated cooling systems are required which saves electricity significantly, hence reduces carbon footprint.
Do not require power backup	•No power backups, such as large batteries and generators are required. Pollution due to the battery elements and diesel fuel are avoided.
Energy-efficient computing	•SMD processors are typically energy-efficient. They consume much less energy than other computing systems to perform the same operation.

Fig. 1.5. Environmental advantages of MCC

1.6.3 Environment-friendliness of MCC in Comparison to Other HPC Systems

To establish our argument that MCC is a sustainable alternative to the HPC systems, in this section, we statistically compare MCC with other computing systems viz. desktop grid computing, supercomputers, and data centres, in terms of environmental impacts.

In recent years, the computing services offered through cloud computing have got tremendous popularity. People can rent computing resources on usage and requirement basis. The cloud service providers maintain big data centres to cater to the computing resource needs of the clients. A data centre, abstractly, can be described as an abundant number of computers stacked together. To make the cloud service available 24x7, these computers are always kept on which makes them very hot. As a result, a huge amount of power is consumed not only to run these computers but also to keep them cool. About 30 billion watts of electricity is needed to run the data centres (comparable to the electricity generated from 30 nuclear power plants) which cause nearly 17% of the total carbon footprint caused by technology [37]. Data centres consumed 416.2 TWh of electricity in the year 2015 only which is roughly 3% of the global electricity supply [38]. A single data centre can consume more power than an average town. To provide uninterrupted power supply in case of power failure, the data centres run generators that emit diesel exhaust. Today's data centres cause roughly 0.3% of overall carbon emissions [5], which is equivalent to the carbon footprint generated by the airline industry [38].

Of the total global greenhouse gas emissions, the power-hungry data centres account for nearly 2%. This is putting an immense impact on the environment, leading to global warming. The bad news is, every four years, this energy requirement is getting doubled and the total energy requirement of the data centres, globally, will increase threefold in the next decade. By 2025, data centres are expected to use 20% of the world's energy [39]. The efficiency of the data centre is measured in terms of PUE. PUE compares the non-computing energy to the amount of energy to power actual machines. Data centres operate at 70% of overhead energy. It means another 0.7 units are used behind the infrastructure of the data centres. So, the total PUE goes up to 1.7 [37]. Typically, the PUE of the common data centres is about 2.0 [5].

Due to their computing capacity and power, the energy requirements of supercomputers are gigantic and might well be equivalent to that of a small city. The correct response relates to electricity; specifically, 17.8 megawatts of power is required to run Tianhe-2, one of the Top500 ranking supercomputer boasting 33.9-petaflop through 3.12-million processors. An exaflop (1,000 petaflops) computer needs approximately 500 megawatts, which is equivalent of the total output of an average-size coal plant, and enough electricity to cater the needs of all the households in a city like San Francisco [40].

Though desktop grid computing involves lesser power consumption than the above-mentioned two systems, the average desktops and laptops still consume more power than SMDs. As mentioned earlier, the desktop grid is an affordable option for sustainable computing and can lower the environmental impacts of supercomputers and data centres considerably. However, MCC promises to minimize it further. Table 1.2 summarises the comparative environmental impacts of SMDs, desktops and laptops, data centres, and supercomputers.

Though MCC can reduce the amount of e-waste to a great extent the extensive adoption of SMDs becomes worrisome for likely e-waste generation. Therefore, it is extremely crucial to opt for the proper disposal of discarded devices and try to recycle as much as possible. The role of governments and industries towards this direction is already discussed in Section 1.3.2. Fig. 1.6 lists the responsibilities of the

users and the responsible authorities suggesting what to be done when the SMDs are discarded.

Table 1.2. Comparing environmental impacts of SMDs with data centres, supercomputers, and Grid computing (desktops and laptops)

Environmental impacts	Data centre	Supercomputer	Grid Computing		Smartphone
			Desktop	Laptop	
Energy consumption	200 TWh/year [5].	17.8 mW for Tianhe-2, the 33.9-petaflop supercomputer with 3.12-million processors [40].	100-150 Wh, 600 kWh/year [41].	60 Wh, 300-150 kWh/year [41].	1.5-3 Wh. An average phone needs 2 kWh/year.
CO_2 emission in the manufacturing process	171,630 kg CO_2 [42]. Around 0.3% of overall carbon emissions [5].	0.175 million kg/ year per (a supercomputer, equivalent capacity of 1000 PCs).	380 kg/ desktop [43].	227 to 270 kg/laptop [44].	16 kg/year [45]. An average mobile emits 35 kgs of carbon while manufacturing [46].
Other environmental hazards	Along with the common hazardous materials such as Fe, Cu, Al, Ag, Au, Pt, Pd, Pb, Hg, As, Cd, Se and hexavalent Cr and BFRs, other harmful elements such as ethylene/ propylene glycol for cooling systems, diesel fuel for backup generators, lead-acid batteries for UPSs, and compressed gases for fire suppression makes data centre real peril to the environment [47] [48].	Same as data centres.	The metals contained in PC's commonly include Al, Ag, As, Au, Ba, Be, BFR, Cd, Co, Cr, Cu, Fe, Ga, Hg, Mn, Pb, Pd, Pt, PVC, Sb, Se, and Zn. Most of them are really hazardous and contaminate soil, water, and air, if not properl y disposed off [49].	Almost all the hazardous elements of desktops are also found in a laptop, but the quantity is less as laptops are typically smaller than desktops.	The hazardous metals such as Al, Ag, Au, Cu, Fe, Pb, Hg, Cd, etc. are needed in smartphone manufacturing also, but in much less quantity than desktops and laptops.
E-waste generated	32360 metric tons of e-waste in 2018.	9.3 million tons/year [50]	41698.8 metric tons [51].	3230 metric tons [51].	In India, out of 650 million mobile users, 40% have changed their

Environmental impacts	Data centre	Supercomputer	Grid Computing		Smartphone
			Desktop	Laptop	
					phones in 2017, generating huge e-waste. [52] In the USA, yearly, nearly 150 million mobile phones are discarded.
Weight fraction of materials (%)	N.A.*	N.A.*	47.2 Fe, 0.9 Cu, 2.8 plastic, 9.4 PCB [53].	19.5 Fe, 2.4 Al, 1.0 Cu, 25.8 plastic, 13.7 PCB, 14.4 battery [53].	0.8 Fe, 0.3 Cu, 37.6 plastic, 30.3 PCB, 20.4 battery [53].
E-waste decomposition	Decomposing is challenging as a huge volume of e-waste generated due to a large scale of components. Require large dumping round; risk of toxic metals and chemicals; and contamination risk. But, since the data centres are generally owned by big companies/ institutes, they are expected to follow the systematic decomposing and recycling process.	Same as data centres.	As the number of users is very large scale, the proper and systematic decomposition of e-waste is really difficult. Most of the computers are public-owned or owned by small organizations. Most of them do not follow the proper decomposition and recycling processes.	The same problem, but moderate due to less equipment as compared to desktop.	The continuous growth in smartphone users with very brief use-cycles is a great challenge in terms of decomposing and recycling as the lack of awareness and eagerness among the public. But if the civic authorities take active roles and are able to convince people the necessity of proper disposal of discarded devices, the problem can be tackled.

* Despite of our best effort, we couldn't find reliable data.

Fig. 1.6. Procedures to be followed for SMD e-waste management [54]

1.7 Motivation of this Study

Our main motivation for exploring MCC is its sustainable benefits, as discussed in Section 1.6. We understood that using the devices that are already in use for computational and other purposes would reduce the requirement of buying IT infrastructure separately because the public would buy SMDs for their own purposes, anyway. Moreover, the utilization of these public-owned SMDs to achieve MCC instead of traditional HPC will reduce the requirement for dedicated large computers.

In this section, we discuss the rationales which we believe favours and complements the idea of MCC, making our motivation stronger. Thereafter, we shall establish the potential opportunities through which the benefits of MCC can be reaped.

Specifically, the inducement factors (benefits, opportunity, and potential) of MCC discussed in this section in addition to the discussions in Section 1.6 and Section 3.5 motivated us in fostering the proposed system.

1.7.1 Powerful SMD Hardware

Over the last few years, the SMD industry has seen an unprecedented focus on the hardware. Be it CPUs or GPUs or even DSPs, the processing capability of SMDs, to meet various purposes, has been increased exceptionally. The CPU and memory architectures are designed and tuned to boost heterogeneous computing. The

development of ARM processors, the most popular processor architecture used in SMDs and supported by most of the major SMD operating systems, has made them a serious contender in consideration for a range of scientific applications due to their high competence in floating point performance. The GPUs are also engineered to enhance GPGPU computing performance. The modern SMD GPUs are capable of delivering more than 800 GFLOPS. Advancement of CPU, GPU and DSPs has led to massively powerful SoCs. SoC like Tegra X1 from NVIDIA can deliver 1 TFLOPS while the computing cores with clock frequencies 2.5 to 3 GHz have become common [55]. Advancement on each module, though separately, makes the SMD, as a whole unit, a great possibility to become a powerful computing platform.

In the foreseeable future, more powerful processors with more cores are anticipated. With dense fabrication technologies like 7 nm and less, more muscle can be put up in a single core which will enable shoving more computing capacity without compromising the chip size. The future SMDs loaded with these powerful SoCs will be, in the true sense, the little computing giants.

1.7.2 Mass Adoption of SMDs

The SMD market has witnessed astonishing growth in the recent years. According to the recent research market statistics, globally smartphone shipments had reached 1.55 billion [56]. As per a study made by IDC, a USA-based major data analytics company, a 5.7% growth has been noted in the YoY change in global smartphone shipment in the year 2021 with 1,354.8 million of smartphone units have been shipped globally compared to 1,281.2 million in 2020; and it is expected to reach 1.52 billion units in 2025 [57]. An estimation by Statista, a leading market and consumer data provider, suggests that the number of global smartphone subscriptions will reach about 7.7 billion by 2027 [58] [59], while in India, the smartphone userbase is predicted to reach 0.97 billion by 2025 [60]. In May 2019, at the annual developers conference, Google proclaimed that it had more than 2.5 billion active Android devices [61].

As per a report from BankMyCell[3], presently 48.37% of the world's population own a smartphone. Adding the number of tablet users with these statistics, therefore, there is a great probability of finding a sufficient number of SMDs at a populous place. Thus, it can be confirmed that the huge adoption of SMDs across the world has put a big platform for MCC.

1.7.3 Abundant Idle Resources

It has been observed that the majority of SMDs are not being used to their capacity. Studies suggest that normal users interact with their SMDs only for a few hours (on average two to four) in a day [62] [63]. So, a huge amount of processing capability remains unused and wasted. Even when SMDs are in use, it is highly probable that some of the CPU cores and the GPUs, alongside DSPs, ISPs, etc., remain free. An enormous processing capability can be generated if these unused processing powers are tapped and exploited properly (opportunistically).

1.7.4 Popularity of Crowdsourcing

Recently, the crowdsourced systems are gaining increasing popularity involving various applications [64] [65] [66] [67] [68]. In these systems, depending on the application's demand, different resources and information of the users and their devices are shared [69] [70]. We perceive MCC also as a crowdsourced system where the users share their SMDs' computing resources. Witnessing public's non-inhibition in sharing, we strongly believe that MCC would also be well acknowledged by the users as in case of other crowdsourced systems.

1.7.5 Implementational Opportunities for MCC

Considering the benefits and potential of MCC we envisage its wide-range utilization. Not only as HPC or organisation computing infrastructure, it can be implemented for ad-hoc mobile cloud computing as well as edge computing, as discussed below.

Organisational HPC: Due to the holistic adaptation of IT-enabled services, the organisational computing requirements have been increased staggeringly. In an

[3] https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

organisational computing setup, besides the financial burden, the traditional in-house computing facility has several factors which put more load on the environment and human health. Some of these factors are listed in Fig. 1.7. These are considered significant hindrance for sustainable computing. On the other hand, along with economic and utility benefits, in Section 1.6, we have realised the environmental advantages offered by MCC. The MCC can reduce the two-thirds share of ICT energy consumption on the count of the use of computers and data centres.

Ad-hoc mobile cloud: In traditional mobile cloud computing, the computationally intensive tasks are offloaded from the mobile devices to the cloud for execution [71]. However, due to latency or unavailability of internet connection, accessing cloud services are not always desirable. This can be mitigated through MCC. A virtual cloud can be set up in ad-hoc manner by using the collective computing power of a group of spatially adjacent SMDs [72]. Due to widescale SMD user base, there is a great probability of finding a sufficient number of SMDs not only at a populous place but also at scantily crowded locations. This infrastructural flexibility of MCC and the omnipresence of SMDs, would make it possible to form an ad-hoc cloud anywhere, allowing to achieve on-demand pervasive and ubiquitous computing [73]. In this ad-hoc mobile cloud, a mobile device can offload the computing-intensive jobs to the cloud which would be executed by the mobile devices in the cloud [40].

Edge computing: The wide adoption of IoT and sensor-based applications has led to the continuous generation of a huge amount of data. For actionable analysis, these data need to be processed in real-time. Cloud computing has been a popular option for this. However, the significant latency makes cloud services unsuitable for real-time applications [30]. For this, edge computing solutions are being proposed where data are being processed at the edge of the network [74]. But setting up these local data processing architectures will incur a considerable additional cost, besides the usual environmental liability. Also, as these solutions are offered by specific vendors, they naturally tend to involve high costs. Another problem with vendor-offered solutions is that there is a minimum scope of interoperability with legacy systems and solutions from other vendors as they tend to be closed and

tightly coupled. This increases the IT infrastructure cost further. In view of these issues of the proprietary edge systems, we believe that MCC can be considered as a feasible edge computing system [75]. Especially, the SMDs available in the vicinity can be leveraged to process the in-campus IoT data [76].

1.7.6 Aim and Scope of the Work

The primary aim of this study is to establish MCC as a flexible computing system that can provide centralised HPC as well as ad-hoc computing. Though the MCC approach sounds simplistic, successful implementation is not straightforward. It involves several research issues such as designing a suitable architectural framework, finding best suitable SMDs, fair and optimised scheduling, mitigating user mobility, minimizing unnecessary job offloading, ensuring fault tolerance, motivating people for volunteer participation, devising appropriate incentive models, among others.

Besides, theoretically establishing MCC as a sustainable computing solution, in this work, we limit our focus of study to only a few selective aspects. The performance of MCC largely depends on appropriately scheduling the tasks to the most suitable SMDs. For this an efficient scheduling algorithm is needed. Additionally, the primary criterion for correct scheduling is to find and determine the most suitable SMDs as computing resources for a given task. Similarly, assessing the availability of SMDs is important to maintain the QoS. Only proposing a concept metaphysically is not enough to establish it. In view of that, we envision presenting a proof-of-concept prototype of MCC as an organisational edge computing infrastructure.

Though MCC can be set up ubiquitously on an ad-hoc basis, we limited the scope of our proposed system primarily within the periphery of a campus or within a building such as an office and residential buildings, educational institutes, hospitals and clinics, shopping malls and retail marts, etc.

Coolant	Heavy cooling is required to mitigate the heat generated from the computing systems.
	This consumes massive electricity.
	The coolant used in the air conditioners causes global warming and ozone layer depletion.
Batteries	Huge batteries are required for continuous power backup.
	Commonly used lead-acid batteries have adverse effects on human health and the environment.
	If not properly disposed of they may contaminate the soil and water.
Cleaning materials	Dusting and cleaning are important in organisational computing systems for efficient operation.
	Varieties of cleaning solution are available, and most of them are toxic as they contain bleach, ammonia, or chlorine.
	These toxic cleaning solutions have an adverse effect on human health.
Diesel fuel	Diesel fuel-based power generators are often used in case of power failure.
	These generators are used especially at the sites which experience recurring power failure, and batteries can not support for longer durations.
	Diesel fuel produces an enormous amount of CO_2 and other chemicals which causes global warming as well as affect human health.
Electronic waste	Electronic equipment have a finite lifespan.
	Most of the computer peripherals need to be replaced by 3-5 years which is increasing the amount of e-waste enormously.
	The e-wastes are not easily degradabel and are harmful to the environment if dumped on the open landfills.
Fire suppression	With electronic equipment there is always chances of fire due to short circuits, etc. Therfore fire suppression sytems are commonly employed.
	Various chemicals used in the fire system may be harmful to the environment such as ozone layer depletion and global warming.
	These chemicals are toxic and may find its way to underground water or to rivers, thus, contaminating the water resources.
Packaging	The packaging materials of the computing equipment purchased by the organisations add huge waste every year.
	Some materials like foams, thermocols, plastic bag, and plastic support accessories are nonbiodegradable and need proper recycling.
	Dumping these on the open area may harm the environment.
Office premises	Running cooling and heating equipments and lights in the office for the entire day/night causes considerable electricity consumption and wastage.
	Daily office chores produce lots of paper, plastics, and packaging wastes.
	Floor cleaning, glass pane, computers, and carpets also produce chemical wastes.

Fig. 1.7. Factors that affect the environment indirectly in an organisational computing setup [77]

1.8 Research Objectives

In line with the above-mentioned aim and scope of this study, we designed our

research objectives as listed below:

- **Understand the gravity of sustainable computing**

  To convince ourselves of the requirement of sustainable computing, we genuinely need to realize the serious concern of the deteriorating environment of the Earth. Moreover, we need to recognize the role of the traditional and existing computing system in this. Based on this understanding, we would be able to identify the more threatening and addressable aspects.
- **Establish MCC as a feasible sustainable computing option**

  As we claim that MCC can be perceived and implemented as a sustainable computing solution, we need to establish the rationality behind this. Specifically, we aspire to:
  - Ascertain the sustainable benefits of MCC
  - Frame the layout of a generalised architecture of MCC
  - Identify the major challenges of MCC and suggest the probable way outs and research progress
- **Develop a systematic approach to profile the static and dynamic resource information of the SMDs for MCC**

  In MCC, the essential prerequisite is to profile and assess the resource parameters and their present status precisely. However, considering the heterogeneity and dynamicity of these resource parameters, profiling them and assessing their fitment for different requirements is not trivial. We aim to develop a model and systematic methodology to profile the static and dynamic resource parameters of the SMDs in real-time.
- **Select the suitable SMDs as per their static and dynamic resources**

  To achieve satisfactory performance and QoS, selecting the best resources (SMDs) is crucial. Scheduling the MCC tasks to the most suitable SMD as pr the task requirements would impact the efficiency of MCC significantly. Suitability of an SMD can be determined based on multiple criteria (e.g., CPU and GPU power including clock frequency and the number of cores, battery remaining, probable availability period, past history of reliability, cost of service, pre-load (whether overloaded or not), mobility behaviour, etc.). Considering

the heterogeneity of the assessment parameters selecting the most suitable SMD poses an interesting research problem.

- **Design resource-aware and energy-aware scheduler for MCC**
  Scheduling is an important aspect for MCC like any other distributed systems. The overall performance and the integrity of the MCC can be assessed by factors such as execution time, resource utilisation, load balancing, etc. An efficient task scheduler should conform to these requirements. Conversely, an inefficient scheduling method will have a negative impact on the QoS of MCC. Furthermore, considering the battery-powered constrained energy of the MCC resources, i.e., the SMDs, it is crucial to minimise the energy consumption to complete the scheduled task. This can be achieved to some extent by optimising the task scheduling to the appropriate SMDs. However, considering only energy efficiency might lead to a huge load imbalance among SMDs, i.e., the most energy-efficient SMDs would be overloaded most of the time. In a dynamic and heterogeneous system like MCC, it is nontrivial to realise an optimised scheduler, in view of the fact that scheduling in a heterogeneous distributed system is an NP-complete problem.
- **Determine the availability of an SMD to improve QoS of MCC**
  To maintain the QoS of MCC, determining the SMD availability is important. Ideally, an MCC task should be assigned to the SMD that has the maximum probability of being connected to the MCC network until the completion of the assigned task. Leaving an SMD without sending the result back would increase the overload for task offloading. Frequent task offloading would hamper the MCC performance significantly. Hence, it is required to assess the probability and the confidence of a particular or a set of SMDs being staying for a certain period of time (till completion of the job that is to be assigned).
- **Explore the suitable approaches for mobility-aware service provisioning in MCC**
  Mobility is a crucial issue in mobile computing, especially for a local MCC where the SMDs are connected through WLAN or other short-range communications. For a successful implementation of MCC, this has to be addressed. There is a requirement for finding the best possible ways to ensure the MCC

service provisioning mitigating the user mobility.

- **Present a proof-of-concept of MCC for practical application and implementation**

  To assess and convince the feasibility of MCC, it is important to present a working model of it. In this work, we would aim to present a proof-of-concept of MCC with the minimalistic features, which can be considered as a prototype for further development.

1.9 Thesis Structure

The thesis is structured as following. Fig. 1.8 illustrates a hamburger organisation of the thesis.

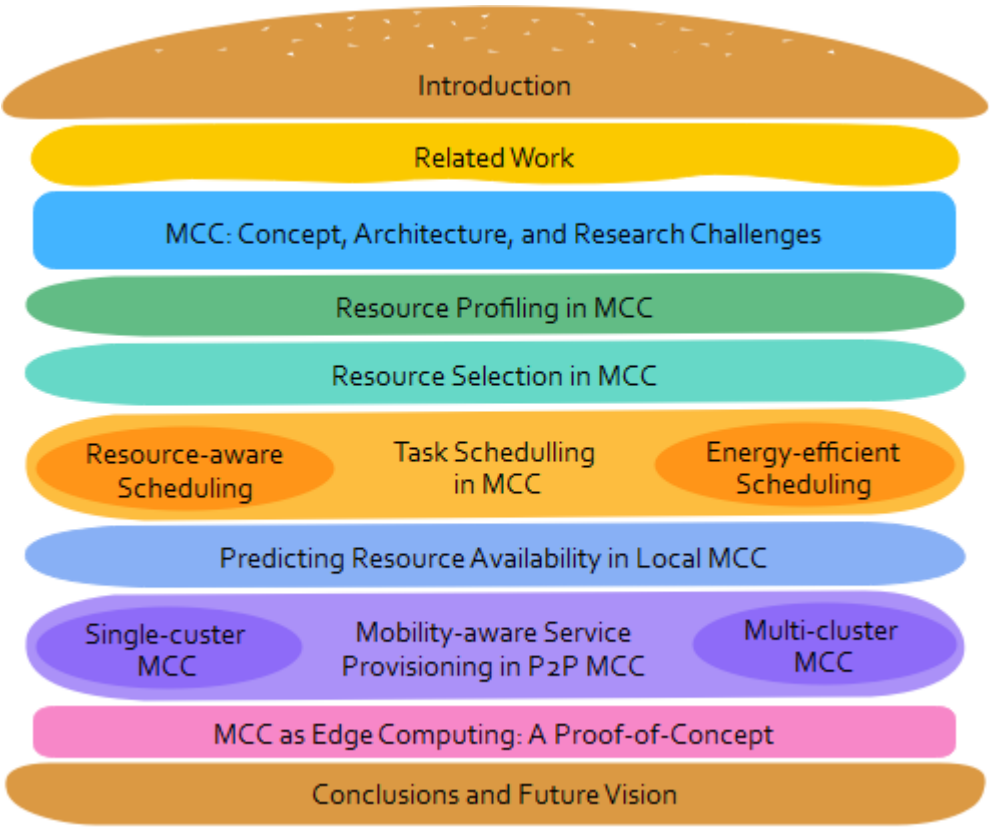


Fig. 1.8. Hamburger model of the thesis organisation

Chapter 1: We try to establish the need and importance of sustainable computing and how MCC would help achieving it. The primary aim of this chapter is to investigate and unveil the environmental impacts of the existing computing systems and set the background of presenting the need for an alternate sustainable solution in order to minimise the environmental hazards. Here, we aim to establish MCC as

the sensible and feasible solution to sustainable computing while highlighting the environmental benefits and the enablers of MCC in achieving the goal of sustainable computing.

Chapter 2: We look into the similar work found in literature. We report the research works and projects that are close to the overall concept of MCC. We also explore the related work of the individual research presented in chapters 3 to 8.

Chapter 3: We present the technicalities and the working of MCC in details, which include the probable models and architectures suitable for different implementations of MCC. Besides the sustainable benefits discussed in Chapter 1, other advantages of MCC are also mentioned. Furthermore, the issues and challenges associated with MCC implementation along with the open research problems are meticulously analysed.

Chapter 4: We present a methodological approach to profile the candidate SMDs to assess their resources for job scheduling. The intricacies of the designing, development, and implementation of the SMD profiling and selection, the two necessary components in realizing an organisational MCC, are presented in detail.

Chapter 5: We aim to find out a suitable MCDM method for resource selection in a dynamic and time-constraint environment like MCC. For this, we present a comparative analysis of various MCDM methods under asymmetric conditions with varying selection criteria and alternative sets.

Chapter 6: We present two scheduling algorithms for MCC. In the first part, we use a heuristic approach to propose a resource-aware multicriteria-based scheduling algorithm for MCC. In the second part, we use a PSO-based metaheuristic approach to propose a load balance aware energy-efficient scheduling algorithm for MCC.

Chapter 7: We provide an effective model to predict the availability of the users (i.e., their SMDs) in a local MCC environment to prevent frequent job offloading and job loss. We propose an advanced convolutional feature extraction mechanism that is applied to LSTM and GRU-based time-series prediction models for predicting SMD availability.

Chapter 8: We address the mobility issues of the users in a P2P MCC scenario. In the first part of this chapter, we aimed to predict a stable group of SMDs so that resource servicing takes place within the group only. And in the second part, we propose a service provisioning scheme considering the mobility of the resource providers and the consumers.

Chapter 9: We aspire to establish a proof-of-concept for the feasibility and use of MCC as a sustainable edge computing solution (MCC-edge). A typical smart HVAC system of an office building has been considered for the experiment case. We aim to process the HVAC data in real-time using the MCC-edge set up within the building for auto adjustment of the AC controller and error notifications.

Chapter 10: We summarize the thesis while discerning the probable future directions of MCC and related aspects.

2

Related Work

"Research is to see what everybody else has seen, and to think what nobody else has thought." --- Albert Szent-Gyorgyi

2.1 Introduction

The concept of MCC is certainly not absolutely new. It is based on several established computing paradigms and approaches, as briefly mentioned below.

Distributed computing: In a distributed system, the computing nodes are physically separated and connected through local or global networks [78]. The computing units work collaboratively to achieve a common goal.

Parallel computing: Parallel computing refers to the concurrent execution of multiple subtasks, descendent from a large task, on multiple processing units [79]. Generally, distributed computing adheres parallel computing though it is not mandatory for every case.

Cluster computing: To attain HPC, researchers suggested forming a parallel and distributed computing system by connecting multiple standalone computers through high-speed LANs [80]. By this, a single, integrated HPC can be garnered by aggregating the computing capacities of the connected computers [81].

Grid computing: Diverse types of computing resources are connected through a local or global network forming a grid of resources. In a typical computing grid, any grid member can consume resources from or provide its resource to other grid members.

Volunteer computing: Computing resources are voluntarily shared by owners from across the globe [82] [83]. Though grid and volunteer computing seem similar, there are a few differences, as listed in Table 2.1. However, many extended grid computing systems support volunteer computing [84].

P2P computing: It is a decentralised distributed computing system where computing loads are shared among peer computers connected through local or global

networks [85]. Each participating computer, i.e., a peer node, can work both as a client and a server, depending on the context of the computation [86].

Opportunistic computing: When two or more devices come into contact, it can be perceived as an opportunity to share resources, services, and information with each other. Opportunistic computing, essentially a kind of distributed computing, exploits the advantages of neighbourhood and proximity among computing devices. The continuous advancement of communication technologies, especially wireless communication technologies such as cellular, Wi-Fi, WiMAX, Wi-Fi hotspot, Bluetooth, etc., has broadened the avenue for opportunistic computing on a large scale. Opportunistic computing does not mean only locating the devices and accessing their resources opportunistically but also the opportunistic way to reach those devices [87]. That is termed an opportunistic network, which suggests finding the best path whenever possible to reach the device involved in opportunistic computing, probably in a pervasive manner [88].

CPU scavenging: Resource scavenging refers to tapping the potentially accessible computing resources whenever they are available and possible [89]. In volunteer computing, though resources can be availed from holy souls (contributing entities), they cannot be taken for granted, i.e., it is to be ensured that the normal functioning of the volunteering node should not be hampered. So, the external computing task should be executed when the CPU is free or very lightly loaded. This is known as CPU scavenging and can be perceived as another form of opportunistic computing [90] [91]. Grid computing follows this approach [92].

Crowdsourced systems: A crowdsourced system is a distributed system whose constituent components or the whole system are materialised by scrounging the hardware or software resources owned or generated by the crowd [34].

Crowd computing: Crowd computing is a form of volunteered crowdsourced system where computing resources belonging to the general public are utilised to meet HPC requirements. Some real-life social and scientific problems (e.g., drug discovery, gene mapping, cancer and AIDS research, mathematical modelling and simulation etc.) are so complex and computation intensive that they cannot be solved by means of conventional processing speeds and capabilities within a

feasible time boundary without employing heavily expensive supercomputers which most of the research institutes and organisations cannot afford. Crowd computing is a practical alternative to purchasing big compute clusters for carrying out computations required to solve these problems [93]. Many individuals and organizations come forward to allow these problems to be processed on their computers when they are not in use. The collective processing power can be compared to powerful supercomputers, minus the huge cost [55]. Having an internet connection and willing to participate in crowd computing, a user should download and install a client application through which he interacts with the project application running on a server. A middleware helps the client application to communicate to the server where the crowd computing project is hoisted. The highly jobs are distributed to the participating computers. The client applications on those computers opportunistically hunt for the unused computing cycles. On availability, assigned computing jobs are executed, and after completion, the results are sent back to the project's server. The client application ensures that lending the computing resources does not affect the user's own jobs.

Table 2.1. Differences between grid computing and volunteer computing **[94]**

	Grid computing	**Volunteer computing**
Resource owner	Grid resources are generally owned by organisations such as universities, research labs, public organisations, etc.	Resources are usually owned by the general public.
Resource availing approach	The jobs are pushed to the grid resources.	The volunteers pull the computing jobs.
Resource sharing	Both ways. A grid resource provider may also be a resource consumer.	One way. Volunteers are always the resource providers.
Resource class	Organisational	Personal
Reliability	High	Low
Security threat	Low	High
Public outreach	Low	High

A typical MCC system either follows all of the above-discussed computing paradigms or most of them. Nearly all of these technologies are decade old and a colossal number of research work can be found on these, addressing various areas. In this chapter, we focus on only those works which very closely related to the problems addressed in this thesis. In the following, we examine the related literature on that basis, as each subsequent chapter is organised as individual section. For each section, exploring the existing research, we try to find out the research scope.

2.2 MCC as Computing Paradigm

As mentioned above that several existing computing paradigms have incited the notion of MCC. In this section, we try to find out the works that incorporate these computing approaches with mobile computing. We also mention a few prominent global projects that fostered the idea of MCC in its basic form.

2.2.1 Related Research

In the following, we categorically report the pioneer and recent research works that specifically relate to computing on mobile devices. The categories are in no way exclusive because all the concepts discussed below are either overlapping or represented similarly in the literature. However, we considered only those papers which deliberated mobile devices as computing resource providers in some way.

Distributed computing: Networked mobile devices are extensively explored for performing distributed computing [95] [96]. In his Master's thesis [75], Marinelli used Android-based smartphones as computing resources by porting Hadoop Apache to the smartphones. He developed a mobile cloud computing system named Hyrax that offered a networked computing environment for smartphones. However, the performance of Hyrax was not up to the mark mainly because of the inferior smartphones of that period, and also, Hadoop was not optimized for mobile devices. Authors of [97] and [98] proposed a MapReduce-based distributed computing framework for mobile devices, named Misco, that supports the development of distributed data clustering applications on networks of smartphones. Lee *et al.* [99] discussed the advantages of using mobile devices for distributed analytics. They carried out distributed analytics with a CPU, memory and/or I/O intensive workload on a Hadoop-based cluster of Android mobile devices and assessed the performance using typical Hadoop benchmarks. The latest Hadoop software framework was entirely ported to a high-end device (e.g., Google NEXUS 7 tablet) without degradation of the overall performance. Arnold [100] presented a framework for distributed computing over smartphones. In [93], the authors did a feasibility experiment by implementing a small cluster of Android smartphones to show that a local and mobile ad-hoc cluster can be built using powerful smartphones. Datla *et al.* [96] suggested local cooperation of mobile devices to

execute computing-intensive tasks. Dong *et al.* [101] proposed a generic randomized task assignment framework for participatory computing named REPC, in which the mobile devices were used as computing devices to process the computing-intensive tasks. They used pedestrians' mobile devices to identify wanted criminals. They implemented it on a testbed of twenty Android smartphones connected through a wireless network in an open area. Dumont [102] designed and constructed a REST web service based distributed mobile computing system. They proposed a communication protocol, MEXP, for data exchange between devices. They tested the system in a biology laboratory scenario. Salem [103] proposed a 4-tier architecture for a web-based distributed computing system comprising a network of smartphones to execute machine learning models and provide predictions. Sanches *et al.* [104] presented a data-centric distributed computing framework. The proposed infrastructure-less distributed system consists of co-located mobile devices and can process batches or streams of data generated by the devices. Here, instead of sending the tasks to specific resource-providing devices, to minimise the data exchange, the authors proposed to process them at the data generating node only.

Cluster computing: Büsching *et al.* [93] presented a proof-of-concept demonstrating the feasibility of building a mobile cluster. They built a cluster with six Android mobile phones connected through a USB hub with a controlling node and evaluated the cluster's performance by running LINPACK, a standard benchmark for HPC systems. The cluster attained a 75% performance level of the ideal linear scaling of six times. The 25% deficit was mainly due to network latency and inability to exploit the devices' resources 100% (following the non-intrusive and opportunistic execution policy). In a similar work, Attia *et al.* [105] built a mobile cluster of two Android devices with six cores. They used standard C programming with MPI for task assignment and communication between two mobiles. As task size grew, the authors observed significant performance improvement using the cluster compared to individual processing.

Opportunistic computing: Conti and Kumar [88] observed the prospects in opportunistic computing along with opportunistic networks in the context of social

and pervasive computing. An encouraging discussion on opportunistic computing can be found in [106]. Murray *et al*. [107] suggested a crowd computing system formed by utilising an opportunistic network of cooperative mobile devices inspired by human social interaction to achieve large-scale distributed computation. Shi *et al*. [108] presented opportunistic mobile computing, where an in-motion device that needs external resources to run a computational task avails the same from a resource-providing mobile device that it encounters on its trajectory and gets connected intermittently. They designed a job distribution and task allocation scheme for this uncertain and unreliable computing environment. Mtibaa *et al*. [109] presented a proof-of-concept of dynamic and adaptive opportunistic mobile computing. They proposed a generic peer-to-peer computation offloading architecture to offload the task to various resource-providing entities such as mobile device clouds, cloudlets, and clouds. For this work, the authors considered both computation opportunities and network opportunities while counting minimized response time, reduced energy consumption, and increased network lifetime as offloading criteria.

Volunteer computing: Tapparello *et al*. [110] presented the state-of-the-art of volunteer computing on mobile devices and a literature review of how the different parallel and distributed computing architectures have been adapted to use mobile devices for opportunistic computing. Erick Lavoie *et al*. [111] presented a first-of-its-kind web browser based distributed computing tool, Pando, for volunteer computing on personal mobile devices. The devices can be connected via either LAN or VPN, or WAN. The researchers used a declarative concurrent programming model and implemented it using JavaScript, WebRTC, and WebSockets. Due to its independence of specific communication protocols or input-output libraries, the authors claimed Pando is also re-implementable in other programming environments. To support the HPC requirements in ALICE, one of the four main experiments of CERN's LHC project, Jenviriyakul *et al*. [112] developed a prototype of a volunteer computing platform on mobile devices based on BOINC [113]. The prototype named ALICE Connex exploits and aggregates the unused computing power

of the volunteer smartphones to calibrate the ALICE's time-of-flight (TOF) particle detector.

Enterprise computing: Acknowledging the opportunities and potential, mobile devices are being proposed to be utilised as organisation-level computing resources. Arslan *et al*. [114] [35] proposed an enterprise-level distributed computing framework using SMDs where organisations could leverage the idle SMD resources of their employees when the devices are left for overnight charging. They implemented a prototype to demonstrate the viability and efficacy of the system called 'computing while charging'. For executing enterprise applications, Schildt *et al*. [115] aimed to utilise the idle CPU cycles of the smartphones given to employees by the organizations. The goal is to offload a major portion of the enterprise computation to employees' mobile phones while charging during working hours. They designed a Java-based software framework that can distribute computing tasks to a network of Android devices and personal computers and gather the results. Using Java gave the researchers twofold advantages: i) they could reuse the existing business logic written in Java, and ii) the framework not only supported Android mobile devices but also could run regular desktop or server hardware, allowing seamless job distribution.

Mobile grid computing: In early 2000, with the prolificity of mobile phones, researchers endeavoured to incorporate mobile devices to grid computing in different ways [116] [117] [118] [119] [120]. Most aimed at extending the desktop grid to mobile devices, i.e., the accessibility of grid resources from mobile devices. But gradually, as the mobile devices became more powerful, they were treated as computing units altogether. As a result, they became part of grid computing not only as resource consumers but also as resource providers [121]. Kurkovsky and Bhagyavati [122] [123] proposed a wireless mobile grid to carry out resource-intensive tasks. Any device in the grid can initiate a task that it cannot execute on its own due to resource limitations. The task is distributed to one or more devices in the grid, and after execution, results are collected. Katsaros and Polyzos [124] proposed a campus-wide hierarchical mobile grid architecture where the resource-providing mobile nodes are connected through WLANs. Black and Edgar [125] demonstrated

the feasibility of using mobile devices as grid resources by implementing the BOINC client on an Apple iPhone via an emulated x86 virtual machine. The computing tasks were downloaded from a BOINC server, executed on the virtual machine (mimicking iPhone), and results were sent to the server. Viswanathan *et al*. [126] presented a resource-provisioning framework for a hybrid grid comprising both static and mobile computing grids. The proposed system boasts autonomic capabilities, such as self-organization, self-optimization, and self-healing, while considering energy- and uncertainty-aware resource allocation. Sriraman [127] revisited the possibilities of using smartphones and tablets in grid computing while briefly mentioning some challenges and solution approaches.

Ad-hoc mobile cloud computing: The concept of the mobile grid was extended as an ad-hoc mobile cloud where a minuscule local cloud is formed in an ad-hoc manner by connecting several mobile devices available nearby [128]. Canepa and Lee [129] presented a framework for creating an ad-hoc mobile cloud with stable devices in the vicinity of the resource consumer. They implemented the prototype in Java and using Hadoop and tested its performance a Korean character recognition on an OCR and then translating it to Romanize. The authors observed a negligible (1% on average) performance degradation in carrying out the task in the mobile cloud than executing it on a single device. Khalifa *et al*. addressed various aspects of ad-hoc mobile cloud in their series of works [130] [131] [132] [133]. Miluzzo [72] presented a theoretical perspective on the feasibility and potential of local cloud computing made up of a collection of cooperating mobile devices available nearby for running resource-intensive applications. In [134], while deploying ad-hoc mobile cloud in a small hospital scenario besides mobile devices, they also included semi-stationary on-board computing resources of vehicles. Nishio *et al*. [135] proposed architecture and mathematical framework for heterogeneous resource sharing in a mobile cloud. Based on the idea of service-oriented utility functions, they aimed to develop a unifying utility function which could map all the heterogeneous resources into a single parameter. Funai *et al*. [136] proposed and implemented a mobile computing system connected through an ad-hoc network. Here, a volunteer mobile device elects itself as a local task distributer and invites

others to join the ad-hoc cloud for computation via D2D communication such as Wi-Fi Direct. Remédios [137] presented an early prototype of an infrastructure-less local mobile cloud for processing locally generated big data. To minimize execution time and energy consumption, Yaqoob *et al*. [138] proposed a heterogeneity-aware task allocation scheme for ad-hoc mobile cloud computing. Shila *et al*. [40] proposed an automatic, scalable, and efficient service/resource management framework for an ad-hoc cloud. The proposed framework is a generic one, i.e., it considers both static and mobile resources in the vicinity. Balasubramanian and Karmouch [139] presented a framework of an ad-hoc mobile cloud for P2P IaaS provisioning. In their proposed system, a needy mobile device can search and select peers and form an ad-hoc network with them. After forming the virtual cloud, the jobs are obtained from the resource-seeking device, executed, and the results are sent back. On completion of the tasks, the resources are released. Lately, ad-hoc mobile clouds are proposed to attain edge computing aiming to meet the need for time-constraint applications [140] [76].

Mobile crowd computing: Loke *et al*. [141] argued the feasibility of MCC. They presented a job distribution approach among the mobile devices in such a network connected via Bluetooth. In this framework, a mobile device designated as a delegator distributes the jobs to other mobile devices designated as workers, which execute the assigned jobs in a work-stealing fashion. Fernando *et al*. [142] [143] introduced a crowd computing framework for mobile devices, named Honeybee, through which mobile devices can share work and utilize local resources, employing the work-stealing strategy and load balancing among different devices. They proposed a work-stealing method for MCC, which utilises locally available smartphones, combinedly and collaboratively, to form a local mobile resource cloud. In his book "Crowd-powered mobile computing and smart things" [144], Loke discussed various potential aspects of MCC in applications like ubiquitous computing, context-aware servicing, drone services, and smart things. Prem Kumar *et al*. [145] developed a client/server-based distributed computing platform by harnessing the computing power of public mobile devices. To carry out some tasks through this system, a user must upload a dataset, the executable Java code, and another that would combine the results. The authors offered a task distribution

and scheduling algorithm that abstracts the computational heterogeneity of the devices, task execution complexities, and the uploaded dataset size. The proposed system also offers storage services utilising the space available on public devices. For security, they used threshold cryptography on the uploaded files to create encrypted shares. Going beyond the mobile phones, Kündig *et al.* [146] utilised other portable smart personal devices, in addition to smartphones, to realise crowdsourced edge computing. An ad-hoc edge network is formed by connecting the crowd peers within a nearby neighbourhood. The peers connected in a dynamic mesh architecture can be task handlers, workers or message brokers. They demonstrated the applicability of such crowdsourced edge by implementing a video-enhanced object search in a campus-based wide local area to generate informative heat maps or identify a specific object using deep learning techniques.

2.2.2 Global Projects

SETI@home, the first truly crowd computing project, was initiated by the University of California, Berkeley and made public in 1999 to search for evidence of extra-terrestrial life. For this, they designed BOINC[4] (Berkeley Open Infrastructure for Network Computing), an open-source middleware that provides support for volunteer grid computing projects. Since then, many volunteer and crowd computing projects have been instigated. Some of them have successfully been completed. Some are still going on. Later, in late 2004, IBM joined hands in the mission to solve various problems, including medical, environmental and other humanitarian posers, by launching a grid computing framework named World Community Grid[5] (WCG). BOINC and WCG jointly or individually initiated several large-scale projects focusing on different scientific problems. Some notable projects among them are Einstein@home[6] (pulsars and gravitational waves detection), FightAIDS@Home[7] (HIV/AIDS research), Folding@home[8] (disease research),

[4] https://boinc.berkeley.edu
[5] https://www.worldcommunitygrid.org/
[6] https://einsteinathome.org/
[7] https://www.worldcommunitygrid.org/research/fahb/overview.s
[8] https://foldingathome.org

LHC@home[9] (high energy physics), etc. GPUgrid.net[10] (focuses on problems related to biomedical research), Citizen Science Grid[11], and distributed.net[12] are some other popular examples of crowd/volunteer computing platforms and applications. All the projects have a common motivation for reducing costs compared to traditional HPC systems.

Another online computing community service named grid.org[13] was launched in 2001 which ran several different volunteer computing projects. It started with a cancer research project aiming to screen the target molecules against known cancer target proteins. Successively, varieties of other projects such as finding cures for small pox and anthrax, analysing human protein folding, hidden Markov modelling and web load testing were introduced.

Though almost all these projects are desktop-based, it won't be off beam to be optimistic, inspired by the attainment of desktop crowd computing, about the success of MCC as well. That is why many crowd computing projects are also fitting themselves to run on smartphones. BOINC published an Android version, which allows Android devices to join and contribute voluntarily to crowd computing projects [147] [148]. Users can choose one or more projects among several Android-supported projects (e.g., Asteroids@home, Rosetta@home and many others) available. If a user has an Android-based smartphone wish to volunteer his phone's processing cycles, he downloads BOINC client software which receives jobs from a designated server [149]. Whenever the client application senses an opportunity, it utilizes a user's phone's processor to execute assigned tasks. On completion of the particular task, the result is sent to the server. But since battery life is a severe concern to users, the middleware checks for the battery status of the smartphone before initiating processes. By default, the application runs if the battery level is over 90 percent or if the phone is being charged. Moreover, BOINC works only

[9] https://lhcathome.web.cern.ch/
[10] http://gpugrid.net/
[11] csgrid.org/csg/
[12] http://www.distributed.net/Main_Page
[13] www.grid.org

when the Wi-Fi connection is enabled and available so that the user's mobile data is not burned up.

Vodafone Foundation, along with The Garvan Institute of Medical Research in Australia, came up with a similar project called DreamLab[14], intending to find a cure for cancer by pooling the public's smartphone computing power [150]. Like BOINC, volunteers simply have to install the DreamLab app and allow it to utilize their smartphone's ideal CPU cycles. The app automatically downloads the processing task, process it and sends the result back to the Institute. This helps accelerate research work to counter different cancers like breast, ovarian, prostate, and pancreatic cancer. The institute expected to speed up the data crunching processes up to 3000 times if 100,000 users lend their smartphones to this project.

Ubispark[15], a research project nurtured by the Department of Computer Science, University of Helsinki, aimed to utilise smartphones and other smart devices (e.g., smart TVs) for large-scale data processing related to science problems. Like BOINC for Android and DreamLab, Ubispark also can be downloaded as a regular Android app by users who are willing to contribute their device's idle computing power. As per the user's preference, the app can be set to contribute only when the device is connected to Wi-Fi and while it is being charged.

Neocortix[16], a USA-based company, is running a commercial MCC project where users lend their smartphone's idle processors and earn money. The users need to download the PhonePaycheck app. The app executes the tasks when the smartphone is on charge and connected to Wi-Fi. Neocortix has a range of services, all of which run on the public's Android SMDs. The Neocortix Cloud Services, launched in February 2020, provides a scalable compute service for which a full Debian Linux is run in a secure container on the SMDs, and is accessible via a standard SSH interface. The BatchRunner, launched in September 2020, allows users to launch their own customized batch jobs with minimalistic changes. They only need to edit a few lines of Python code and run a plain script. It offers many

[14] https://www.vodafone.co.uk/mobile/dreamlab
[15] https://ubispark.cs.helsinki.fi/
[16] https://neocortix.com/

benefits, such as supporting multiple programming languages, load testing tools, image renderers, machine learning environments, etc. The company provides user-friendly tutorials and a certification program for user learning and training. PocketScience, launched in February 2021, is a crowdsourced academic research app that allows Android/Arm device users to contribute to COVID-19 and other medical research.

2.2.3 Research Scope

From the above discussions, we observe that much work has been done toward MCC and other related areas. However, none of them covers MCC wholesomely. All of them either discussed different aspects of mobile-based computing discretely or presented some specific empirical works mainly to show the feasibility of such systems. The only paper we found with a similar presentation theme that of ours is by Lavoie and Hendren [151]. But they concentrated only on volunteer computing. Also, this paper does not discuss architectures, design criteria and considerations, advantages, or issues and challenges.

Moreover, most papers were published just after SMD was launched in the market. Only a few works have been published intermittently since then and also in recent years. Since the initial launching of the SMDs, they have evolved a long way. Today's SMDs are not only for making phone calls, browsing the internet or keeping personal notes; rather, they have become full-fledged computers. Many people use SMDs as computers by hooking them up to external keyboards and mice. In view of this development, MCC needs to be given a fresh look and perspective with the potential to achieve more.

Crowd computing has been exercised in sophisticated science and research projects in the form of volunteer computing (discussed in Section 2.2.2). But despite the apparent benefits and success of these projects, the idea of MCC is yet to find its feet in general user applications [152]. Even though MCC has an unbound potential for providing on-demand and ubiquitous computing services, it has not yet garnered sufficient attention from the research community and other related stakeholders, including general users. People need to be made aware of the MCC more convincingly. For this an exclusive documentation is needed that covers

every aspect, in general, required to know to understand MCC, which can fill the purpose of a preliminary reference for the interested researchers, both novice and experienced, who are keen to work on MCC, as well as other stakeholders willing to explore the benefits of MCC.

2.3 Resource Profiling in MCC

Profiling the potential resources is an important aspect of any distributed computing where the efficiency of the system mostly depends on the quality and capability of the computing resources. It becomes more crucial when we deal with mobile distributed computing systems such as mobile grid computing or mobile cloud computing, where the resource providing devices are mobile.

2.3.1 Profiling Mobile Devices' Information for Smartphone-based Computing

To realise enterprise-level volunteer computing using smartphones, Arslan *et al.* [35] profiled the mobile device charging behaviours of users for identifying suitable scheduling times. Using an Android application, they tracked three states - the phone being charged, not charging, and switched off. The application logged the event to a server with a timestamp (recorded at the user's end) whenever there was a change in state. Also, to assess the networking activity, they logged the number of bytes transferred over the wireless interfaces when they were in a charging state. They also measured the variations in the bandwidth of the mobile devices to precisely assess the data transmission performance for efficient task scheduling across different devices.

2.3.2 Research Scope

Though profiling is crucial for understanding the potential resources, surprisingly, a thorough search of the relevant literature yielded no related article, except the one discussed above, that deals with extensive resource profiling for mobile computing systems. This opens up a great opportunity to design and implement a suitable resource profiling framework for MCC.

2.4 Resource Selection in MCC Using MCDM Method

Choosing the best or the most suitable resources optimally among the available ones can significantly improve the effectiveness of any distributed system. Various

approaches are being tried for selecting the resources in mobile distributed computing environments such as mobile grids and mobile clouds. However, depending on the application requirements, the parameters are considered for selection may vary. On the other hand, MCDM techniques have been used for decision-making in several application domains for a long time [153] [154]. They have been extensively used in engineering [155]. Here, we highlight the use of MCDM in the related areas of distributed computing comparable to MCC. Before that, we point out some research work addressing optimized resource selection in a mobile computing environment.

2.4.1 Optimization-based Resource Selection in Mobile Grid/Cloud

Different optimization techniques are used by many researchers for resource selection and task allocation and scheduling. Zhou *et al*. [156] proposed an optimal mobile device selection approach for a mobile cloud computing environment considering the stability of the devices. To find out the mobile devices with maximized usable computation capabilities in a mobile cloud system, Habak *et al*. [140] adopted a greedy heuristic based optimization method that would select the most suitable mobile device among the available ones. They considered the computation and bandwidth capacity of the device and its departure time as the selection parameters. Venkatraman *et al*. [157] proposed a linear programming based model to select the best mobile devices in a mobile ad-hoc cloud, considering the devices' resources such as CPU, RAM, storage, etc., aimed to use. Viswanathan *et al*. [158] used an optimization approach to select a mobile node as a computing service provider in a mobile grid computing for ubiquitous healthcare. Among the considered selection parameters such as CPU, RAM, available battery, network resources, the present location of the node and its availability period, some were intended to be maximized (e.g., response time and resource availability duration) while some were minimized (e.g., battery consumption).

2.4.2 MCDM for Resource Selection in Distributed Computing

Besides web service selection [159] [160], MCDM methods are also popularly used for cloud service selection [161] [162] [163]. Youssef [164] used a combination of TOPSIS and BWM to rank cloud service providers based on nine service evaluation

criteria, including sustainability, response time, usability, interoperability, cost, maintainability, reliability, scalability, and security. Singla *et al*. [165] used Fuzzy AHP and Fuzzy TOPSIS to select optimal cloud services in a dynamic mobile cloud computing environment. They considered resource availability, privacy, capacity, speed, and cost as selection criteria.

MCDM methods are being used to improve the efficiency and effectiveness of job offloading in mobile cloud computing [166] [167]. To choose the suitable hosts (e.g., cloud, cloudlet, and peer mobile devices) for offloading the computing tasks from a mobile device, Ravi and Peddoju [168] used the TOPSIS method. They considered the waiting time, the energy required for communication, the energy required for processing in mobile devices, and connection time with the resource as the selection criteria.

Mishra *et al*. [169] proposed an adaptive MCDM model for resource selection in fog computing, which could accommodate the new-entrant fog nodes without reranking all the alternatives. The proposed method is claimed to have less response time and is suitable for a dynamic and distributed environment.

To ensure the quality of the collected data in mobile crowd sensing applications, Gad-ElRab and Alsharkawy [170] used the SAW method for selecting the most efficient devices based on computation capabilities, available energy, sensors attached to the device, etc.

MCDM methods have been used for resource selection in grid computing as well. For instance, Mohammadi *et al*. [171] used AHP and TOPSIS combinedly for grid resource selection. They considered cost, security, location, processing speed, and round-trip time as selection criteria. Abdullah *et al*. [172] used the TOPSIS method to select resources for fair load balancing in a multi-level computing grid. For resource selection, they considered three criteria expected completion time, resource reliability, and the resource's load. Kaur and Kadam [173] used MCDM methods for a two-phased resource selection in grid computing. They applied the SAW method to rank the best resources in the local or lower level and then used enriched PROMETHEE-II combined with AHP for a global resource selection or to select the best resources across all the top-ranked resources at each local level. Nik *et al*. [174]

used the TOPSIS method to select the resource with the best response time for asynchronous replicated systems in a utility-based computing environment. To achieve a shorter response time, they considered four QoS parameters (efficiency, freshness of data, reliability, and cost) as selection criteria.

2.4.3 MCDM for Smartphone Selection

Several works are proposed for evaluation and selection of smartphones [175] [176] [177] [178] [179] [180] [181]. Various aspects were considered for selection by matching the consumers' choices and interests. However, in all these works, smartphones were considered consumer devices than computing devices.

2.4.4 Comparing Different MCDM Methods

Triantaphyllou, in his book [182], extensively compared the popular MCDM methods such as WSM, WPS, TOPSIS, ELECTRE, and AHP (along with its variants). The methods were discussed based on real-life issues, both theoretically and empirically. A sensitivity analysis was performed on the considered methods, and the abnormalities with some of these methods were rigorously analysed. Velasquez and Hester [183] performed a literature review of several MCDM methods, viz., MAUT, AHP, fuzzy set theory, case-based reasoning, DEA, SMART, goal programming, ELECTRE, PROMETHEE, SAW, and TOPSIS. This study aimed to analyse the advantages and disadvantages of the considered methods and examine their suitability in specific application scenarios. Besides these, several other authors attempted to present comparative studies of different MCDM methods with respect to different application areas. Table 2.2 presents a comprehensive list of such works.

2.4.5 Research Scope

As mentioned in Section 2.4.3, the existing works that have used MCDM for smartphone selection are consumer-centric. Here the main purpose of smartphone selection is to determine the best device according to the consumer's choice and preference. These selection criteria focus more on the cosmetic aspects of the SMDs, such as screen size, camera, design, etc., than their computing. In fact, we could not find any work that applied MCDM for selecting SMDs as computing resources. Moreover, these selection factors are very much fixed, whereas, in MCC, the selection parameters are dynamic in nature. Considering this, we need to

search for an MCDM method that would be suitable for the resource selection problem in MCC.

As mentioned in Section 2.4.4, several works exist which tried to find out the most suitable MCDM method appropriate for a particular application or problem domain. However, despite our best effort, we could not find any comparative analysis of MCDM methods for resource selection in a dynamic environment like MCC or any other related applications. From Table 2.2, it can be observed that barring only a few works, none has conducted a time complexity analysis. Furthermore, we found not a single paper that calculated the actual runtime of the MCDM algorithms. For swift job scheduling, which in turn would lead to improved throughput, the resource selection algorithm should be time-efficient. This prompted us to search for such an MCDM method.

Table 2.2. Survey of comparative analysis of different MCDM methods

Ref.	MCDM methods compared	Application focus	Analysis performed				
			Sensitivity analysis	Result comparison	Statistical test/analysis	Rank reversal	Computation/time complexity
[184]	ELECTRE, TOPSIS, MEW, SAW, and four versions of AHP	General MCDM problem of ranking	√	√	√	√	
[185]	AHP and SAW	Ranking cloud render farm services	√	√	√		
[186]	TOPSIS, AHP, and COMET	Assessing the severity of chronic liver disease		√	√		
[187]	CODAS, EDAS, WASPAS, and MOORA	Selecting material handling equipment		√	√		
[188]	TOPSIS, DEMATEL, and MACBETH	ERP package selection	√	√	√		
[189]	AHP, ELECTRE, TOPSIS, and VIKOR	Enhancement of historical buildings		√	√		
[190]	MOORA, TOPSIS, and VIKOR	Material selection of brake booster valve body		√	√		
[191]	AHP, TOPSIS, and VIKOR	Manufacturing process selection		√	√		√
[192]	Multi-MOORA, TOPSIS, and three variants of VIKOR	Randomly generated MCDM problems (i.e., decision matrices) as per [184].	√	√	√		
[193]	WPM, WSM, revised AHP, TOPSIS, and	Sustainable housing affordability	√	√	√		

Ref.	MCDM methods compared	Application focus	Analysis performed				
			Sensitivity analysis	Result comparison	Statistical test/analysis	Rank reversal	Computation/time complexity
	COPRAS						
[194]	SAW, TOPSIS, PROMETHEE, and COPRAS	Stock selection using modern portfolio theory		√	√		
[195]	COMET, TOPSIS, and AHP	Assessment of mortality in patients with acute coronary syndrome		√	√		
[196]	SWARA, COPRAS, fuzzy ANP, fuzzy AHP, fuzzy TOPSIS, SAW, and EDAS	Risk assessment in public-private partnership projects	√	√	√		
[197]	WSM, VIKOR, TOPSIS, and ELECTRE	Ranking renewable energy sources	√	√	√		
[198]	WSM, WPM, WASPAS, MOORA, and MULTIMOORA	Industrial robot selection	√	√	√		
[199]	WSM, WPM, AHP, and TOPSIS	Seismic vulnerability assessment of RC structures	√	√	√		
[200]	AHP, TOPSIS, and PROMETHEE	Determining trustworthiness of cloud service providers	√	√	√		
[201]	TOPSIS and VIKOR	Finding the most important product aspects in customer reviews		√	√		
[202]	MABAC and WASPAS	Evaluating the effect of COVID-19 on countries' sustainable development	√	√	√		
[203]	WSM, TOPSIS, PROMETHEE, ELECTRE, and VIKOR	Utilization of renewable energy industry	√	√	√		
[204]	WSM, TOPSIS, and ELECTRE	Flood disaster risk analysis	√	√	√		
[205]	MAUT, TOPSIS, PROMETHEE, and PROMETHEE GDSS	Choosing contract type for highway construction in Greece		√	√		
[206]	TOPSIS, VIKOR, EDAS, and PROMETHEE-II	Suitable biomass material selection for maximum bio-oil yield		√	√		
[207]	TOPSIS, VIKOR, and COPRAS	COVID-19 regional safety assessment	√	√	√		
[208]	EDAS and TOPSIS	General MCDM problem	√	√	√	√	
[209]	AHP, TOPSIS, ELECTRE III, and PROMETHEE II	Building performance simulation	√	√	√		
[210]	AHP, fuzzy AHP, and ESM	Aircraft type selection		√	√		
[211]	AHP, TOPSIS, and SAW	Intercrop selection in rubber plantations		√	√		

Ref.	MCDM methods compared	Application focus	Analysis performed				
			Sensitivity analysis	Result comparison	Statistical test/analysis	Rank reversal	Computation/time complexity
[212]	AHP, TOPSIS, SAW, and PROMETHEE	Employee placement		√	√		
[213]	TOPSIS, VIKOR, improved ELECTRE, PROMETHEE II, and WPM	Mining method selection		√	√		
[214]	AHP, SMART, and MACBETH	Incentive-based experiment (ranking coffee shops within university campus)		√	√		
[215]	AHP, fuzzy AHP, and fuzzy TOPSIS	Supplier selection		√	√		
[216]	TOPSIS, SAW, VIKOR, and ELECTRE	Evaluating the quality of urban life	√	√	√		√
[217]	AHP, MARE, ELECTRE III	Equipment selection		√	√		
[218]	VIKOR and TOPSIS	Forest fire susceptibility mapping		√	√		
[219]	PIPRECIA, MABAC, CoCoSo, and MARCOS	Measuring the performance of healthcare supply chains	√	√	√	√	
[220]	MOORA, MULTIMOORA, and TOPSIS	Optimize the process parameters in the electro-discharge machine	√	√	√		
[221]	AHP, AHP TOPSIS, and fuzzy AHP	Mobile-based culinary recommendation system		√	√		√
[222]	TOPSIS, COPRAS, and GRA	Evaluation of teachers		√	√		√
[223]	AHP, TOPSIS, ELECTRE III, and PROMETHEE II	Urban sewer network plan selection		√	√		
[224]	TOPSIS and AHP	Dam site selection using GIS		√	√		
Our work	EDAS, ARAS, MABAC, COPRAS, and MARCOS	Resource selection in MCC (discussed in Chapter 5)	√	√	√		√

2.5 Task Scheduling in MCC

The success of any distributed and collaborative computing based systems very much depends on the proper scheduling of the tasks at hand. However, optimised task scheduling in a distributed system, especially in a dynamic environment, is nontrivial. Researchers have tried different approaches to achieve optimal task scheduling, considering different parameters pertinent to the application or the executable task.

With the proliferation of soft computing techniques and artificial intelligence,

nature-inspired algorithms like PSO, GA, etc., are gradually getting prominence in the current era because of their ability to find a near-optimal solution for NP-complete problems. Several works are there in which optimisation techniques have been used for resource selection and task allocation, and scheduling under various considerations in a range of distributed computing systems [225] such as grid computing [226] [227], cloud computing [228] [229] [230], mobile cloud computing [156], mobile edge computing [231], vehicular cloud [232] [233], and software-defined networks [234].

2.5.1 Resource-Aware and Multicriteria-based Scheduling in Mobile and Distributed Computing

Habak *et al.* [140], in their proposed mobile cloud, named FemtoClouds, aimed to maximise the usable computation capabilities of the mobile devices for optimised task scheduling. They followed a greedy heuristic based optimisation approach to select the desired device considering the parameters like the computation and bandwidth capacity of the device and its departure time.

For parallel task distribution to a pool of mobile devices in a mobile ad-hoc cloud, Venkatraman *et al.* [157] calculated the composition score for the shareable resources (e.g., CPU, RAM, storage) of the devices. Based on the composition score, they used a linear programming based model to map the tasks to the mobiles ideally so that all the tasks could be executed using a minimum number of devices.

Viswanathan *et al.* [158] proposed a mobile grid computing framework for ubiquitous healthcare where the service-providing nodes were selected based on several parameters such as CPU power, memory, available battery, network resources, the current position of the node, and its availability period. Based on the number of available service providers for the required duration, the tasks were optimally distributed and scheduled to them based on certain optimising objectives such as minimising the battery drain while maximising the response time and availability time of the resource. For this problem, they adopted a self-optimisation and self-healing approach.

Various optimisation techniques and strategies have been tried and adopted for

scheduling in multi-core and heterogeneous distributed systems, considering different scheduling and optimising criteria [235] [236] [237]. Akbari *et al.* [238] presented a genetic algorithm based static task scheduling method for heterogeneous computing systems. They aimed to obtain near-optimal solutions with reasonable execution time. Sulaiman *et al.* [239] proposed a genetic algorithm based heuristics for task scheduling for heterogeneous computing systems. They used a list-based approach and guided random search to reduce the schedule length and the procedural complexity. Biswas *et al.* [240] proposed multicriteria-based scheduling algorithms for heterogeneous computing systems to minimise the makespan, energy consumption, and load-balancing while maximising resource utilisation. They considered the resource parameters such as CPU speed, memory capacity and residual energy of the systems. The same authors applied PSO [241] and GA [242], to achieve optimised scheduling in muti-core systems to minimise makespan and load balancing and maximise resource utilisation and speed-up ratio.

2.5.2 Energy-Efficient Scheduling in Mobile and Distributed Computing

Optimisation techniques are used for energy-efficient scheduling in heterogeneous and high-performance computing systems [243] [244] [245]. Considering the battery limitations of the mobile devices, to conserve the energy, researchers proposed to offload the power-demanding jobs to the cloud [246] [247] or to other mobile devices [108] [248] [249]. Several works have been proposed on energy-efficient scheduling for task offloading from mobile devices to the cloud [250] [251] [252].

Shah and Park [253] presented an energy-efficient resource allocation scheme for an ad-hoc computational grid of mobile nodes. They aimed to minimise the communication energy of the nodes that are allocated some interdependent tasks to execute. The authors used the kNN search algorithm to find a group of closest nodes in a grid so that the overall data travelling path became minimum, which would minimise energy consumption and communication costs. Shah *et al.* [254] proposed another resource allocation scheme for such mobile ad-hoc computational grids to reduce communication costs, including energy costs. They utilised user mobility patterns and task dependencies to minimise the communication cost. Viswanathan *et al.* [158] adopted a self-optimisation and self-healing

approach to minimise the battery drain while maximising the resource's response time and availability time in a mobile grid computing framework for ubiquitous healthcare.

Chen *et al.* [255] proposed a heuristic algorithm for resource allocation in an ad-hoc mobile cloud. They assumed that the devices could harvest energy from the ambient environment. Hence, their main goal was to minimise the response time rather than energy consumption. Bonan *et al.* [256] proposed a PSO-SA (simulated annealing) based dependent task assignment algorithm for ad-hoc mobile cloud. They considered time delay, the total energy consumption and fairness of resource usage of all devices simultaneously. Shi *et al.* [257] proposed an adaptive scheduler for local mobile clouds, where the tasks from multiple source nodes could be scheduled to nearby processing nodes. They aimed to minimise the execution time of the tasks and the energy consumption.

2.5.3 Research Scope

From the above-cited works, we see that several scheduling algorithms and approaches for mobile and distributed computing systems are proposed in the literature. However, we could not find any such readymade solution that suits our proposed concept of MCC. Despite our best effort, we could not find any work that considered the dynamic resources of SMDs to achieve efficient multicriteria scheduling in MCC. Though there are a few similar works (e.g., [140] , [157], [255], [256], [257]) that attempted task scheduling in mobile clusters, the scale of simulation (in terms of the number of mobile devices and task size and variety) are not wide-ranging. Whereas in our experiment, simulation is done on 50 SMDs for two sets of tasks with the size of 100 and 200 instructions. Also, we consider several additional resource parameters that are crucial to adjudge the practical fitness of an SMD as a computing resource. We could not find any energy-efficient scheduling for MCC that considers load balance also which is very crucial especially in SMD-based distributed crowdsourced computing. Hence, it open ups the scope for designing efficient scheduling algorithms that fulfil the objectives required to enhance the performance of MCC while maintain user satisfaction. A comparative summary of the related works and this paper is presented in Table 2.3.

Table 2.3. Summary of the works related to resource scheduling

Reference	Application area	Problem addressed	Goal	Solution method/tool	Parameters considered	Performance Compared with	Experimental data/setup	Attainment	Limitations/ further scope
Shah *et al.* [254]	Mobile ad-hoc computational grids	Resource allocation	Minimize: communication costs Reduce: task failure.	Markov chain model (for mobility management), proposed two-phase resource allocation scheme (TPRA) (for task allocation).	Node mobility, node distance, and task dependencies.	Distance-based resource allocation (DRA) and next location-based resource allocation (NLRA) schemes [258].	Simulated scenario	As per simulation results, the proposed scheme significantly reduced the communication cost between interdependent tasks.	Computational cost was not considered.
Viswanathan *et al.* [158]	Mobile grid computing	Selecting service-providing nodes.	Maximize: minimum residual battery capacity.	Self-optimisation and self-healing.	CPU power, memory, available battery, network resources, current position of the node, and its availability period.	Round-robin, FCFS-based CometCloud [259].	Testbed of multiple Android-based mobile devices with heterogeneous capabilities for distributed object recognition.	The proposed solution achieved symmetric battery drain by exploiting the varied battery capacity of the devices.	Makespan and resource utilization were not considered as optimization criteria.
Habak *et al.* [140]	Mobile cloud computing	Task scheduling	Maximize: overall throughput of the cluster, resource. utilization, and network	Greedy heuristic	Computation and bandwidth capacity of the mobile device and its	A presence time oblivious scheduler (PreOb) and an emulated. arrival/departure	Experimental prototype designed using Android devices.	The proposed prototype offered better computation performance while consuming fewer	Very small-scale simulation setup.

			utilization.		departure time.	scenario.		network resources.	
Venkatraman *et al.* [157]	Ad-hoc mobile cloud	Task scheduling	Minimize: number of devices required for task execution.	LPP-based model - an adapted bin packing algorithm.	CPU, RAM, storage.	Wang *et al.* [260]	Simulator designed that comprised the subtask traffic generator and topology generator modules.	Selecting the resources based on the total resource composition score improved the resilience and overall computation performance.	Scheduling was not done by accurately assessing the total resource requirement. Hence the allocated tasks needed to be partially offloaded.
Chen *et al.* [255]	Ad-hoc mobile cloud	Resource allocation	Minimize: response time and overall task completion time.	Proposed heuristic algorithm.	CPU cycles, energy consumption for task execution.	With different simulation parameter considerations.	Matlab simulation with random mobile device and task generation	The overall response time was reduced for various task dependency topologies and task sizes.	Performance was not compared with other algorithms or similar work.
Bonan *et al.* [256]	Ad-hoc mobile cloud	Dependent task assignment	Minimize: time delay and total energy consumption Improve: fairness of resource usage.	PSO-SA (simulated annealing) based algorithm.	Each task's data and instruction size, and dependency.	Binary PSO and GA.	A simulated ad-hoc mobile cloud scenario that virtually spans for 100m2.	For large number of tasks and large number of working nodes, the proposed algorithm performed better.	Overall throughput and the energy consumption for receiving results were not considered.
Shi *et al.* [257]	Local mobile cloud	Dynamic task scheduling	Minimize: execution time and energy consumption.	Proposed distributed adaptive probabilistic Scheduler.	Data size, computation capacity, communication distance.	Round robin, greedy, and probabilistic schedulers.	A local mobile cloud simulation setup developed on OMNET++	Achieved satisfactory throughput and energy efficiency for varied task sizes, and can be a viable solution for scheduling tasks of real-time	Load balance and resource utilization were not considered.

								applications.	
Akbari *et al.* [238]	Heterogeneous computing systems	Static task scheduling	Minimize: execution time Maximize: parallel task assignment.	Proposed GA-based algorithm.	Processors' computation capacity, tasks' computation requirement, task parallelism.	HEFT-T, HEFT-B, CPOP [261], BGA [262], SA [263], and SLPSO [264] algorithms.	Simulated using C# by generating various graphs, e.g., fast Fourier transformation (FFT), molecular, Gaussian graphs and random graph with different parameters.	The proposed algorithm achieved better scheduling with less iterations.	Load balance and resource utilization, and energy consumption were not considered.
Sulaiman *et al.* [239]	Heterogeneous computing systems	Static task scheduling	Minimize: schedule length Reduce: procedural complexity.	GA-based heuristic (list-based approach and guided random search).	Computation and communication costs.	New GA (NGA) [265], enhanced GA for task scheduling (EGA-TS) [238], heterogeneous earliest finish time (HEFT) [261], and predict the earliest finish time (PEFT) [266].	Simulated dataset, generating four types of graphs, i.e., random graphs, Gaussian elimination, FFT, and molecular dynamic code task graph, having diverse attributes, such as the number of tasks in the graph, graph	Achieved better results than the compared algorithms in terms of best result occurrences, average makespan, average schedule length ratio, average speed-up, and the average running time and quicker convergence time.	The convergence speed could be bettered. There is a scope for using a multi-objective fitness function.

							shape, nodes' out-degree, number of processors, and communication and computation costs.		
Biswas *et al.* [240]	Heterogeneous computing systems	Multicriteria-based scheduling	Minimize: makespan, energy consumption, and load-balancing Maximise: resource utilisation.	Proposed resource-aware heuristic algorithm.	CPU speed, memory capacity and residual energy.	Min-min, MCT, priority based performance improved algorithm (PPIA), and GA.	Simulated on synthetic and standard benchmark datasets.	Achieved better results in the simulated consideration.	Considered resource parameters were less.
Biswas *et al.* [241]	Heterogeneous computing systems	Dependent workflow scheduling	Minimize: makespan and load balancing Maximize: resource utilization and speed-up ratio.	Proposed PSO-based algorithm.	Dynamicity in CPU speed, instruction length, and resource capability.	Gravitational search algorithm (GSA) Cloudy-GSA, and HGSA.	Simulated on synthetic and standard benchmark datasets.	Achieved satisfactory results compared to Cloudy-GSA, and HGSA and equivalent performance with the GSA.	Energy consumption was not considered.
Biswas *et al.* [242]	High-performance mutiicore systems	Independent workflow scheduling	Minimize: makespan and load balancing Maximize: resource utilization and speed-up ratio.	Proposed GA-based algorithm.	Dynamicity in CPU speed, instruction length, resources capability and mutation factors.	GA, PSO, and PPIA	Simulated on synthetic and standard benchmark datasets.	The proposed algorithm performs considerably better than GA and PPIA. For makespan and load balance, it performs better than GA; however, for resource utilization and computation speed up the	Energy consumption was not considered

								performance of the proposed algorithm and PSO are closely similar.	
Our work 1	MCC	Multicriteria-based dynamic scheduling for SMDs.	Minimize: makespan and load balancing Maximise: resource utilisation.	Proposed heuristic algorithm.	CPU (clock frequency, no. of cores, and current load), RAM (currently available), battery (total capacity and current charge %), and device temperature.	PSO, GA, and MCT	Synthetic as well as real dataset (generated specifically for MCC experiment).	For all the scenarios, the proposed algorithm significantly outperforms PSO and GA. Though it performs better than MCT, in some scenarios, their performances are close.	Energy consumption is not considered, which can be crucial for mobile devices.
Our work 2	MCC	Energy-efficient task scheduling.	Minimise: overall energy consumption with load balance.	PSO-based proposed algorithm.	CPU clock frequency, no. of CPU cores, and current CPU load.	MCT, MinMin, MaxMin, and PPIA, and GA.	Synthetic as well as real dataset (generated specifically for MCC experiment).	Proposed algorithm performs significantly better than others while only energy efficiency is considered. For energy efficiency with load balance, the performance difference is slightly marginal.	Energy consumption for data transfer is not considered.

2.6 Resource Availability Prediction in MCC

Assessing the availability of the resources-providing entities in a distributed

computing environment is crucial because it affects the system's QoS considerably. Though in several works, to achieve an improved job scheduling, the suitable resources are targeted based on different criteria, there is not much work found that addresses the availability issue of the resource, especially in a dynamic environment. In this section, we try to identify the research works that exactly or subliminally address the resource availability problem in different distributed systems. We also review the use of deep learning in closely related problems.

2.6.1 Resource Availability Prediction in Mobile Grid/Cloud Computing

Brevik *et al*. [267] aimed at enabling the grid job scheduler to make on-the-fly decisions by providing live availability predictions. To predict the availability duration of a resource, they used the Weibull method (parametric model fitting technique) along with Resample and Binomial methods (non-parametric techniques). The authors attempted to estimate a specific quantile for the availability distribution and the confidence for each estimation. Andrzejak *et al*. [268] attempted to predict the availability of the grid resources within a time interval [T, T+p], where p is the prediction interval length with values [1,2]. For this, they used the Naive Bayes and Decision trees based predictive models. They also aimed to identify the resource predictability indicators and the factors that incite prediction error. But these works do not cover the availability prediction of mobile devices in a non-dedicated mobile grid environment.

Vaithiya and Bhanu [269] proposed a task scheduling algorithm for the mobile grid, predicting the dynamic availability of mobile resources. To address the node mobility issue in an ad-hoc mobile grid, Selvi *et al*. [270] profiled the mobile users' regular movements over time. But none of these considers the historical characteristics of the devices, which may hinder achieving the optimal effects in SMD selection.

In FemtoClouds, a mobile device cloud control system presented by Habak *et al*. [140], the presence time prediction of mobile devices is incorporated. In this work, it is assumed that the controller has knowledge of the exact departure time of each device for each session. But, in practice, some dishonest users may depart before the declared departure time, while some may be forced to cut off from the network

due to some genuine reasons such as battery used up. To counter this problem, Zhou *et al.* [156] suggested considering the historical characteristics of the devices to evaluate the record of honouring their departure promise. They proposed a mobile device selection method, considering the status and stability of the devices. However, both of these works assume that the SMDs declare the departure time voluntarily, which may not be practical.

Sipos and Ekler [271] proposed a method to estimate mobile devices' availability where these devices were used to form a distributed storage system in a P2P (peer-to-peer) fashion. They predicted the actual availability based on the nodes' self-declared availability or unavailability for the subsequent considered time period. Different classifiers were used to check the accuracy of the prediction model in the simulated mobility scenario.

Haryanti and Sari [272] predicted the mobility of a group of resource-providing nodes with respect to a resource-seeking node. The purpose was to ensure that the task from the requesting node should be given only to those resource-providing nodes which are supposed to be in contact with the requesting node until the task is completed. Farooq and Khalil [121] also proposed a method to predict a time duration for which a resource-requesting node would remain within reach of the resource-providing node in a mobile grid. Based on the predicted time, the task assignment decision is taken. The prediction is based on the previous record of the time duration of their contact, whereas the contact is calculated by the distance between them based on their locations, assessed by their GPS coordinates.

2.6.2 Deep Learning for Resource Management and Prediction

Considering its potential, deep learning has been applied in various domains and applications for different purposes [273] [274] [275] [276] [277] [278] [279] [280] [281] [282] [283]. Specifically, in time-series forecasting, LSTM [284] [285] [286] and GRU [287] [288] [289] are widely used.

Many researchers exploited the convolutional aspect of CNN in combination with LSTM to improve the performance of time-series prediction/forecasting in various applications, such as for inventory prediction [290], stock price prediction [291]

[292] [293] [294], gold price forecasting [295], Bitcoin price forecasting [296], tourist flow forecasting [297], sentiment prediction of social media users [298], household power consumption prediction [299] [300], photovoltaic power prediction [301], wind power forecasting [302], $PM_{2.5}$ prediction [303] [304], predicting NO_x emission in processing of heavy oil [305], forecasting natural gas price and movement [306], urban expansion prediction [307], predicting waterworks operations at a water purification plant [308], predicting sea surface temperature [309], typhoon formation forecasting [310], crop yield prediction [311], COVID-19 detection and predictions [312] [313] [314], human age estimation [315], and so on.

Deep learning based techniques are used for efficient resource management and prediction in cloud [316] [317] [318] [319] [320] [321], edge computing [322] [323] [324] and other wireless distributed systems [325] [326] [327] [328] [329].

The inherent capability of capturing short-term as well as long-term instances has led LSTM [330] [331] [332], CNN [333], and convolutional LSTM [334] [335] to be popularly used in mobility predictions. In his master's thesis [336], Pamuluri compared different deep learning methods, including LSTM, CNN-LSTM, and GRU, to predict users' mobility with respect to a mobile base station. Cui *et al.* [337] used LSTM to predict the availability of mobile edge computing-enabled base stations depending on the vehicle's mobility for offloading the computation jobs from the vehicle to the base station. Li *et al.* [338] used LSTM to track user mobility for efficient dynamic resource allocation across different network slices in a 5G network.

2.6.3 Research Scope

As we mentioned above, assessing the availability of resources in a dynamic distributed computing system is very crucial. In a truly dynamic environment like MCC, it becomes more vital because here, the resources are non-dedicated and tend to be in mobility unexpectedly. However, in spite best of our effort, we could not find any significant work that endeavours to predict the availability of the SMDs or the users in a dynamic MCC environment.

2.7 Mobility-Aware Service Provisioning for P2P MCC

It is normal nowadays for users to run numerous applications on their SMDs. They

perform several essential and not-so-essential tasks using their SMDs. Some of these tasks are resource-demanding, due to which all SMDs, especially with lower resource specifications, cannot afford to execute them. In these cases, an affordable and feasible option is to take the help of the neighbouring SMDs. In other words, when an SMD needs some additional resource, it avails the resource from a peer SMD, if available. When this approach is applied to MCC, we call it P2P MCC. Since, in this type of computing system, both the resource provider and consumer are mobile, it is crucial to address the mobility issue. In this section, we focus on the research works related to such P2P mobile computing, mobility tracking and prediction, and mobility-aware service provisioning.

2.7.1 P2P Mobile Computing

Considering the limitations of low-constraint mobile devices, researchers nurtured the idea of P2P mobile ad-hoc grid computing [248] [339] [340]. To avoid the latency involved in traditional cloud services, a mobile ad-hoc cloud has been proposed where a number of mobile devices cooperatively form a local, minuscule, and ad-hoc cloud [129] [341] [342]. To study the feasibility of the ad-hoc mobile cloud, Büsching *et al*. [93] built a small-scale proof-of-concept cluster with six Android devices connected through Wi-Fi. Huerta and Lee [129] presented a framework of a virtual mobile cloud that detects the nearby mobile devices of the users, which will stay in the same area or follow the same movement pattern. Shi *et al*. [108] presented a P2P mobile computing system in which a computational task is offloaded from a resource-constrained mobile device to other mobile devices with which the initiator device comes into contact on its route. The authors assumed an ideal network environment where the future contact between resource providers and consumers can be predicted accurately. However, the system does not address the node mobility issue.

2.7.2 Mobility Prediction Approaches

The majority of the mobility prediction schemes collect users' movement history somehow and analyse them to assess and predict the mobility patterns of mobile users. The prediction accuracy depends not only on the method used for collecting the mobility patterns but also on the algorithm used for mobility prediction.

Typically, the research on mobility prediction for the infrastructure-based wireless network is mainly aimed to improve the performance of wireless networks on different aspects such as admission control, QoS, handoff management, etc. These algorithms, as per literature, can be classified into three major approaches, as shown in Fig. 2.1 and discussed in the following subsections.

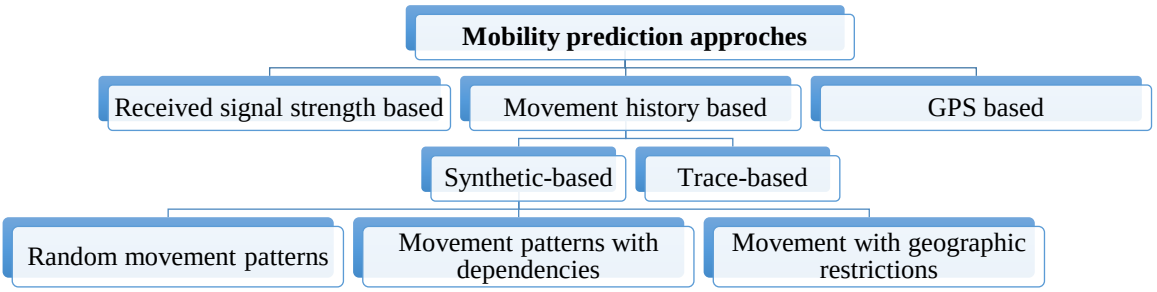


Fig. 2.1. Mobility prediction approaches

2.7.2.1 Movement History Based Mobility Prediction

This class of algorithms predicts the future location of the mobile nodes with the help of the past and current states of the user and can be further classified into two categories, as discussed in the following.

2.7.2.1.1 The Synthetic Based Mobility Model

The synthetic-based mobility prediction uses mathematical models (e.g., sets of equations), which try to capture the movement of the devices by studying users' mobility [343] [344]. There is significant research on mobility prediction for cellular networks available in the literature, where the prediction approach is based on the user's movement history [345].

The mobility prediction algorithm proposed by Lium and Maguire [346] considered the notion that every human has some extent of consistency in his/her movement. The proposed approach consists of regularity detection algorithms (for predicting users' future regular movement) and a motion prediction algorithm (for predicting the next states of a user's movement). For predicting the subsequent movement trail of the mobile user, the motion prediction algorithm utilizes the database of regular movement patterns of the users and the random probability information with constitutional constraints. Therefore, this model is well suited for regular patterns like hourly, weekly, etc. The regular and random components

of user movements can correspond with circle/track patterns, or they can be simulated by the Markov chain model also.

Liu *et al*. [347] proposed a hierarchic position prediction algorithm in which user's movements are mapped to previous mobility patterns by the proposed mobility prediction model.

For simulating the highway traffic that comprises mobile entities travelling in forward and backward directions with different constant speeds, a shadow cluster concept was proposed by Levine *et al*. [348]. In this distributed framework, mobile terminals inform the neighbouring base station about their requirements, position, and movement parameters, based on which the base stations predict future demands and reserve resources accordingly. To predict mobility, it uses the user's movement history traces.

Mobility prediction solutions proposed in [349] and [350] assume that the nodes move according to the random way point mobility model [351]. The mobility prediction algorithm proposed in [352] used different mobility models such as the movement circle model, movement track model, and the Markov chain model to model the user mobility behaviour.

Synthetic mobility models are generally simple to model and implement and easy to use and have been reported as an effective means to solve problems like signal attenuation and also consume less power. However, the random nature of the synthetic mobility model often fails to capture human movement patterns correctly and leads to unrealistic-scenarios and non-uniform distribution [353]. The observation made and the conclusion drawn from such models may be misleading, especially where the user movements are predictable to some extent (e.g., mobile users in an educational campus, office, etc.).

2.7.2.1.2 Trace Based Mobility Prediction

In trace-based mobility prediction, the traces are obtained by using the deployed systems' measurements and typically consist of connectivity logs or location information. A trace-based mobility model is developed based on datasets collected from real scenarios by tracing or monitoring the movements of the persons

carrying mobile devices. The collected movement traces are then analysed to find specific mobility patterns of those mobile users.

A significant amount of research work has been done to predict mobility using movement traces of real-life users [354]. The data collected for movements of real users are used to predict future locations of the users using various prediction methods.

Kim *et al*. [355] proposed a method for estimating the physical location of users from a sizeable trace data of mobile devices associated with APs in a wireless network. The mobility traces collected by Dartmouth College is used to extract users' locations and their movement trails which were estimated using methods like triangle centroid, time-based centroid and Kalman filter.

The method proposed by Khalifa and Abbas [356] is supposed to predict the future locations of mobile users and the duration for which they would remain at those locations. The algorithm is evaluated using the UCSD dataset. The method is found to have better prediction accuracy than a third-order (O(3)) Markov predictor [357].

Burbey and Martin [358] used the UCSD dataset to evaluate the proposed Markov model based mobility prediction method. The method is based on the data compression algorithm called Prediction by Partial Match, which predicts the location of a user at a given time. The model was tested by trying to predict the location of a user by giving a particular future time.

The mobility model proposed by Musolesi and Mascolo [359] uses the concept of social networking, in which the hosts are grouped based on their social relationships. This clustering is mapped to a topographical space, where the potency of dynamic social ties determines the movements. The model was validated with real traces indicating the potential of the synthetic mobility traces as a decent approximation of human movement patterns.

The Autoregressive Hello protocol was proposed by Li *et al*. [360] for neighbourhood discovery. Here, each node and its neighbouring nodes predict their positions through an ever-updated, auto regression-based mobility model. Each node

estimates its neighbour's position regularly, using previous location traces. When the predicted location is very far from the actual location, which causes a topology distortion, the node transmits a 'hello' message to the state and updates its current location. This helps in the autocorrection of the mobility model of the concerned node and its neighbours.

The prediction of topological changes in trace-based mobility prediction requires constant prediction of the node's location, which may be considered an overhead in a resource-constrained environment. However, if the location is predicted from the change of neighbourhood, then it reflects the topological change in the network and helps to determine the node's stability, which is crucial for selecting stable SMDs in PMC.

2.7.2.2 Received Signal Strength Based Mobility Prediction

In this approach, the mobility of a node is estimated by measuring how much the received signal strength is dependent on the distance from the source. The estimated values of the node's location and its mobility information are acquired using signal attenuation versus distance travelled. This type of mobility prediction method is very simple [361].

A novel mobility metric for mobile ad-hoc networks (MOBIC) was proposed by Basu *et al.* [362]. Here, the node with the lowest mobility in the neighbourhood is selected as the cluster head. Each node assesses the signal strength received from its neighbours continuously, and based on the variance, the movement rate of that node relative to the neighbouring nodes is estimated. This mobility measurement is used for the formation of mobile clusters to improve the scalability of different services. For selecting a cluster head, MOBIC employs Aggregate Local Mobility (ALM), a novel mobility metric. However, in the cluster maintenance phase, nodes' mobility behaviours are not always considered, and hence, a cluster-head does not guarantee to stand a low mobility characteristic [363].

As an extension of MOBIC, MobDHop, proposed by Er and Seah [364], used a distributed algorithm to form a stable cluster that can serve as underlying routing architecture. The variance in received signal strength is used to predict the

neighbourhood mobility and, based on mobility metric, forms a d-Hop cluster, as suggested in [362]. Compared to MOBIC, it exercises more samples of the received signal to estimate the predicted mobility. The prediction model assumes that the mobility patterns of nodes will be exactly the same in the future as they were in the recent past. In MAPLE [365], based on the signal strength received from the cluster head, each host estimates its distance from the corresponding cluster head.

In a wireless network, poor signal reception is frequently experienced due to the obstruction of different objects such as trees, buildings, etc. The different object has a significant effect on path loss and received signal strength. However, if the environmental condition is known, the power of the signal may include the actual power plus the signal loss. In such a case, the receiver must be aware of the information, which requires an extra field in the transmitted packet and results in an overhead per packet.

2.7.2.3 GPS Based Mobility Prediction

The GPS-based mobility prediction approach uses geographical locational information to determine users' mobility patterns [366].

To elect a cluster-head, the algorithm proposed by Wang *et al.* [367] used GPS to predict node mobility and location information. The mobility prediction algorithm proposed by Su *et al.* [349] is based on location and mobility information provided by the GPS system. The algorithm uses the Network Time Protocol [368] or GPS clock to synchronize the nodes to avoid inconsistent data traces. The duration of a node at a place was determined by using GPS system data like speed, direction, radio propagation range etc.

To predict the mobility of the node, a novel routing protocol called Zone-Based Stable Routing (ZBSR) was proposed in [369]. This prediction algorithm uses the traces collected by GPS. The area is divided into the non-overlapping square zone. Every zone has a zone-head that does the job of a router in the network and also keeps the information about the other nodes in that zone. The node ID is used to decide the path to every zone.

GPS may not work in particular environments (like indoor places, places lacking a

strong cellular network, etc.) and consumes considerable battery power; hence, dependence on GPS is undesirable in MCC.

2.7.3 Mobility Tracking

The above-mentioned mobility prediction methods have several limitations, as mentioned in [370]. Researchers tried different means to track human mobility by analysing their accompanying devices. For example, to analyse human mobility, Smoreda *et al.* [371] discussed the data collection methods from mobile phones. In [372], users' mobility is analysed and characterised by the data collected from smartphones and smartwatches. Williams *et al.* [373] measured human mobility based on the user's mobile phone records and GIS data. Wang and Ak Yildiz [374] predicted user mobility with respect to a set of mobile switching cells based on the aggregated history of the mobile users and system parameters. To determine the user's current location, Ma, Fang, and Lin [375] considered the user's movement and the current system time.

2.7.4 Mobility and Stability Prediction in Mobile Computing Systems

Shah *et al.* [254] proposed to use the history of users' mobility patterns in an ad-hoc mobile grid to select a resource that would probably remain connected for a longer period. Zhou *et al.* [156] proposed a device selection method based on the stability resource status of the devices in a mobile device cloud. In the proposed model, the cloud controller maintains the historical information of each participating device. Based on these information, each device's stability is assessed, which helps in selecting a suitable device as a resource provider. Haryanti and Sari [272] predicted the mobility of a group of mobile nodes to identify a cluster of nodes among the available resources-providing nodes. Each node of that cluster has corresponding mobility with respect to the resource-requesting node. The idea was to identify a stable cluster of nodes, which would ensure the completion of the assigned tasks. Farooq and Khalil [121] proposed a mobility model where the previous records of the contact duration of two devices were used to predict the duration a resource-providing node may remain in the vicinity of the resource-requesting node in a mobile grid. Based on the predicted time, the task assignment decision is taken.

2.7.5 Mobility-aware Service Discovery and Delivery

Deng *et al*. [376] proposed a mobile service provisioning architecture for sharing services among the mobile device user community. Tyagi, Som and Rana [377] proposed a reliability-aware data delivery protocol for MANET (mobile ad-hoc network) based on AODV (ad-hoc on-demand distance vector), considering the speed of intermediate nodes in the route. Vadde and Syrotiuk [378] studied the impact of various factors such as QoS architecture, routing protocol, medium access control protocol, offered load, and mobility and their interactions on service delivery, based on different measures like real-time throughput, total throughput, and average delay. Various service discovery protocols are discussed in [379], while a decentralized service discovery mechanism is presented in [380]. Chang, Srirama, and Ling [381] proposed a mobile device-hosted service-oriented workflow-based mediation framework for mobile social network in proximity (MSNP). The proposed framework, named as adaptive mediation framework for service-oriented MSNP (AMSNP), is based on a public mobile P2P network in which mobile users can interact with the neighbouring mobile devices.

2.7.6 Research Scope

As we saw in the previous sections there are several works that addressed the mobility issue in mobile and ad-hoc systems. However, for P2P resource-provisioning in an infrastructure-less dynamic environment like MCC, mobility is still an issue. For P2P resource-provisioning, instead of assessing absolute mobility of the nodes it is sufficient to assess relative mobility or stability between them which can accommodate the change in absolute mobility and still allow to continue exchange services. To the best of our knowledge, there is no such work available in the literature. The existing mobility prediction algorithms proposed in the literature are good for different applications domain. But they fail to predict the relative stability of a node with respect to its peers, which is the key to selecting a service providing SMD in PMCC.

2.8 MCC as Edge Computing

Edge computing is getting popularised as the complementary of cloud computing

mainly because of its latency advantages [30] [382] [383]. In the following, we report some of prominent research works in which the computing competencies of the mobile devices are leveraged, especially in some form of edge computing. We also report the research works, though not many found, that mentions use of edge computing in smart buildings.

2.8.1 Mobile Devices Edge Computing

In several works, the researchers proposed or experimented with using mobile devices to form local fog/edge computing [384] [385] [386]. Hirsch *et al.* [387] proposed a resource management scheme for a dew computing model comprising a mobile cluster made of citizens' mobile devices for some participatory sensing applications. The aim of this system is to offer a distributed computing environment for processing or preprocessing the locally generated sensor data in real-time from a smart city perspective. In the extended work [388] of this paper, the authors demonstrated the usefulness of such smartphone-comprised dew computing for achieving compute-intensive edge jobs such as real-time stream processing using Tensorflow object recognition models. In their working paper, Kündig *et al.* [146] presented a theoretical model of a crowdsourced edge computing architecture along with highlighting the rudimentary challenges of this computing system. In their prototype use case, they distributed some object detection tasks to crowds' mobile devices on a university campus. Zhang *et al.* [389] suggested utilizing the idle GPUs of the peer gamers and other nearby private/professional GPU owners to form an edge network for high-performance online video gaming. Xing *et al.* [390] attempted to minimise the computation latency in a local edge computing comprised of wireless devices such as smart wearable devices, cell phones, tablets, and laptops. The computations are offloaded from the user device to the mobile edge in a P2P fashion. Pan *et al.* [391] also envisioned optimally offloading the tasks to other mobile devices aiming at minimizing the energy expense and maximizing the throughput in a crowdsourced mobile edge computing framework where mobile devices at the edge share their heterogeneous resources with each other.

2.8.2 Edge Computing for Smart Buildings

Several research works recognised the need for and demonstrated the use of edge

computing for smart buildings [392] [393]. Vilalta *et al.* [394] presented a fog architecture named TelcoFog, as an edge computing service for telecom operators. The proof of concept was validated in an HVAC environment. Raspberry Pi has been popularly used in home automation [395] [396] and controlling HVAC systems [397] [398] [399].

2.8.3 Research Scope

As discussed in Section 2.8.1, establishing mobile and wireless device enabled edge computing has garnered the researcher's attention. However, the idea is still at a very nascent stage and needs extensive study for successful implementation. Also, to the best of our knowledge, no work has been attempted yet to conceptualise and implement edge computing explicitly using crowdsourced SMDs for real-time processing. In this direction, our work, presented in Chapter 9, for implementing edge computing for a use case of smart HVAC utilising the SMDs available in the vicinity of the building is certainly the first of its kind.

2.9 Summary

In this chapter, we reported the research works that are related to MCC and the problems we addressed in this thesis. There are several aspects such as distributed nature, resource mobility, non-dedicated resources, energy constrained resources, dynamically changeable resource parameters, etc. make realising MCC challenging. In this thesis, we covered some selected aspects of MCC such as generalised architecture of MCC, resource profiling and selection, task scheduling, resource availability assessment, mobility handling. And to establish the feasibility of MCC we also presented a proof-of-concept with a use case. Being a distribute system, MCC naturally inherits several issues from it. Therefore, it is expected that many of the issues' solutions also can be found from the existing literature on distributed systems and computing. However, in this chapter we saw that not all the issues have existing solutions that can readily be applied on MCC. This is because though MCC has similarities with other known systems it has its own uniqueness on several grounds, as discussed in Chapter 3. Therefore, we needed to think for and come up with novel solutions that would most suitably be applicable to the context of MCC presented in this thesis.

3

MCC: Concept, Architecture and Research Challenges

"We don't have to engage in grand, heroic actions to participate in change. Small acts, when multiplied by millions of people, can transform the world."
--- Howard Zinn

3.1 Introduction

The innovation of computers has changed many aspects of mankind. Computers have not only led to solutions to many existing problems but also to many other unforeseen and innovative problems that have influenced and eased our life greatly. Recognising the problem-solving ability, computers are asked for various purposes, and accordingly, the demand for more powerful computers has grown steadily.

To meet the high computing demands of organisations, mainframes came into existence in the 1960's. At the same time, for number crunching operations, supercomputers were developed that could produce unparalleled performance [400]. With the popularity of microcomputers, the distributed system was the next revolution in HPC in the 1970's. In the 1990's, Ian Foster and Carl Kesselman proposed a new computing paradigm named grid computing as an economical alternative to the costly supercomputers in which idle computing resources, within the organisation or distributed over the globe, are shared over the network [401]. In early 2000, with the emergence of cloud computing, making forward the vision of grid computing further, the HPC indeed became utility computing [402] [403] [404].

With the introduction of PDAs by Apple in 1993, the computing horizon changed drastically. It actually initiated the era of mobile and ubiquitous computing. Since then, mobile devices have evolved enormously. Over the last couple of years, the SMD industry has seen an unprecedented focus on hardware. The processing capability of SMDs to meet various purposes has increased exceptionally, be it CPUs or GPUs or even DSPs. The CPU and memory architectures are designed and tuned

to boost heterogeneous computing. The GPUs are also engineered to enhance GPGPU computing performance.

Today's SMDs such as smartphones, phablets, and tablets have become computationally so powerful that they can easily hands-on defeat yesteryear's powerful supercomputers. For example, NVIDIA's Tegra X1, released in 2015, became the first-ever mobile SoC to reach the Tera-FLPOS mark. It is interesting to mention that Deep Blue, one of the most powerful computers of that time, which defeated Gary Kasparov in a much-hyped chess rundown in 1997, displayed a performance figure of a meagre 11.38 GFLOPS. NVIDIA claims that Tegra X1 is more potent than the ASCI Red, the first Tera-FLPOS supercomputer of 20 years back, employed for ten years by the Sandia National Laboratory of Department of Energy, United States. This significant escalation of the modern-day's SMD's capability gives us the confidence to consider them a viable option for carrying out computation-intensive tasks.

Alongside, thanks to their capability of hosting and supporting various services and applications, SMDs have become all-purpose and indispensable personal devices in our daily lives. Consequently, the number of worldwide SMD users has seen a steep ascent in recent years. Leaving behind desktops, laptops, and notebooks, SMDs have become the primary computing device for most users [405]. Global Stats, the research arm of the web analytics firm StatCounter found that for internet uses in October 2016, for the first time, the number of worldwide SMDs users (51.3%) exceeded the number of desktop users (48.7%) [406]. As per their June 2022 report[17], the market share for SMDs and desktops are 62% and 38%, respectively.

However, like desktop users, SMD users also use their devices for a fraction of the time in a day. Therefore, a huge computing potential remains unutilised. In MCC, we envisage tapping and exploiting these idle computing resources of the SMDs of the mass user base, very much like grid computing. We advocate utilizing the collective processing powers of these mighty SMDs to achieve HPC. A substantial

[17] https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet

virtual computing capacity can be garnered utilising the SMD globally (discussed in Section 3.3.5.1), and MCC can also be employed locally to cater to the needs of real-time applications. The emergence of big data, IoT, and AI based applications has escalated the demand for pervasive and ubiquitous HPC significantly [30]. A local MCC (discussed in Section 3.3.5.2.1) or an ad-hoc MCC (discussed in Section 3.3.5.2.2) can be feasible solutions.

Although the idea of computing on mobile devices is not new, the significant augmentation of today's mobile devices' competency has prompted many researchers to explore utilising their computing capabilities in different ways. Researchers proposed to incorporate mobile devices into grid computing to leverage their computing resources [125] [121] [126]. Few works aimed to utilise the accumulated computing power of a cluster of smartphones [93] [96]. To minimise communication latency, local clouds have been made utilising neighbouring mobile devices [40] [135] [128]. Mobile devices are proposed to use for edge and dew computing as well [140] [76] [387] [388]. Also, utilising volunteered mobile devices to attain MCC has been introduced in [141] [142] [143]. However, we could not find any paper that exclusively presented the concept and associated aspects of MCC. Seeing the lack of comprehensive theoretical discussion in the literature, we felt the need to submit one.

This chapter presents an in-depth study on MCC, attempting to cover every nitty gritty of this computing paradigm. In the following sections, we explore the feasibility and the promises of this very idea, also mentioning related research works and projects. We also discuss the obstacles that can be faced to realize this concept and pledging progress toward their solutions. We suggest some exciting application areas of MCC.

In particular, in this chapter, we aim to achieve the followings:

- Extending and re-presenting the concept of MCC.
- Establishing the competence of the modern SMDs as computing devices and validating achieving HPC utilising them by recognising and associating the favourable external factors.
- Discussing the idea of crowd and opportunistic computing.

- Rendering a taxonomy of grid computing that leads to MCC.
- Stating a concrete definition of MCC and identifying its defining properties.
- Presenting exhaustive comparisons between MCC and other HPC systems such as desktop grid computing, cloud computing, cluster computing and supercomputers, and between MCC and other mobile computing systems such as mobile grid computing and mobile cloud computing, ad-hoc mobile cloud, and mobile crowdsourcing.
- Laying out different probable general architectures of MCC, including the working, major components, advantages, and disadvantages of each architecture.
- Identifying different types of MCC depending on the infrastructure and applications.
- Meticulously analysing the essential requirements and considerations for MCC system design, development, deployment, and operations.
- Discussing the advantages of MCC with specific discussions on pervasiveness and sustainability of MCC.
- Elaborately discussing the issues and challenges of MCC while mentioning its limitations.
- Identifying and recommending the practical applications of MCC in various scenarios.

3.2 Enabling Backdrops for Realising MCC

In this section, we analyse some other favourable factors that have directly or indirectly boosted and accelerated the realisation of MCC.

3.2.1 Competence of Contemporary SMDs as Computing Resources

Over the last few years, the SMD industry has seen an unprecedented focus on its hardware. The processing capability of SMDs to meet various purposes has increased exceptionally, be it CPUs or GPUs or even DSPs. The CPU and memory architectures are designed and tuned to boost heterogeneous computing. The GPUs are also engineered to enhance GPGPU computing performance. SMD manufacturers are putting continuous effort into boosting their products to have computer-like capabilities. They are even going further through their attempt to

bundle multiple SoCs on a single unit [407], which will undoubtedly revolutionise the ability of SMDs as computing devices. These devices are further augmented by excellent battery life, storage capacity, networking, and several powerful and effective sensors on a single platform. In this section, we discuss the decisive properties of modern SMDs that make them competent computing resources.

3.2.1.1 Advancements in Mobile CPU

The CPU architecture for mobile devices has made unprecedented advancements in recent years. In the following, first, we mention the key technicalities in this regard. Then we check out some examples of a few latest SMDs' CPU specifications to argue about their computing capabilities.

3.2.1.1.1 Symmetrical Multi-Processing

SMP technology applies to a multi-core shared memory architecture where each identical core is capable of operating independently but maintaining healthy co-operation. They tend to share workloads whenever possible, lessening the burden on a particular core. This allows the cores to run at a lower frequency, resulting in less power consumption, which is crucial to mobile systems [408]. So, SMP empowers SMD processors to not only produce greater performance but also tackle peak performance demands (thanks to sharing workloads) while limiting the power appetite of SMDs reasonably.

3.2.1.1.2 Heterogeneous Multiprocessing

Modern ARM-based SMD's multi-core CPUs comprise two different sets of cores paired together into a single unit. One set of cores is more powerful than the other. The powerful cores are for high performance, whereas the other set is for better power efficiency. The decision to submit jobs to the appropriate core is taken dynamically by mapping to the varying computational demand of the application. This scheme is generally known as HMP, which is fashionably termed big.LITTLE™ by the ARM. When tasks are run on the 'LITTLE' cores, they use less power. Hence, they drain the battery less; however, they may run slightly slower. When tasks run on the 'big' cores, they finish sooner, but they eat more battery. Typically, background tasks featuring in-order execution are employed to the energy-efficient 'LITTLE' cores, whereas the user-interactive tasks featuring out-of-order execution

are operated by power-incinerating 'big' cores, which require being in action typically for shorter periods. The aim is to use 'LITTLE' cores most of the time and use the 'big' cores only for high-frequency operations. Furthermore, when 'big' cores are not in use, they are powered off. The whole concept can save up to fifty percent of the energy for a standard mobile workload [409]. The prevalent achievement of the HMP is that of significant average power savings without compromising the peak performance, which is an essential gain for SMD computing.

3.2.1.1.3 Powerful and energy-efficient CPUs

The first multi-core smartphone, announced at the end of 2010, was the LG Optimus 2X, loaded with the Tegra 2 processor from NVIDIA, with a maximum frequency of up to 1.2 GHz [410] [408]. Since then, all SMD manufacturers have been in a war to load their products with an increasing number of cores. To buttress this exigence, the chip makers are regularly coming out with more powerful mobile processors. Modern SMD CPUs typically consist of two to eight highly efficient cores. It is very common to find SMDs with a CPU clock speed of more than 2.5 GHz. As per the recent launch, the prime cores of Qualcomm's Snapdragon 870 and 8+ Gen 1 run at 3.2 GHz, while Kirin 9000 has 3.13 GHz cores. Moreover, taking along the HMP concept, the CPUs are adequately balanced for computing power and energy efficiency by an optimal combination of high-power and high energy-consuming cores with low-power and energy-saving cores. For example, the Snapdragon 8 Gen 1 comprises one Cortex-X2 core of 3.0 GHz, three Cortex-A710 cores of 2.50 GHz, and four Cortex-A510 cores of 1.80 GHz [411]. Depending on the processing requirements, different cores are employed, minimising unnecessary energy wastage.

3.2.1.2 GPU-Accelerated Computing

Bestowed with number-crunching power, GPUs are the key potency that makes today's SMDs powerful computing devices. The CPU cores are generally optimized for sequential serial processing, whereas a typical GPU consisting of many smaller and more efficient cores, often known as shader cores, is designed for massively parallel processing [412]. The inherent parallelism property of a GPU enables it to execute thousands of parallel threads, handle multiple tasks simultaneously, and

solve large problems real fast. Several benchmarks have established the superiority of GPUs over CPUs in terms of raw processing [413].

In GPU-accelerated computing, the compute-intensive segment of the application is passed on to the GPU; however, the rest of the code is executed on the CPU. GPU-accelerated computing can deliver exceptional performance if the right process is scheduled for the right core. Serial portions of the executable codes are submitted to CPU cores which are optimized for low latency on a single thread, while parallel portions of code are sent to GPU's mass cores which are optimized to maximize accumulated throughput. This tactic confers enhanced performance per unit area than either CPU or GPU cores can alone [414]. In general, accelerated computing is revolutionizing HPC these days in the way that systems with special accelerators offer highly energy-efficient computing delivering the utmost performance for HPC. So, GPU-accelerated computing plays an important role in SMD computing not only by executing heavy processes much faster than CPUs but also by consuming less energy per computation than CPUs.

With the increasing popularity of virtual reality based gaming apps, SMDs are getting loaded with considerably powerful GPUs. For example, the latest offer from Qualcomm, the Adreno 730[18] (used in Snapdragon 8/8+ Gen 1), comprises 768 shading units, with a base and boost clock speeds of 812 MHz and 970 MHz, respectively, producing a hopping 1.8 TFLOPs for single precision (32) floating point operations, which is a very good offering for computing-intensive parallel tasks.

3.2.1.3 SoC Technology

At the heart of every SMD, there is a module known as a system-on-a-chip (SoC). The SoC of an SMD is like the motherboard in a desktop computer. It incorporates various chips and components that make up an entire electronic setup on a single chip. Among the components are the CPU cores, GPU, multimedia processor, signal processors (DSP and ISP), security processor, different types of memories with a memory controller, wireless radios (Wi-Fi, 3/4/5G, etc.), power management

[18] https://gadgetversus.com/graphics-card/qualcomm-adreno-730-specs/

circuits, timers, interfaces, OS, and other utility software. Day by day, the SoCs are becoming more energy-efficient and smaller despite being more powerful. The advancement in fabrication technology allowed the manufacturers to pack more power in the same chip volume. For instance, the Snapdragon 870, 888, and 8 Gen 1 are of 7 nm, 5 nm, and 4 nm, respectively. Apparently, SoC has made it materialize to put a whole computer on a chip and reduced the size to a thumbnail, which is very crucial to SMDs. Some of the obvious benefits of SoC, from the perspective of SMD computing, are listed below.

- Higher performance is achieved by embedding heavy computational functions and logic in a large number of highly integrated circuits [415].
- Smaller footprint and space requirements of SoCs have allowed SMD makers to place larger batteries for longer power backup.
- SoCs do not demand much power, thanks to the very high level of integration and considerably shorter wiring, which is a big boon regarding SMD computing.
- Producing a single chip is far more cost-effective than traditional multi-chip motherboard-based computers [416]. It has been observed that mobile SoCs are roughly 70 times cheaper than other HPC systems like PC clusters [417].
- An integrated environment offers greater system reliability.

3.2.1.4 Sufficient Memory

In any distributed computing, dividing a large job into a parallelly runnable number of small jobs and distributing them to the different nodes for processing is an overhead. The grain size of the sliced job should be at par with the primary memory available in the computing node for smooth execution. Though SMDs have a lesser memory than desktops and laptops, mainly due to the size factor, lately, SMDs are getting loaded with an abundance of memory (RAM, ROM, and cache) in conjunction with increased internal data transfer speed. SMDs with 8 GB RAM and 128 GB ROM are very commonplace. Several high-end phones (e.g., Nothing Phone (1)[19],

[19] https://in.nothing.tech/pages/phone-1

OnePlus 8T[20], OnePlus 10 Pro[21], Sony Xperia Pro-I[22], Red Magic 7[23], and many more) offer LPDDR5 RAMs of 12 GB and more with ROMs of 256 GB and higher. And most of these phones allow to expand the inbuilt memory with external flash memories to 1 TB and more (e.g., Samsung Galaxy S10[24] series phones). This abundance of internal and external memories encourages carrying out data-intensive tasks besides compute-intensive ones.

3.2.2 SMD Market and User Development

For the last few years, if a single device has to be named, which has impacted the human lifestyle and business market, most are undoubtedly the smartphone. Thanks to their all-round capabilities and utilities, the worldwide adoption of SMDs has exponentially increased in recent years. In 2011, vendors shipped more smartphones than PCs for the first time in history [418]. A report from Cisco shows that the number of mobile devices and connections raised to 7.9 billion globally in 2015 from 7.3 billion in 2014 (smartphones 32%, phablets 6% and tablets 2%) [419]. Statista[25], a leading market and consumer data provider, estimated that by 2027, the number of smartphone users would cross 7 billion marks, as shown in Fig. 3.1. Fig. 3.2 shows the forecasted number of smartphone connections (in millions) in the top ten countries by 2025, as estimated by Statista. As per GSMA[26], a leading global mobile market analysis organisation, it is expected that globally there will be 2.5 billion unique mobile subscribers by 2025 [420].

Besides the justifiable reasons such as reduced usage friction via excelled hardware, improved user interface, ease of use, and expanded and multidimensional services, the explode in SMD sales is a result of the aggressive penetration of low-cost SMD makers in emerging markets [418], in conjunction with affordable 4G data plans [421]. The cost of SMDs is getting lower following the common business-economic rule that a higher volume of SMD market helps in reducing design, production,

[20] https://www.oneplus.in/8t
[21] https://www.oneplus.in/10-pro
[22] https://www.sony-asia.com/electronics/smartphones/xperia-pro-i
[23] https://www.nubiamart.com/nubia-red-magic-7.html
[24] https://www.samsung.com/us/app/mobile/galaxy-s10/
[25] https://www.statista.com/
[26] https://www.gsma.com/

and marketing cost, which also leads to faster product evolution with superior performance on a low budget. So, not only the technological demand but also the economic senses decisively motivate companies to produce SMDs with HPC features [417].

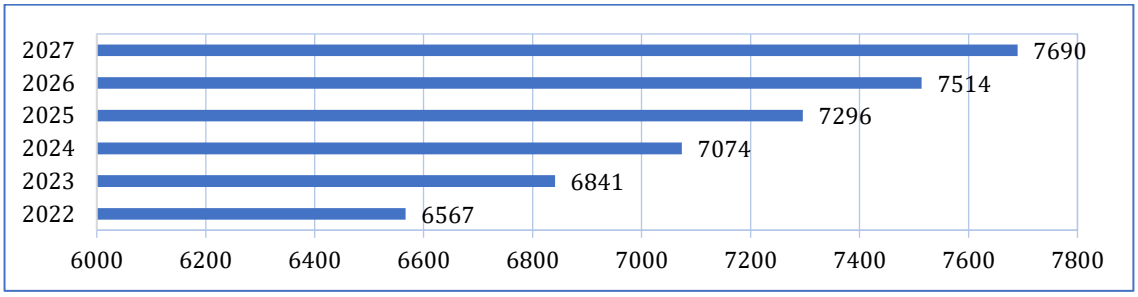


Fig. 3.1. Estimated number of worldwide smartphone subscriptions (in millions) from 2022 to 2027 [422]

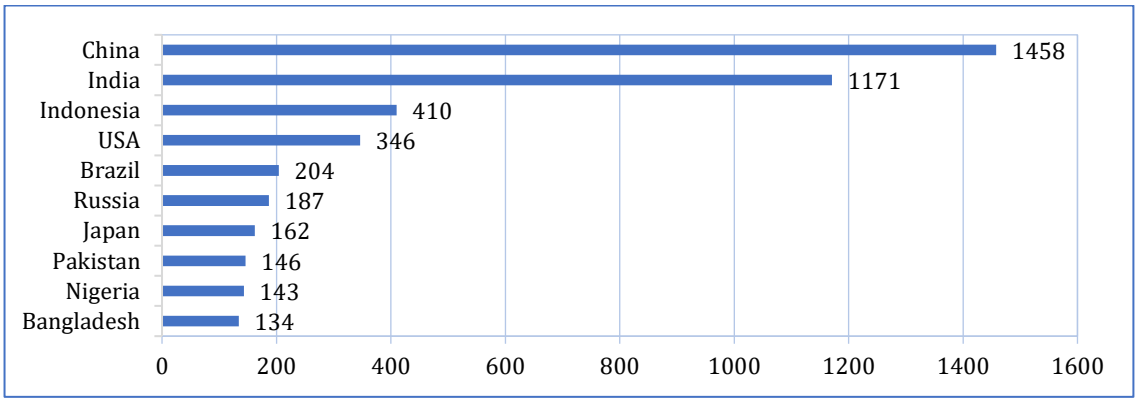


Fig. 3.2. Estimated number of smartphone connections (in millions) of top ten countries by 2025 [423]

There is a number of factors influencing the growth of SMD use in the global market as mentioned below [424]:

- Rapidly falling price of SMDs has accelerated the customers to move from basic and standard feature phones to smartphones.
- Developing SMD technologies have a reason for the increase in the sale of low-end SMDs.
- The increasing availability of the highspeed 4G/5G spectrum with increased mobile broadband connections all around the world.
- The availability of highspeed 'data-centric' services and low tariffs has increased the adoption of SMDs in both developed and developing societies. Further, the availability of cheap data tariffs tailored as per the customers' needs has also reasoned for smartphone adoption in developing countries.

- The concept of tailoring the data tariffs for cost-conscious prepaid consumers can be linked with the selling rate of SMDs.
- Efficient retail channels and supply chain have helped manufacturers to reach customers from every corner across the globe.
- The government policies in support of the growth of the SMDs and subsidiary industries have a significant role in price slicing and the growth of mobile networks.

3.2.3 Increasing Wi-Fi Zones

As the number of mobile devices and connections proliferates, more and more places are coming under a Wi-Fi zone due to the dense installation of Wi-Fi hotspots. These hotspots may be commercial or public, including homespots and community hotspots. Public hotspots can significantly influence the realization of MCC. Community hotspots use dual SSIDs that allow nonsubscribers to access Wi-Fi service as guests. Commercial hotspots are also being offered to the public in places like shops and malls, cafés and restaurants, hotels, railway stations and airports, public transport, etc. A study from Cisco shows that, globally, total public W-Fi hotspots will grow to 432.5 million by 2020 from 64.2 million in 2015, i.e., a seven times upsurge, while commercial hotspots are estimated at 9.3 million by 2020 [419]. The advantage of a greater number of hotspots is the reduced cell size with strong signal coverage, which results in robust crowd computing systems with low latency in message passing.

3.2.4 Low-cost and Highspeed Mobile Data

The advancements in mobile network technology have made mobile computing more practical and affordable. The mobile data bandwidth jumped a big leap with the launch of 4G/4G LTE, rising to 100 MHz from 25 MHz of 3G. This reached a new high with a massive upscale to 30-300 GHz with 5G networks thanks to the technologies like EMBB, URLLC, and MMTC. 5G offers speed in the range of 100 Mbps (low-band) – 20 Gbps (high-band). With these advancements, the cost per unit data transfer rate is significantly lowered. Furthermore, an innovative idea like 5Gi (5G radio interface technology) promises to offer more range at a lower frequency. 5Gi is a joint initiative by IIT Hyderabad, IIT Madras and other premier

academic institutions in India, along with the Centre of Excellence in Wireless Technology. The project is backed by the Department of Telecommunications, Govt. of India. Its potential for large-scale deployment with the enhanced coverage in remote areas, rural regions and difficult terrains makes 5Gi much more economical in the countries like India.

GSMA estimated that two-thirds of the world's mobile connections will be running on 4G and 5G networks by 2025 [425], while their latest estimation states that there will be 2 billion 5G connections globally by 2025 [420]. While the full potential of 5G is yet to realise as it is still in its early days, people have already started working on 6G aiming for more speed and increased bandwidth. Organisations like ITU-R 6G Vision Group [426], Orange [427], North America's Next G Alliance [428], The University of Texas (6G@UT[27]) [429], Oppo [430], MIT-Ericsson [431], to name a few, have kicked off planning, researching, and working towards 6G. Considering this, we expect to witness more low-priced mobile data with massively higher speed. This would boost the vision of MCC substantially.

3.2.5 Highspeed and Energy-efficient Short-range Communication

Modern SMDs are equipped with energy-efficient, high-speed, short-range communication technologies such as BLE [432], NFC, Wi-Fi Direct, etc. These technologies allow forming ad-hoc or P2P MCC in the absence of WLAN and cellular internet. Besides low power consumption and high-speed data transfer, BLE offers several advantages over classic Bluetooth, such as cost minimization, robust transmission with minimised interference, extended connection range, ease of use and integration [433]. Although NFC requires three times less energy than BLE, the very short-range coverage makes it a little impractical for MCC. Compared to BLE and NFC, Wi-Fi Direct offers the highest data transfer speed with greater coverage but compromises energy efficiency. A comparative statistic of these three technologies is given in Table 3.1.

[27] http://6g-ut.org/

Table 3.1. Comparing different short-range communication technologies for MCC

Parameters	BLE	NFC	Wi-Fi Direct
Frequency	2.4 GHz	13.56 MHz	2.4 GHz
Data rate	Up to 2 Mbps	Up to 424 Kbps	Up to 250 Mbps
Range	< 100 m	< 4 cm	< 200 m
Power consumption	0.01–0.50 W	0.025 W (approx.)	< 20 W
Energy efficiency	High (15 mA)	Very high (5 mA)	Low
Connection stability	High	High	Medium

3.2.6 HPC Through MCC

From the above discussions, we are convinced that the capabilities of the modern SMDs are sufficient to consider them as individual computing units. However, normal SMD users do not utilise the full potential of their device's capabilities. In fact, the majority of SMDs are not being used to their capacity. Studies suggest that normal users interact with their SMDs only for a few hours (on average, two to four) in a day [62] [63]. So, a huge amount of processing capability remains unused and wasted. The market buzz of 8/10 core processors makes the scenario more interesting. Often 2-core processor is sufficient for a regular user unless he operates heavy applications like 3D games. So even when SMDs are in use, it is highly probable that some of the CPU cores and the GPUs, along with DSPs, ISPs, etc., remain free. An enormous processing capability can be generated if these unused processing powers are tapped and exploited properly (opportunistically). By accumulating these unused resources on a large scale, we can achieve virtual HPC in the same way as desktop grid computing. For instance, if 100 or more SMDs with GFLOPS higher than 500 (e.g., Imagination PowerVR GT7900 or NVIDIA Tegra) are connected together, the accumulated GFLOPS can challenge some of the mighty supercomputers in the world[28].

3.3 Rudiments of MCC

In this section, we formally and elaborately introduce the concept of MCC along with its architectures and types while comparing it with other similar computing systems.

[28] https://top500.org/

3.3.1 Definition and General Properties of MCC

A *crowd* is a definite collection or group of people with shared purpose and emotions [434]. The term *crowd computing* has been used by many people to refer to the involvement of humans in the computation process. No doubt, computers have evolved into very powerful and 'intelligent' machines. However, there are still certain kinds of tasks that humans can do far better and more accurately than computers owing to their intrinsic cognitive abilities and natural instinct. Crowd computing aspires to fusion human intelligence and computer algorithms to make computers more knowledgeable and intelligent. Human capabilities are tapped to solve the computational problems in crowd computing applications (e.g., Wikipedia, Yahoo! Answer, etc.), which are otherwise difficult to accomplish by computers only.

However, we intend to envisage MCC as only a distributed computing platform. Our perception of crowd computing resonates with that presented by Murray *et al.* [107] and Fernando *et al.* [142]. We define MCC as the following:

It is a distributed computing approach where public-owned SMDs are opportunistically utilised in a resource-scavenging fashion for executing computing-intensive tasks.

According to our proposed idea of MCC, it can generally be characterised by the following particulars:

- It is a distributed computing system.
- Every MCC application has a predetermined and defined purpose on which the corresponding inputs, task properties, and outputs depend.
- The task generator creates distributable tasks dispensed to one or more computing devices (task executors).
- The task generator/distributor and the task executor are different devices.
- The task generator and the task executor are connected through a local or global network.
- The computing devices (crowdworkers/task executors) are not owned by any organisation or any single entity; rather, they are owned by the general public

as their personal devices.

- The crowdworkers provide their SMD resources voluntarily.
- The crowdworkers might demand or receive some incentives for lending their SMDs.
- Each crowdworker is treated as an individual computing node.
- Each crowdworker is typically mobile in nature; however, it can have different mobility statuses, as shown in Fig. 3.3.
- The assigned tasks are executed on the executing device in an opportunistic or CPU-stealing fashion.
- Generally, the tasks are independently executable, but depending on the applications, there might be dependency and different workflows.
- The assigned tasks are supposed to be entirely executed by the crowdworkers and return the results before leaving the network.
- The task generator collects the results from the crowdworkers and assembles them correctly to get the final result.

Static	•The device is always static
Mobile	•The device is generally mobile
Pseudostatic	•Mobile but relatively static with another device as they move together
Temporarily static	•Mobile but takes time to relocate from one position to another
Always mobile	•Continuously mobile

Fig. 3.3. Different mobility states of the resource provider and consumer

The common terms used in this chapter in discussing MCC are described in Fig. 3.4.

3.3.2 Comparing MCC with Other HPC Systems

We project MCC as an alternative and sustainable HPC system. This section examines how MCC differs from other popularly known HPC systems. Table 3.2 summarises the comparison between MCC and other HPC systems that are discussed below.

SMD	•Smart mobile devices such as smartphones, phablets, and tablets.
Crowd	•Gathering of people, but could be generalized as gathering of entities for resolving an issue by collective decisive power of individuals.
Crowd computing	•Computing performed by a number of computing devices in distributed manner for solving a common computational problem.
Crowdworker	•Agents involved in crowd to do functional job like computation, decision making, etc.
MCC	•Collection of SMDs connected through local or global communication networks in a centralised or P2P fashion to provide aggregated computing power.
MCC application	•A project wih high computation demand that is catered by MCC.
MCC task	•A task that is originated from the MCC application. It is large and can be many for a particular application.
Microtask	•An MCC task is devided into severall smaller subtasks that are sent to a crowdworker for processing with definite input and output specifications.
Result	•The output of the executed task on the crowdworker, and is to be returned to the coordinator.
MCC coordinator	•It is the software component of MCC that is responsible for most of the operations starting from job creation to result aggregation and validation.
Middleware	•It is the main part of the coordinator that communicates with the crowdworkers and performs related operations such as task dispatching and result collection.
MCC server	•Resourceful computing device that hosts the coordinator and runs the middleware.
MCC client	•Part of the MCC application, installed on the crowdworker and is responsible executing the assigned task on the device opportunistically and return result to the coordinator.

Fig. 3.4. Common terms used in discussing MCC

Table 3.2. Comparing MCC with HPC systems

Comparing parameters		MCC	Grid computing	Cloud computing	Cluster computing	Super-computing
System set up	**Resource anthology**	Highly scattered	Scattered	Integrated	Highly integrated	Unified
	Nature of resources	Dynamic	Mostly fixed	Fixed	Fixed	Fixed
	System ownership	Small/medium/large organizations	medium/ large organizations, educational and research institutions	Large companies	Large organizations and research institutions	Large educational and research institutions
	Resource ownership	Individuals	Organizations, institutes and individuals	Service providing company	Organizations and research institutions	Educational and research institutions

	Dedicated system	No; resources are acquired opportunistically	Many often, dedicated, but mostly opportunistic	Dedicated, but not for a particular service consumer	Yes	Yes
Architecture	**Processing topology**	Distributed	Distributed	Centralised/ distributed	Distributed	Centralised
	Computing node counts	Large	Large	Very large	Many	Unified (of multiple cores/CPUs)
	Stateful/ stateless	Stateless	Can be both	Both	NA	NA
	Task grain size	Smaller	Medium/larger	Larger	Larger	Larger
	Resource coupling	Loose	Loose/ medium	Medium/ tight	Tight	Tight
	Resource access	Centralised/ decentralised	Centralised/ decentralised	Both	Centralised	Centralised
	Resource allocation	Decentralised	Decentralised	Both	Centralised	Centralised
	Resource handling	Decentralised	Decentralised	Both	Centralised	Centralised
	Batch processing or interactive	Batch processing	Batch processing	Both	Batch processing	Batch processing
	Centralised controlling	Yes, for centralised MCC; no, for P2P MCC	Yes; no for P2P grid	Yes	Yes	Yes
Desirable properties	**Mobility support**	Yes	Yes, for mobile grid	Yes	No	No
	Flexibility	Yes	Yes	Yes	No	No
	Elasticity	Medium	Medium/high	Unbounded	No	No
	Scalability	Scalable	Scalable	Highly scalable	Scalable	No
QoS properties	**Load balancing**	Moderate	Efficient	Very efficient	Very efficient	NA
	Fault tolerance	Less	Medium	High	High	High
	Processing latency	Medium	Low	Low	Negligible	No
	Performance	Good	Very good	Very good	Very good	Best
	Availability	Not guaranteed	Mostly guaranteed	Guaranteed	Guaranteed	Surely guaranteed
	Reliability	Less reliable	Mostly reliable	Highly reliable	Extremely reliable	Absolutely reliable
	Computing capacity	Low/high/ very high (depends on SMD availability)	High/ very high	Very high	Very high	Very high

Networking and data communication	**Network connectivity required**	Yes	Yes	Yes	Yes	No
	Distance between computing source and sink	One hop for local MCC, multiple for global	One, in case of the campus grid; otherwise, multiple hops	Multiple hops	One hop	One hop
	Data transfer latency	High for global MCC; low for local MCC	High for public grid; low for campus grid	High	Negligible	No
	Connectivity	WAN/WLAN/WiMAX/ hotspot/ Bluetooth/ NFC	WAN/LAN	Internet	LAN	NA
Servicing	**Service provided**	Computing	Mostly computing; also, data and storage	Various services	Computing	Computing
	Option for value added services	No	Yes	Certainly	No	No
	Multi-tenancy	No, but can be	Yes	Yes	No	No
	SLA	Less scope but required	Usually, SLAs are defined and followed	Defined and followed	NA	NA
	QoS	Not guaranteed	Protected	Guaranteed	Guaranteed	Absolutely guaranteed
Security, privacy and trust	**Security threat to host device**	Low for genuine MCC applications	Low	Low	Very low	No
	Security threat from host device	High	High	Very low	No	No
	Attack vulnerability while in transmission	Negligible for local MCC, high for global MCC	Negligible for campus grid; otherwise, high	Very high	No	No
	Privacy issue	High concern	Concern	Yes	No	Absolutely no
	Trust issue	Very high	High for public grid	Low for known service providers	Absolutely no	Absolutely no
Financial	**Upfront investment**	Negligible	Very high, for setting up new	Extremely high	Very high	Extremely high

aspects			organizational grid; negligible, for utilising existing computers in the organisation and also for public grid			
	Operational cost	Negligible for centralised MCC; otherwise, none	High for organizational grid; negligible for public grid	Extremely high	High	Very high
	Maintenance overhead	Negligible for centralised MCC; otherwise, none	High for organizational grid; negligible for public grid	Very high	High	High
	Bandwidth utilization cost	High	High	Very high	Very high	None
	Availing price	Zero or minimal	Zero or minimal	Moderate	NA	NA
	Business oriented	No, but there is potential	Generally, no; but can be	Absolutely	No	No
Environmental effects	**Energy consumption**	Very low	High	Very high	High	Very high
	Environmental hazard due to production	Less due to using existing devices	Less due to using existing devices	Very high	High	High
	E-waste	Less	High	Massive	Very high	Very high

3.3.2.1 MCC vs Grid Computing

The fundamental philosophy of grid computing is to voluntarily share idle computing resources collaboratively [435] [436]. Though grid computing has been used in different flavours such as data grid, knowledge grid, application grid, sensor grid, etc., as shown in Fig. 3.5, in this chapter, our reference to grid computing is limited only to the computational grid [401] [437]. It is obvious that MCC has been conceptualized from traditional desktop grid computing [438] [439]; hence, they share many similarities. But there are a few striking differences, as mentioned below, that make MCC distinct from the desktop grid.

- MCC has more opportunities than the desktop grid because the number of SMDs used worldwide is significantly higher than desktops.
- Due to their mobility, setting up an MCC is more flexible than desktop grids. Using a P2P MCC, an ad-hoc HPC can be created anywhere.
- In desktop grid computing, the internet is used to access distributed resources, whereas the internet is not necessarily required to build up MCC systems. The SMDs can be connected through WLAN technologies such as Wi-Fi, WiMAX, Bluetooth, mobile hotspot, etc.
- Ensuring the availability of SMDs and maintaining reliability is far more challenging in MCC.
- Architectural limits in SMDs force the client application to have fewer functionalities. The users do not have much to choose except either run or do not run the application. At max, they can schedule when their phones should be accessed. Conversely, resource providers in the desktop grid have the luxury of having more functional client applications. They enjoy more autonomy and can set multiple preferences for resource sharing.
- Due to the small memory size in SMDs, the distributed tasks should be much finer-grained than a desktop grid.
- Privacy and trust issues are more critical in MCC since SMDs are far more personalized devices than desktops.

Both systems have their fortes and limitations, so combining them should complement each other. We can have a much superior system comprising the flexibility, accessibility, and obtainability (a large number of SMD users) of MCC and robustness, reliability and security of the desktop grid.

3.3.2.2 MCC vs Cloud Computing

Cloud computing offers abstracted, centralised, and virtually unlimited computing-related services on demand to the consumers for a fee [440]. Cloud computing offers various online services such as storage, CPU, GPU, development platforms, software and endless applications [441] [442]. Although cloud computing provides several benefits such as Increased processing power, scalability, utility-based

pricing, and dynamic, flexible and agile service provisioning, a few crucial issues such as the requirement of continuous internet connection and significant latency limit the utility of cloud computing [30].

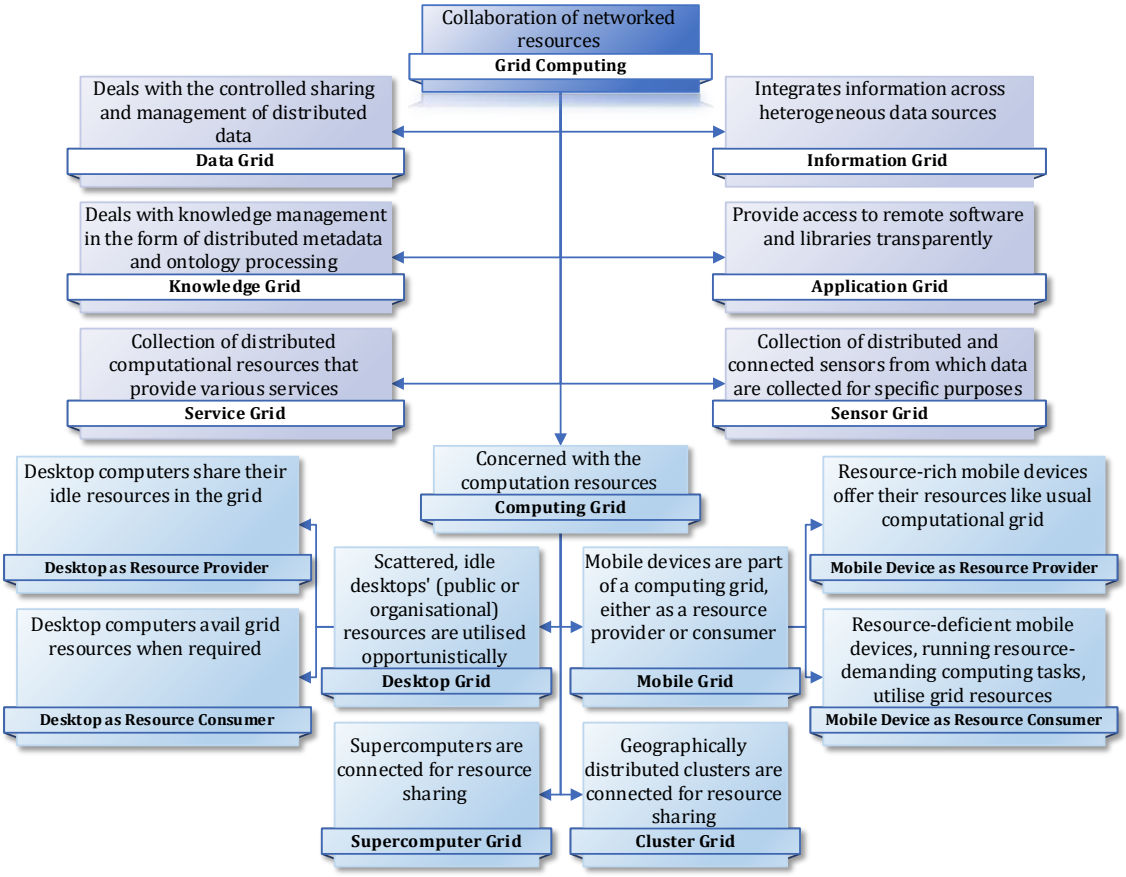


Fig. 3.5. A taxonomy of grid computing

Though MCC also aims to provide scalable HPC like there are several fundamental differences between the two, as mentioned below [34]:

- The service provisioning philosophy is exactly the opposite of the two approaches. In cloud computing, the public is the service consumer, whereas, in MCC, they are the service providers.
- Cloud resources are centralized, but in MCC, it is highly distributed.
- In MCC, ensuring crowdworkers' availability is a real issue, whereas availability of services anytime is guaranteed in commercial cloud computing.
- Computing services in MCC can be availed without or with minimal cost. But availing of cloud services always entails money, and the price depends on the type and duration of the service availed. Compared to MCC, cloud computing is considerably expensive.

- Not only computing services but cloud computing offers several other services, as mentioned above. But MCC is meant for computing services only.
- Establishing and operating cloud infrastructure is hugely expensive, whereas MCC has almost zero cost in this regard.
- Cloud infrastructure needs dedicated space with continuous cooling and power backup facility, which is not required in MCC.

3.3.2.3 MCC vs Cluster Computing

In cluster computing, several computers are tightly or loosely connected, generally through highspeed LANs [443]. The connected computers work as a unified entity, providing considerable HPC [444] [445]. Though MCC and cluster computing both intend to provide HPC by amassing multiple computing units, they differ by far in the following aspects:

- The most significant difference between the two is the resource type. In cluster computing, the PCs are used for computation. No mobile devices are connected as a resource provider or consumer. Whereas in MCC, only SMDs are the resource providers.
- Establishing a first-hand cluster computing setup is expensive. However, if the existing computers are utilized in an organisational cluster, this expense can be waived. In both cases, the maintenance expense remains, which can be high depending on the cluster size. MCC has zero or minimum establishment and maintenance costs.
- Though clusters are mainly used for computations, other clusters, such as memory and storage clusters, can also be achieved, which cannot be said for MCC.
- A cluster is formed using wired media in contrast to MCC, which connects the SMDs wirelessly.
- The computing resources in cluster computing generally belong to a single administrative domain, whereas in MCC, they belong to individual users.

3.3.2.4 MCC vs Supercomputers

Supercomputing is the oldest attempt to achieve HPC. It is different from all other

HPC paradigms mentioned above. A supercomputer is typically a single, large and non-portable computer stationed at a fixed location and used for applications with complex computation demands. Not only fundamentally, but it differs from MCC in most aspects. The only similarity is the purpose; both aim to fulfil the demands of computing-intensive resources.

3.3.3 Comparing MCC with Other Mobile Computing Systems

In the following, we shall attempt to clarify MCC's misperception with other apparently similar distributed computing systems involving mobile devices. A summary of the comparison is presented in Table 3.3.

3.3.3.1 MCC vs Mobile Grid Computing

The concept of mobile grid computing initially came to integrate the mobile devices with a grid computing system so that the grid service could be availed from mobile devices also [446]. Here, the role of mobile devices was only as resource consumers. However, considering the advancement of mobile devices, many researchers sensed the opportunity to use them as resource providers as well [447] [448]. In general, a mobile grid is a usual grid computing system in which mobile devices also take part as resource consumers, providers, or both. Whereas MCC is purely a mobile grid comprising only SMDs, and except for a P2P MCC, the SMDs are always resource providers.

3.3.3.2 MCC vs Mobile Cloud Computing

Though the terms mobile crowd computing and mobile cloud computing seem nearly homophones, they are different altogether. In MCC mobile cloud computing, the mobile devices offload their works to the cloud [449] [450]. Due to resource limitations in a mobile device, the resource-intensive applications are run on the cloud though it appears they are running on the client mobile device only [451]. Here, mobile devices act only as resource consumers. Whereas in MCC, no external cloud provider is involved.

3.3.3.3 MCC vs Ad-hoc Mobile Cloud

Ad-hoc mobile cloud, as its name suggests, is an on-demand assemblage of neighbouring mobile devices. If a traditional remote cloud service is inaccessible (lack

of internet connection) or undesirable (to avoid latency), the accumulated services of its adjacent other mobile devices are utilised to satisfy the need of a mobile device. In this cloud, the workload and data reside on mobile devices rather than cloud servers [99]. Typically, there is no centralised control in an ad-hoc mobile cloud; instead, it follows a P2P architecture. The purpose, architecture and functioning of ad-hoc mobile cloud are pretty similar to ad-hoc MCC, as described in Section 3.3.5.2.2.

3.3.3.4 MCC vs Mobile Crowdsourcing

Merriam-Webstar[29] defines crowdsourcing as - "the practice of obtaining needed services, ideas, or content by soliciting contributions from a large group of people and especially from the online community rather than from traditional employees or suppliers." Crowdsourcing generally refers to a model for outsourcing tasks to a broad group of humans or machines, or in some cases, to a small group of experts or specialists [452]. It aims to procure services such as data, knowledge, computations, etc., from a large, diverse, and distributed set of service providers (crowd) to attain a set target or a solution cheaply and quickly [453] [454]. Mobile crowdsourcing refers to accessing services, mainly data, information and knowledge, generated or captured by mobile devices [455] [456]. Most mobile crowdsourcing applications' primary goal is to capture context awareness [457] [458]. Though the general perception of mobile crowdsourcing does not consider the computing service, in our opinion, MCC can be seen as a subset of mobile crowdsourcing if we consider computing as a service.

Table 3.3. Comparing MCC with other mobile computing systems

Comparing parameters	MCC	Mobile grid computing	Mobile cloud computing	Ad-hoc mobile cloud	Mobile crowdsourcing
Service type	Computing	Computing, data or storage	Various resource-intensive services for mobile applications	Computing, data or other mobile services	Data

[29] https://www.merriam-webster.com/dictionary/crowdsourcing

Roe of mobile devices	Service provider; both in case of P2P MCC	Usually, service consumer	Service consumer	Both	Service provider
Resource shared by mobile devices	SMDs' processing units (CPU, GPU)	None (if consumer) or processing unit (if provider)	None	Processing unit, data or other mobile services	Data sensed and generated by mobile devices
Service criticality	Very high	High	High	High	Low
Importance of QoS	Absolutely important	Important	Important	Important	Less important
Human intervention	Not required after agreeing to join MCC	Not required	Not required	Not required after joining the ad-hoc mobile cloud	May require, depending on the service

3.3.4 MCC Architectures

Due to the flexibility of the computing nodes in joining each other different topologies of MCC can be laid out, and based on that, MCC can have different architectural models, as shown in Fig. 3.6. Each of the models is characterized by the way how the coordinator connects with other coordinators or crowdworkers. In this section, we discuss each model in detail.

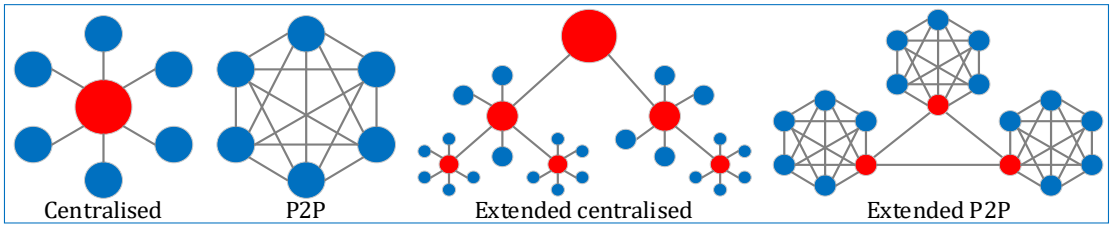


Fig. 3.6. Architectural models for MCC

3.3.4.1 Centralized

This is the most prevalent architecture and most suitable to attain HPC through MCC. In the following, we discuss its basic architecture, components and working.

3.3.4.1.1 Architecture and Working

In a centralized model, the computing tasks across the MCC are managed and controlled by a fixed coordinator from a central point of access. For participating in the MCC, the SMD user should be pre-agreed to be a crowdworker (i.e., sharing SMD resources) by installing the MCC frontend application. In this model, the coordinator communicates with crowdworkers and carries out most of the responsibilities such as searching and selecting suitable crowdworker, task creation, task

scheduling and dispatching to the selected crowdworkers, handling fault issues, etc.

Table 3.4. Comparing four MCC architectures

Comparing parameters	Centralised	P2P	Extended centralized	Extended P2P
Resource coordination	Easy	Not easy	Not complex	Complex
Network infrastructure	WLAN/WAN	Ad-hoc network	WLAN and/or WAN	Ad-hoc network
Resource management	Easy	Difficult	Moderate	Difficult
System maintenance	Negligible	No maintenance	Negligible	No maintenance
Infrastructure cost	Low	No cost	Low	No cost
Scalability	Scalable	Restricted	Highly scalable	Scalable
Ubiquitous computing	No	Yes	No	Yes
Network traffic issue	High	Low	High	High
Susceptible to DoS attack	Yes	No	Yes	Very less
Single point of failure	Yes	No	Yes	No

For processing a large computing-intensive task through MCC, the coordinator split it into several microtasks. Each of these microtasks is queued in a task pool and is dispatched later for processing to suitable crowdworkers. The coordinator searches for the presently available crowdworkers and records their resource details. The available connected SMDs create a resource pool. Among them, it selects the most suitable SMDs as crowdworker for executing the microtasks. The selection is based on various criteria such as processing power, memory availability, battery power, network connectivity and bandwidth, mobility, etc. It then schedules the microtasks from the task pool by mapping the task to a designated crowdworker and dispatching the microtask to that crowdworker.

The selected crowdworkers receive the MCC tasks from the coordinator, execute them non-intrusively, and return the results to the coordinator. The coordinator collects the results obtained from each crowdworker and aggregates them to build the final result. The coordinator's responsibility is to assess the results for error and their validity. The coordinator is also responsible for handling the faults generated due to device mobility, data omission, and other reasons (see Section 3.4.1.5). It

also should take measures to check security and privacy threats (see Section 3.6.2.1). The significant steps for executing MCC tasks in a centralised MCC are shown in Fig. 3.7.

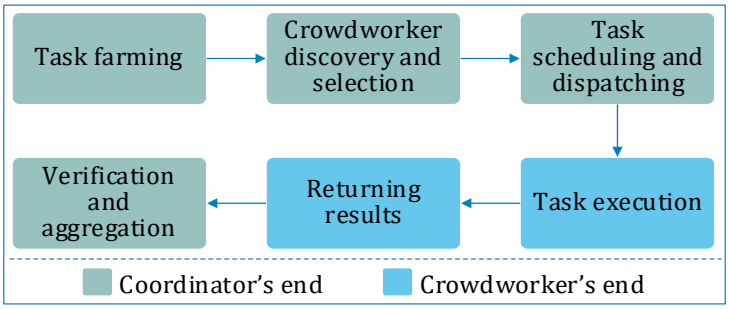


Fig. 3.7. Major steps for executing MCC tasks in a centralised MCC

3.3.4.1.2 Major Components

In the following, we briefly mention a centralised MCC's hardware and software components, which are also graphically represented in Fig. 3.8.

Hardware: A centralised MCC comprises the following hardware components:

- **MCC server:** This computing component hosts the backend and the middleware and acts as the coordinator. It can be a traditional server, a computer, an SBC, a non-mobile and high-end SMD, a programmable edge device such as a Wi-Fi router or switch, or any other competent device having a computational facility with an OS. The selection of the MCC server depends on the application it intends to serve and the infrastructural constraints.
- **Crowdworker:** The crowdworkers are the SMDs that voluntarily take part in MCC and carry out the designated MCC microtasks.
- **Network:** The crowdworkers are connected to the server through WLAN or WAN.

Software: The major software components of a centralised MCC can be categorised as follows:

- **Backend:** The backend component includes the MCC database and the server application.
  - *MCC host application:* It is a computing-intensive application or project that is required to carry out through MCC.

- *Database:* It stores the crowdworkers' details (e.g., login, profile details (see Section 3.4.2.2.2)) and other required data.

- **Middleware:** The middleware is the core part of an MCC which coordinates and accomplishes most of the operations mentioned below. Elaborated discussion on the below-mentioned functionalities can be found in Section 3.4.2.
  - *Resource discoverer:* It is responsible for finding out the SMDs in the network.
  - *Resource profiler and monitor:* It creates a profile of every SMDs connected to the coordinator. The profile describes SMD's software, hardware, performance and log information. It further updates the profile for every subsequent connection the device makes based on its previous MCC performance.
  - *Resource selector:* It selects the suitable crowdworkers from the presently connected SMDs as per the requirement for executing the MCC microtasks. The crowdworkers are chosen based on their profile by considering their capability and performance.
  - *Task farmer:* Its job is to split a large MCC task into batches of microtasks and put them in a task pool.
  - *Task scheduler and dispatcher:* The microtasks in the task pool are mapped to the available crowdworkers for best suitability and dispatched accordingly.
  - *Result aggregator:* This module collects the processed results from the crowdworkers, validates and aggregates them, and prepares the final result.
  - *Fault inspector and handler:* This component checks for any fault in the complete process from the task schedule to obtain the final result (see Section 3.4.1.5).

- **Frontend:** The frontend includes the software applications or set of APIs at the SMD end that allows receiving the tasks from the MCC coordinator, processing and sending back the response to the coordinator. Following are the key components of the frontend:
  - *Task receiver:* It receives the series of tasks from the coordinator and prepares them for execution.

- *Task executor:* It always looks for the availability of free processor cycles in the SMD and executes the given task opportunistically. This module is responsible for ensuring non-intrusiveness (discussed in Section 3.4.1.8).
- *Result dispatcher:* After completion, it sends back the results to the coordinator.

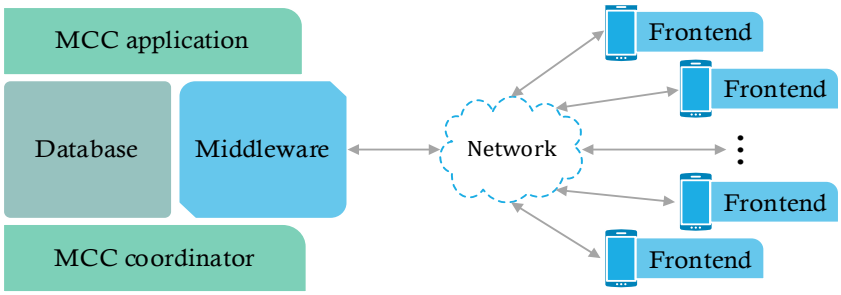


Fig. 3.8. Key components of a centralised MCC

3.3.4.2 P2P

This architecture is primarily useful in an infrastructure-less MCC scenario. In the following, we discuss its basic architecture, components, and working.

3.3.4.2.1 Architecture and Working

A peer-to-peer (P2P) model suggests a system where all participating nodes are equal in different terms. In a general P2P computing system, each node can be a resource consumer and a resource provider. In this architecture, each SMD communicates directly to all other SMDs in the topology. They are responsible for searching for suitable crowdworkers when the need arises and communicating with them for the MCC task processing. The resource-seeking SMD broadcasts the resource requirement in the network. The volunteering SMDs respond by stating their willingness to donate resources. Based on the responses received, the task initiator SMD selects the most suitable ones and sends the tasks to them. To keep up the cluster and task execution, all the SMDs in the cluster continuously broadcast messages among themselves. This causes flooding of messages in the network, creating unnecessary congestion. Also, mitigating faults is more challenging in a P2P MCC than in a centralised one.

A P2P MCC can have the following three entities:

- **Resource seeker:** This is the SMD that requires external computing resources. It initiates the task and sends it to other crowdworker(s) in the topology.
- **Resource provider:** The crowdworker executes the tasks received from the resource seeker and returns the result. Depending on the task's computing requirement, one or multiple resource providers might be required.
- **Leader:** Among the crowdworkers, one is elected as a leader which temporarily acts as a coordinator. This is optional and is applicable only for pseudo-centralised MCC, not required in pure P2P. The resource seeker and the leader are the same SMD in the latter.

A P2P MCC can work either of the following two ways:

- **Pure P2P:** Here, a resource-seeking SMD itself acts as a coordinator and is responsible for all the duties performed by a coordinator, including task initiation, crowdworker selection, task scheduling, and result collection. This architecture lacks a centralized coordinator; hence the resource-requesting SMD itself monitors and manages the kind of service it needs from its peer crowdworkers. The resource seeker may connect with more than one crowdworkers in a one-to-one mode for availing of the wanted resources. The crowdworkers cannot delegate the task further to other SMDs.
- **Pseudo-centralised:** In this form of P2P MCC, among the available crowdworker, one, besides the resource seeker, is elected as the leader which temporarily acts as the coordinator. The usual crowdworker management jobs are done by the leader. The resource-seeking SMD only initiates the tasks and sends them to the crowdworker designated by the leader and later receives the results. This architecture gets the advantages of both P2P and centralised MCC.

The responsibilities of each resource seeker, leader and crowdworkers for pure P2P and pseudo-centralised MCC are shown in Fig. 3.9.

3.3.4.2.2 Major Components

Since the functionality of both centralised and P2P MCC are fundamentally the same, i.e., executing tasks on other SMDs, the basic components are the same; the difference is only in their packaging and localisation. The components of a typical P2P MCC are shown in Fig. 3.10.

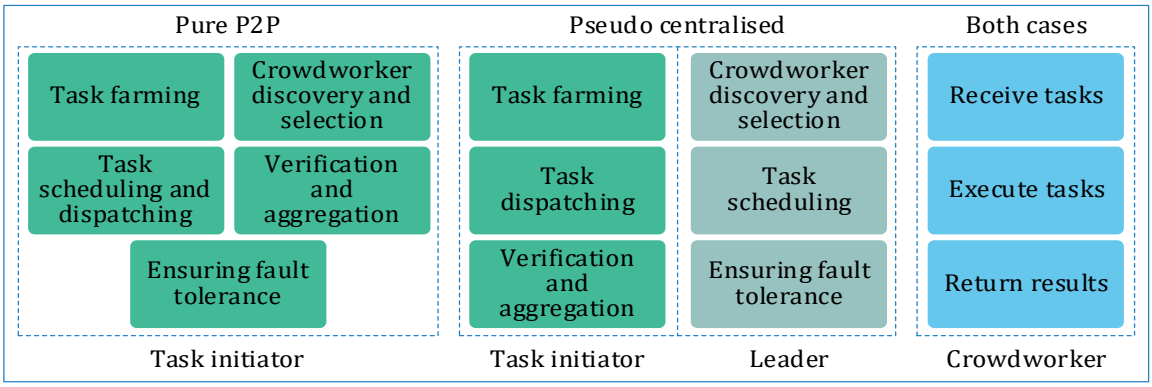


Fig. 3.9. Responsibilities of the entities in P2P MCC

Hardware: In a P2P system, there is no separate MCC server. Therefore, as hardware, only the SMDs and network will be there. Here, instead of an infrastructure-based network, an ad-hoc network is used to communicate between SMDs. On requirement, an ad-hoc network is established between the nearby SMDs through short-range communication technologies such as Bluetooth, NFC, hotspots, etc.

Software: Since there is no separate coordinator, the job of a coordinator is done either by the resource seeker itself of by the elected leader. Therefore, the functionalities of the coordinator (including task farming, crowdworker discovery, task scheduling and dispatching, result collection and aggregation) are embedded with the SMD-friendly MCC package. In a pure P2P MCC, all the crowdworkers should have the same software installed. However, in an asymmetrical P2P MCC, except for the resource seeker and the leader, the other crowdworkers can have only the front-end installed.

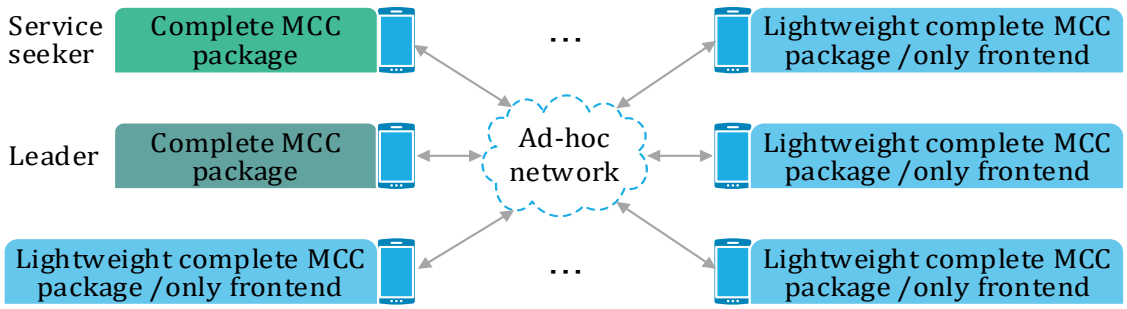


Fig. 3.10. General components of a typical P2P MCC

3.3.4.3 Extended Centralized

This model is an extended form of the centralized architecture and is generally applicable for a largescale MCC. Multiple small centralized MCC clusters are connected hierarchically. A centralised node at the top acts as the chief coordinator,

as shown in Fig. 3.6. Each central node communicates either with the crowdworkers under it, another centralised node under it, or its parent node for resource finding, task distribution, and load balancing.

The extended centralized model helps in offloading job burdens in individual clusters. A particular centralized cluster primarily searches for resources within it and processes the tasks independent of other clusters. Due to uncertainty in SMD availability, the number of crowdworkers in a particular cluster keeps fluctuating. This often causes a scarcity of resources in a cluster leading to task overloading. If a coordinator is out of resources, it forwards its tasks to its children or parent.

Combining and extending network connections through multiple coordinators makes this architecture highly scalable. Though it allows sharing task loads in largescale, it significantly suffers from the overhead of moderating tasks (submitting microtasks and collecting results) from one cluster to another. Especially if the clusters are designed to address any specific type of computing problem, migrating the tasks between multiple clusters requires additional design considerations.

3.3.4.4 Extended P2P

This MCC architectural model allows multiple P2P MCC clusters to coordinate and communicate with each other, extending the small P2P clusters into a large P2P MCC. Each cluster is represented and coordinated by its leader. All the leaders are again connected to each other, thus forming a wider P2P MCC, making a normal P2P MCC more scalable.

If any cluster is found to have a shortage of resources, the leader forwards resource requests to all the peer leaders. If multiple peer leaders come forward for help, the task is forwarded to one among them. The leader of this resource-providing cluster takes the responsibility to get the task executed by the crowdworkers under it and send back the result to the resource-seeking leader, which in turn forwards it to the actual resource-seeking SMD.

Each leader keeps the information about other peer leaders and their corresponding clusters. In case of any change in a cluster, it is broadcasted to all the leaders.

In a dynamic environment like MCC, this message flooding increases data traffic which is a serious issue for an ad-hoc P2P MCC. Further, improper leader selection would make the system unstable, which makes its management a complex job.

3.3.5 MCC Types

Fundamentally, MCC can be implemented either as a local set-up (local MCC) or on a global scale (global MCC) based on the resources exploited and the scale of the network [124]. Further, a local MCC can be infrastructure-based or infrastructure-less. The classifications of different MCC types are laid out in Fig. 3.11, while their topological representations are shown in Fig. 3.12. In the following, we elaborate on each of the MCC types. A comparative summary of three MCC types is presented in Table 3.5.

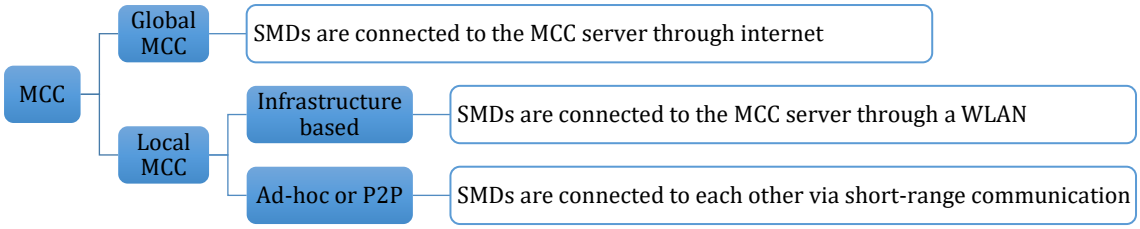


Fig. 3.11. MCC types classification

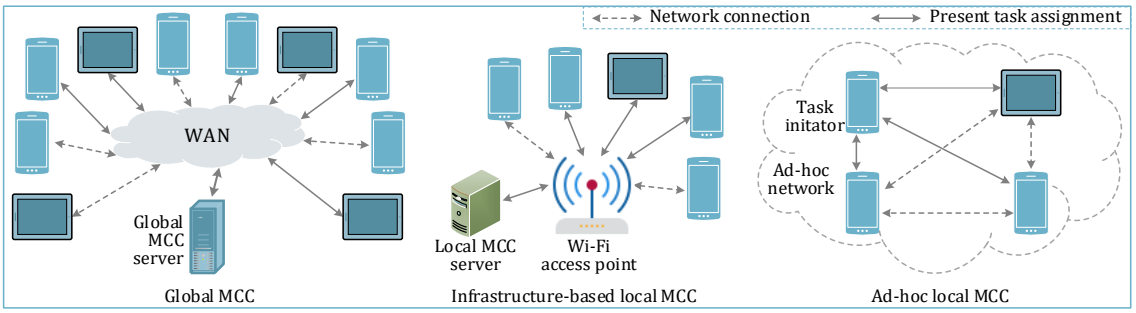


Fig. 3.12. Topological representations of different MCC types

3.3.5.1 Global MCC

Global MCC can be described as a largescale web-based centralized MCC connecting the crowdworkers across the globe. The coordinator in a global MCC is generally a server or a powerful computer. It runs highly computing resource-demanding applications or projects. As in a centralised MCC, the tasks are farmed into several batches of microtasks dispatched to crowdworkers via the internet. Since the crowdworkers are connected through the internet, the crowdworker availability issue is not severe because the SMDs are accessible ubiquitously. Since many

SMDs can be utilised worldwide, a global MCC can garner massive computing power. Global MCC is incredibly useful in satisfying the computing demands of large scientific projects that require vast computation power for solving problems related to physical sciences, astrophysics, mathematics, medicine and health, etc. [145] [459] [460] [461] [462].

Table 3.5. Comparing three MCC types

Comparing parameters	Global MCC	Infrastructure-based local MCC	Ad-hoc local MCC
Number of crowdworkers	Hundreds of thousands	Few hundreds and less	Very few
Network	Internet	Wi-Fi/WLAN	Bluetooth, hotspot
Establishment	Organization	Organization	Anybody, anywhere
Operational periphery	Not limited	Limited to premises	Limited to network range
Cost to SMD owner	Internet data (if applicable)	No	No
Operational fault due to SMD mobility	Low	Moderate	High
Cost for setup	Not much	Very less	None
Reliability	High	Moderate	low

3.3.5.2 Local MCC

Unlike global MCC, a local MCC is connected through local networking. The scope of this type of MCC is limited to a local geographic area where SMDs are connected to a coordinator (centralised MCC) through a WLAN or each other (P2P MCC) through other short-range communication means [463] [464]. Local MCC, based on the usage of network infrastructure, can be of two types - infrastructure-based local MCC and ad-hoc local MCC. In the following, we briefly discuss these two types of local MCC.

3.3.5.2.1 Infrastructure-based Local MCC

An infrastructure-based local MCC system follows a centralised architecture and consists of a coordinator and the nearby SMDs. Here, the coordinator is fixed and can be hosted on a mini server, a computer, or an SBC, but the SMDs are ad-hoc, i.e., they are not enduring. However, in a campus-based MCC, the SMDs are generally re-entrant, i.e., they are regularly available for a certain duration [465]. The SMDs are connected to the coordinator through Wi-Fi. Within a particular Wi-Fi network, a single coordinator manages the crowdworkers connected to that Wi-Fi

access point. The coordinator creates the microtasks, keeps track of the available crowdworkers and based on their resource capacity, the tasks are dispatched, and the results are collected after execution. Whenever a crowdworker connects to a coordinator, it is assumed that the crowdworker is available and willing to lend its computing resources voluntarily or in return for some incentives.

Infrastructure-based local MCCs are found in organizational or institutional premises and cater for local computing. A local MCC can be set up within a campus-based or in-house environment like a college/university campus, office building, libraries, restaurant, shopping mall, factories, etc., where the footfalls of the SMD users are regular and large in number. Organizations may utilise the SMDs available within their premises to set up a local MCC that would serve its organisational computing purposes, not requiring investing in infrastructure-based computers. By taking advantage of many SMDs, organisations can also achieve HPC instead of investing in owning HPC infrastructures or spending on cloud services. Furthermore, the local MCC can be utilized as edge computing infrastructure [140] [76] [143] or mobile cloud computing [40] [75] [72] to process IoT data locally near the source and the sink in real-time. This is helpful for time constraint applications for processing time-sensitive data without sending them to the cloud and avoiding processing delay [30].

3.3.5.2.2 Ad-hoc Local MCC

An ad-hoc local MCC is typically a P2P MCC without any fixed network infrastructure. Here, the SMDs communicate between themselves through short-range communication technologies such as Bluetooth, NFC, hotspots, etc. An ad-hoc local MCC can be formed and utilised on requiring computing resources where a group of SMDs can be found. It can be within a campus such as an office, industry, studio, institute, laboratory, etc. or outside of campus such as roads, fields, parks, accident and disaster sites, etc.

The resource-seeking SMD broadcasts the resource requirement in close vicinity. The neighbouring SMDs willing to donate their resources come together and form an ad-hoc MCC. In this MCC type, it is not always necessary to have multiple SMDs to process a task. Depending on the tasks and resource requirements, an ad-hoc

local MCC can comprise two or more SMDs. If the task size is small and can be processed by a single crowdworker, the topology might consist of only two SMDs – resource seeker and resource provider. And if the task size is big and requires more computation, then other SMDs can be included in the topology to share the task loads.

An ad-hoc local MCC can fulfil the local and instantaneous computing demands. As ad-hoc local MCC does not depend on any fixed network infrastructure, it can be attained anywhere, offering on-demand ubiquitous computing. The ubiquitous nature of MCC allows the processing of the data where ever it is needed near the data source or sink.

The high mobility of the crowdworkers in an ad-hoc MCC makes it unstable. In that case, managing the MCC and maintaining QoS become challenging [124]. As shown in Fig. 3.3, SMDs may have different mobility statuses. Based on these statuses, a resource seeker-provider pair in an ad-hoc MCC may be grouped into different possible categories, as shown in Fig. 3.13. Caution needs to be taken when anyone in the pair is absolutely mobile. If both are mobile and they are moving together (often observed in a group [466]) they can be treated relatively stable and is not a matter of concern. The mobility issue is aggravated in the case of an ad-hoc MCC that follows a pure P2P architecture. A pseudo-centralised architecture can ease the difficulties to some extent. This architecture will also allow scaling of the ad-hoc local MCC by setting up multiple local ad-hoc clusters and connecting them through their respective leaders, as discussed in Section 3.3.5 [467] [468].

3.4 MCC System Design Criteria and Considerations

Designing and developing distributed computing systems are always challenging. MCC adds extra challenges mainly because the computation is done on crowdsourced mobile devices. Ensuring acceptable performance and QoS in a system where the computing entities are neither controllable nor reliable is challenging. This section identifies and discusses the requirements and considerations for designing a functional, efficient and reliable MCC system.

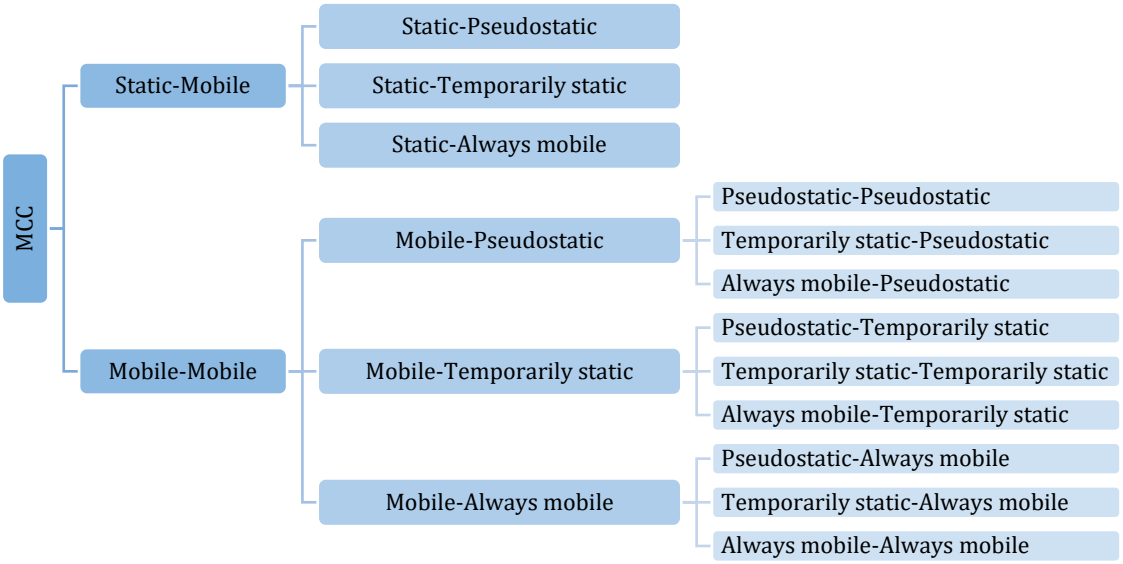


Fig. 3.13. MCC taxonomy based on the mobility of the resource provider and consumer

3.4.1 System Design Criteria

The ultimate success of MCC as a system depends on its quality design. The design goals of an MCC should be to adeptly handle and abstract the inherent issues and make the MCC robust along with delivering expected QoS. In the following, we discuss the specific considerations for designing and developing a typical MCC system.

3.4.1.1 Abstraction

In computing terms, abstraction refers to hiding trivial details from the system designer and/or end user. Crowdsourced systems always set the challenge of abstraction due to heterogeneity. The MCC also involves heterogeneity, generally in the following three aspects:

- **Hardware heterogeneity:** The SMD market is crowded with a variety of makes and models. Most of them differ in diverse hardware specifications such as CPU and GPU clock frequencies, number of cores, other processors such as DSP and ISP, primary and secondary memory, communication technology support and interfaces, etc.
- **Network heterogeneity:** The SMDs might be connected through different networking technologies such as 3G, 4G, 5G, and WLAN. For local MCC also, there are various options as communication mediums such as mobile hotspots, Bluetooth, etc. The SMDs and the coordinator might have different data

transfer capabilities depending on their hardware and the communication network bandwidth, leading to disparity in communication latencies. Assorted latencies create overhead for the middleware for the dependent and time-sensitive tasks.

- **Operating system heterogeneity:** The SMDs might have different operating systems. Although, this heterogeneity is not on a vast scale because, at this time, only two major mobile operating systems are running on most of the SMDs. As per market data[30], in June 2022, Android had a market share of 72%, and the next player, iOS, had 27%. Other mobile operating systems such as Samsung, KaiOS, Windows, etc., have a negligible presence in the market. Besides, the disparity in the operating systems of the coordinator and the SMDs also be considered.

Collaborating with diverse SMDs in a unified way by making these heterogeneities transparent is crucial though not trivial. Obscuring the heterogeneity is also important to achieve seamless interoperability between different entities in an MCC system. The issue of heterogeneity is worsened because many SMDs concentrate on highlighting particular usability (e.g., gaming, photography, music, social networking, etc.). Furthermore, in mobile devices, the components are tightly integrated. This helps optimise the purpose of the device made for because the hardware modules are chosen, particularly according to that. But it may also be possible that the underlying hardware may not support the certain feature(s) of general-purpose software. In general-purpose computers where hardware is not as tightly integrated, this may be handled by swapping out the unsupportive module or plugging in an additional module [469].

3.4.1.2 Generalisation

Ideally, an MCC system should not be application or task-specific, i.e., it should provide a general-purpose computing environment that can support any MCC application with various task specifications and requirements. From a software perspective, the MCC system should be a kind of 'plug-n-play'. The task submitter

[30] https://gs.statcounter.com/os-market-share/mobile/worldwide

should not be bothered about task management details, including microtask creation, data structuring, deciding on processing and storage requirements, etc. It should not also have any hand in resource management (see Section 3.4.2.2). An ideal MCC platform should be able to handle all the necessary measures irrespective of application or MCC task type. An MCC user would only submit a job and get the desired output that is ready to use. The MCC application should be platform and function independent, i.e., it should be generalised enough so that the client application can be installed on SMDs covering a wide range of hardware and operating systems specifications. The coordinator part also should be portable to any computer/SBC/server with equal working efficiency.

3.4.1.3 Adaptability

A system is said to be adaptable if it sustains the short-term and long-term changes in the internal and external operational and environmental changes without compromising on the usual functionality of the system. The SMD market is probably the fastest evolving consumer product market globally. Almost every other day, a new model is launched with upgraded specifications. Commensurately, operating systems are also frequently upgraded to more recent versions. In addition to that, the firmware also gets regular updates and patches. An MCC system should be able to adapt to these changes and revisions effortlessly without requiring to make changes in the MCC software frequently.

3.4.1.4 Reliability

In MCC, the processing units are not owned and administered by a single authority. As a result, maintaining the reliability of the whole system is a great challenge. The coordinator is responsible for verifying the results received from each crowdworker. The erroneous or unsatisfactory results should be discarded and, if necessary, recomputed. The middleware should have a well-defined verification policy and mechanism. Serial offenders (the crowdworkers who send erroneous results regularly) should be blacklisted.

3.4.1.5 Fault-tolerance and QoS

A system is said to be fault-tolerant if it can continue its usual functioning and provide its service even if it encounters expected and unexpected faults which

otherwise might lead to system failure [470]. Due to infrastructure-less-ness and volatility, MCC is inherently fragile. Besides the process-level failures, the dynamic and unreliable wireless environment with its inherent problems, such as the probability of high fault occurrences and bit-error-rates, and random delays [96], can make the MCC more prone to faults. To avoid these inevitable communication problems, steady and stable Wi-Fi connections are desirable. Besides, some foul crowdworker may intentionally relegate the desired service provisioning.

Table 3.6 lists some of the common failures in distributed systems that are equally applicable to MCC [471] [472]. Here, we refer fault tolerance at the abstraction level. We do not discuss the fault tolerance at the core hardware and operating systems level. We believe that the latest Android versions are equipped with sufficient fault tolerance capabilities.

Table 3.6. Failures in MCC

Failure type		Failure site	Description
Crash failure	Amnesia crash	MCC coordinator	Halts and restarts in a predefined initial state before the crash.
		SMD	The app stops working and restarts from a predefined savepoint.
	Partial-amnesia crash	MCC coordinator	Halts, and when restarts, some parts start with the state immediately before the crash while the rest start in a predefined initial state.
	Pause crash	MCC coordinator	Halts and restarts with the state immediately before the crash.
		SMD	The app stops working but restarts with the state before the crash.
	Halting crash	MCC coordinator	Stops and never restarts.
		SMD	The app stops working, or the SMD is hanged or switched off.
Omission failure	Receive omission	MCC coordinator	Fails to receive results.
		SMD	Fails to receive tasks.
	Send omission	MCC coordinator	Fails to send tasks.
		SMD	Fails to send the results.
Response failure	Value failure	SMD	Sends incorrect result.
	State transition failure	MCC coordinator	Fails to synchronise the flow of control correctly.
Timing failure		SMD	Fails to send results within the specified time.
Arbitrary failure		SMD	Produces arbitrary results at arbitrary times.

Making a system fault-tolerant comprises two phases – fault detection and recovery. Detecting faults in the case of an independent task is not complex. Only while aggregating the results, it is to be taken care of. But detecting faults for dependent tasks with multiple workflows is really challenging. Several solutions can be found

in the literature for the classic distributed system research [470] [472] [473] [474]. However, considering the uniqueness of MCC, incorporating them straightforwardly in MCC would not be effective. The fault can either be at the crowdworker's end or at the coordinator's end. After correctly detecting the fault, it needs to be handled properly without affecting the integrity of the MCC system.

Over the years, several approaches have been proposed to mitigate faults in distributed systems [475]. One of the most common approaches is redundancy or replication [474]. If the same task is submitted to multiple nodes, it is highly improbable that all the instances would be faulty. However, the obvious problem with this approach is the wastage of resources. Furthermore, in MCC, a large task is generally split into multiple smaller microtasks, each having a certain input, processing and output load. Excessive replication of these microtasks would enlarge the overall processing chain, especially if there is interdependency between the microtasks [124]. This would, in turn, increase the overall completion time. Moreover, as the processing chain gets longer, the probability of faults also increases. Therefore, a proper balance between the expected level of fault tolerance and replication is highly recommended.

Another common approach to imposing fault tolerance in distributed systems is rollback and recovery. The system periodically maintains checkpoints which save the system state at particular instances [476]. If the system crashes, it returns to the most recent checkpoint and restarts. Instead of setting checkpoints, some systems continuously log the system events [477]. In case of a crash, the system is restated using the logged information. This logging approach is fine to handle the coordinator failure but not ideal for SMD failure. Continuous logging would require considerable space, which is not so affluent in the SMDs. Also, frequent logging would interfere with the processing and slow the execution of tasks. Checkpoint is a better option, but setting them too densely would cause additional overhead for the client app. So, a fair balance must be maintained between the checkpoint scales and acceptable loss due to failure.

Replication and recovery are particularly challenging in MCC. It is difficult to implement them in a loosely coordinated environment like MCC, where resources are

likely to behave erratically, especially in opportunistic MCC. Therefore, it is recommended to go for a reasonable quasi-opportunistic MCC whenever possible. Nevertheless, implementing fault tolerance for acceptable QoS is crucial for successfully realising MCC, especially for commercial and critical applications. An ideal middleware should mask all the possible systems faults, i.e., even though some failure occurs either at the coordinator or the SMD end or in the network, it should be transparent to the system and should not hamper its usual functionality.

The omission failure generally happens due to network failure. In a global MCC, replication and recovery are the only options to tackle this. However, in a local MCC, it can be handled with a little bit of improvisation to save the task/result loss. In a Wi-Fi-connected MCC, the signal strengths of the SMDs are monitored. If a user continues to move away from the access point, the signal strength of the SMD weakens slowly. Two thresholds of the signal strength are set. Whenever it starts decreasing, the first threshold is checked. If it crosses the first threshold, the system will start setting savepoints and taking backups. When it crosses the second threshold, the job is withdrawn from the SMD and submitted to another one which starts from the last savepoint. In a P2P MCC, even if the crowdworker tends to move, the task need not be offloaded if the task initiator is also moving along, i.e., they are relatively stable [466].

As the number of task reassignments increases, the QoS gets more affected because it has to be reassigned to another SMD, which introduces a significant delay or, in the worst case, would result in job loss [249]. To minimise the task offloading circumstances, it should be ideal to assign the job to the crowdworker, which is supposed to be stable and would stay within the grid for a long time or at least till the task is finished and the result is returned, as discussed in Section 3.4.2.2.4.

3.4.1.6 Scalability and Elasticity

A medium or large-scale MCC would have a great number of crowdworkers connected. An ideal MCC should be competently scalable to cope with this large number of SMDs and the uncertainty in their availability and reliability. Furthermore, the MCC should be elastic enough to accommodate the sudden up and down surge in the crowdworker requirement.

3.4.1.7 User Friendliness

Moreover, resources in crowd computing are possessed and functioned by laymen, most of whom are non-technical people rather than professionals. Given that, installing and operating the client application should be straightforward.

3.4.1.8 Non-intrusiveness

One apprehension of the users for participating in MCC is the concern that their native applications in the SMD might get hampered in running the microtasks. To avoid this intrusiveness, the resource utilization of the microtasks on a crowdworker need to be strictly regulated and controlled so that the primary functionalities of the crowdworker in no way get affected. The client application needs to be designed to ensure that the microtasks will not be running when a resource-demanding native application is running. A microtask should be allowed to run only if the resource availability meets a certain threshold [478]. Also, the client application should be able to automatically lower down or stop resource consumption when the crowdworker's own computing requirement goes up. Non-intrusiveness is one of the vital design goals of MCC, not only for attracting new crowdworkers but also to retain the existing ones [82].

3.4.1.9 Energy Efficiency

As discussed in Section 3.6.1.1, battery constraint is an issue for computing on SMDs. The fear of fast power loss might deter SMD users from participating in MCC. Therefore, the client application must maximize energy efficiency in its operations while maintaining the required performance level and QoS. Moreover, in many MCC use cases, especially real-time sensor-based and IoT applications, the coordinator is hoisted on a battery-operated SBC. In these cases, one of the primary design goals of the middleware should be to minimise the power consumption accounting for its operations so that the working life span of the SBC extends.

3.4.1.10 SLA, Liabilities and Legalities

A service-level agreement (SLA) typically notes a commitment or pre-set rules and conditions, describing different aspects of the service, such as quality, availability,

responsibilities, etc., that are agreed upon by both the service provider (here, crowdworker) and the service user (MCC host). For a commercial MCC, documenting and adhering to a well-defined SLA is very important. Though strict enforcement of SLA in volunteered MCC is not very sensible, it certainly can be for non-volunteered MCC where the crowdworkers are incentivised for their services. To maintain a steady QoS, it is to be ensured that the services received from the crowdworkers are always up to the mark. The common aspects that an SLA for MCC should cover are:

- **Scope:** Services that are expected from the crowdworkers, and the services that are not being covered in the SLA.
- **Constraints:** Specifying minimum hardware and software requirements for certain microtasks.
- **Computing performance:** The minimum performance benchmark in terms of computation, such as throughput and turnaround time.
- **Network performance:** Minimum network quality to be maintained for efficient data transmission and to avoid the loss of data and results.
- **Timeliness:** The results are to be returned within a stipulated time, i.e., till they are relevant and valid.
- **Availability:** Once agreed to be a crowdworker, an SMD should be available for a certain duration in a particular session.
- **Workload:** The minimum and maximum workload per session for a crowdworker should be pre-decided.
- **Service failure:** If any alternative measures can be taken in case of service failure.
- **Truthfulness:** The validity of computations and the results to be assured.
- **Security:** It is to be ensured that there is no security threat or breach from both crowdworker and MCC ends.
- **Penalties:** The penalties applied in case of agreement violations (i.e., if a crowdworker fails to meet the promised service level expectations).
- **Termination:** Specification of the SLA termination criteria and procedure, ensuring the associated configuration information is removed from the SMD.

Furthermore, if the termination is due to an SLA violation, the likely actions to be taken against the violating party depend on the degree of the breach.

Enforcing the SLA involves some legalities. But in a crowdsourced system, it is non-trivial to demarcate the liabilities in case of serious failures and resultant losses. This makes it difficult to set the jurisdiction in case of any law infringement. For example, who should be held legally responsible if the MCC project sends data to a crowdworker for unauthorised processing? The legalities get complicated in the case of a global MCC. Enforcing penalties is challenging as the penalty clauses might function differently in different countries. For a transparent and legal-hassle-free MCC implementation, proper permissions, disclaimers, policies, protections, encryption, and remedies need to be outlined and implemented at all levels [479].

3.4.2 System Design Considerations

The efficiency of an MCC system depends on various factors. Several aspects need to be carefully considered to run the system efficiently. In this section, we discuss the most crucial ones among them.

3.4.2.1 Determining Architectural Model

Adopting a suitable architectural model is important for the proper implementation and utilization of MCC. This choice should vary as per the type of MCC application, as well as other environmental conditions. For example, in the case of a requirement of small-scale organisational HPC (e.g., scientific computing), a centralised MCC would be ideal. Whereas, in the case of a large-scale HPC and inter-organisational MCC, a hierarchical system would be preferable. On the other hand, when the number of available SMDs is less, and the computing load is low, centralized or P2P architectures may be attractive. The choice of the architecture needs to be determined by estimating the effectiveness of the architecture in a particular situation and the performance expectation. However, an ideal architectural framework should be able to adapt to dynamically changing requirements, complying with the reactive service management architecture for MCC.

3.4.2.2 Crowdworker Management

The success of any distributed system depends on efficient resource management. The same applies to MCC as well. In this section, we deliberate the different aspects of resource (crowdworker) management in the context of MCC.

3.4.2.2.1 Crowdworker Discovery

A crucial and primary feature for the usability of MCC is the discovery of available SMDs. The MCC system should be able to automatically locate the potential crowdworkers, either by following periodical or on-demand discovery approaches [82]. In periodical discovery, the coordinator regularly learns about the connected crowdworker. The periodic approach can further be categorised as the pull (proactive approach from the coordinator's end) and push (reactive approach from SMD's end) method. In the pull model, the coordinator continuously polls the network to search for the new crowdworkers. This approach is a bit costly and involves a delay in the discovery process. In the push method, the crowdworkers spontaneously declare their availability by alerting the controller whenever it joins the MCC network. But the negative side of this approach is that the control is in the crowdworker's hand. For a balanced discovery, a combination of both methods can be employed. On the other hand, the on-demand discovery (reactive approach from the coordinator's end) follows the pull method where the controller checks for the available crowdworkers only when required, i.e., some tasks are to be executed.

For crowdworker discovery, whether to use a periodical push or periodical pull or an on-demand pull model is an MCC design decision. In either case, the coordinator should be able to handle the false and masqueraded crowdworkers. An effective crowdworker discovery approach that is compatible with the MCC application requirement and infrastructure would enhance the efficacy of the MCC.

3.4.2.2.2 Crowdworker Profiling

In the context of MCC, SMDs can be characterised by several resource attributes [36]. Some are fixed in nature, such as the clock frequencies of the CPU and GPU, no. of cores in CPU and GPU, the processing capability of other on-board

processors, total memory, battery capacity, etc. Some attributes are variable, i.e., their values change dynamically, such as current CPU and GPU loads, available memory, available battery, and signal strength for communication. Besides, some other persistent parameters, such as crowdworker's availability and reliability, user's mobility pattern, etc., denote aggregated observations over a period and vary on a smaller scale compared to the dynamically variable attributes.

In MCC, precisely profiling and assessing these resource attributes is an essential prerequisite so that the microtasks can be mapped to the most suitable crowdworker. However, due to the heterogeneity of the SMDs in terms of their resource types and capacities and the dynamic variability of these resource attributes, profiling them and assessing their fitment for different requirements are not trivial. Because capturing and storing these attributes' values require different policies and implementations. To profile the static and dynamic resource parameters of the SMDs in real-time, a systematic methodology and model need to be designed and developed.

3.4.2.2.3 Crowdworker Selection

Based on the profiled resources of the potential crowdworkers, the most competent and fitting crowdworkers are selected for the tasks to be executed. Due to assorted heterogeneity, as discussed in Section 3.4.1.1, it is obvious that not all the crowdworkers are equally potent in terms of computing ability. Therefore, to get the best performance out of MCC, selecting the most suitable crowdworker as per the task requirement in different application scenarios becomes crucial. The presence of several attributes of different criteria makes the crowdworker selection a multicriteria decision making problem [480].

The selection parameters differ depending on the MCC model. In an MCC where the crowdworkers are generally common, the crowdworkers' historical information (such as mobility, availability and resource usage patterns, reliability, etc.) can be considered. But in an ad-hoc MCC, the MCC coordinator does not have any historical information about the crowdworkers. The selection should be made on-the-fly. The problem with the on-the-fly selection is that it may not reflect the

actuality of the candidate. Because an SMD may seem fittest in the present context, it may happen that this fitness may not be consistent over a more extended period.

3.4.2.2.4 Crowdworker Availability

One major hindrance in maintaining QoS in MCC is the uncertainty of crowdworkers' availability. Crowdworkers' sudden dropping out of the grid severely affects the performance and QoS of the system. A crowdworker might go off due to connection termination (intentionally or unintentionally), device malfunction/switch off, or users' mobility (for local and ad-hoc MCC). The availability issue can be either overall or for a particular SMD to which a task is to be assigned or already assigned. Different approaches are needed to handle these two issues, as discussed in the following.

3.4.2.2.4.1 Availability of Sufficient Crowdworkers

The scale of MCC depends on the number of crowdworkers connected at any point in time. In fact, the uncertainty of getting sufficient SMDs when required makes MCC undependable. Especially, running critical and time-bound applications is not a good idea if there is the slightest probability of unavailability of a sufficient number of service providers through the course of job completion. To make the system reliable, it is to be assured that the minimum required number of SMDs should be available at any point in time, which may not be achievable in all conditions. The good thing is that, as mentioned in Section 3.2.2, due to mass adoption, there will not be a dearth of SMDs. Even for local MCC also, Loke *et al.* [141] estimated that there is a high probability of finding a sufficient number of SMDs. Only, they need to be tapped by motivating the users, maybe with lucrative incentivising as discussed in Section 0 and 3.6.2.3.

3.4.2.2.4.2 Availability of a Particular Crowdworker

One major issue in MCC is handling the uncertainty of the device availability due to various reasons such as users' mobility, network failure, device malfunction, etc. [249]. If a crowdworker leaves before completing the assigned task and returning the result, the task should be resubmitted to another crowdworker and restarted from the beginning [481]. This negatively affects the performance and QoS of MCC.

And if this frequently happens for many crowdworkers, the performance degradation would be substantial.

One way to mitigate this is to have prior knowledge of the probable departure of the crowdworker before submitting the task. One option is that each crowdworker declare their availability period when they join an MCC session [140]. However, it cannot be taken for granted every time. Besides the unintentional causes of leaving (e.g., network failure, device hanged/switched off, etc.), some dishonest users may leave before the declared departure time. Another option is to predict the availability of a particular crowdworker by analysing her connection or availability history [156]. Before submitting the task to an SMD, the probability of its availability till the job is finished is assessed. If the predicted availability is greater than or equal to the job length, then only the job is assigned to that particular SMD [465], as shown in Fig. 3.14. But every time calculating this before the job submission will delay the job submission process. To avoid this, the prediction algorithm may run periodically (with a small periodical gap) in the background, and the job submission module can consult the availability prediction module before job submission [36]. This approach is suitable where the crowdworkers are recurring, i.e., they have a fixed timing pattern of joining MCC. Although the abrupt issues cannot be evaded by prior planning, the issue of intentional departure can be mitigated reasonably by this approach.

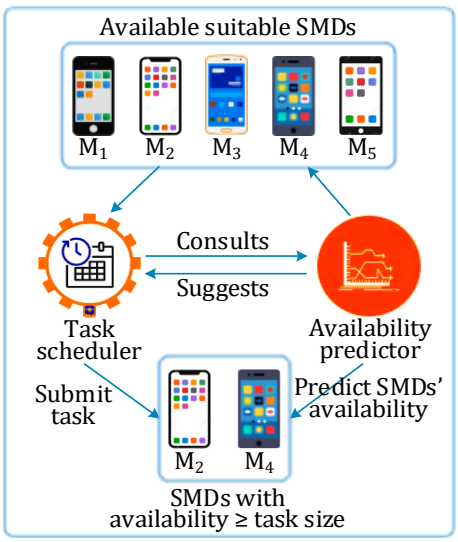


Fig. 3.14. A crowdworker availability-based task assignment scenario

3.4.2.2.5 Crowdworker Monitoring

One-time crowdworker selection does not end the course. Depending on the MCC application and task granularity, tasks must be scheduled regularly. For every scheduling event, suitable crowdworkers need to be selected. Considering the dynamicity of the MCC environment, continuous monitoring is required. The changes in the status of the variable and persistent resource attributes need to be regularly updated in the corresponding crowdworker's resource profile, which should be reflected in the next crowdworker selection decision. Continuous monitoring is also required to keep track of the unavailability of the existing crowdworkers. Like discovery, crowdworker monitoring can also follow either of the three approaches - periodical push, periodical pull, or on-demand pull. In the periodical pull method, the controller periodically fetches the resource status from the crowdworkers. In contrast, in the periodical push method, the crowdworkers send their resource status information periodically to the controller. Here also, a combination of both of the methods can be employed. In on-demand monitoring, the controller procures the resource information from the crowdworkers only when required, i.e., a new task is to be scheduled for which a suitable crowdworker needs to be selected.

3.4.2.3 Task Farming

The proper utilization of MCC can be realised only when a larger task can be divided into several granular microtasks, which are sent to the crowdworkers for execution. Ideally, these tasks should be independent, i.e., parallelly executable. Usually, the distributed systems follow a predefined task farming policy. However, static task farming does not hold good in a complex application (e.g., interactive, real-time, or process with dynamic and variable inputs). In these cases, the microtasks with varying sizes are created on-the-go. This makes resource discovery, selection, and scheduling complex. Because the crowdworkers need to be compatible with the tasks to be assigned to them, this mapping is to be done dynamically for every task assignment. Considering this, innovative approaches are needed to realise a more straightforward yet robust task farming mechanism.

3.4.2.4 Task Scheduling

Efficient task scheduling is a crucial feature of distributed computing. It becomes more important when the computing resources are mobile and crowdsourced. A non-smart scheduler, to avoid complexity, would follow the straightforward task allocation approach, i.e., whichever crowdworker is available, send the task-in-hand to it if it satisfies the resource requirements. But this straightforward approach not only degrades the system's overall performance but also becomes a serious issue in a crowdsourced system. In the following, we acknowledge some crucial properties that need to be considered for framing a decent scheduling policy.

3.4.2.4.1 Optimized Scheduling

The MCC is comprised of public-owned devices which are vastly diverse in terms of resource capacity and quality. In such an environment, it is pretty challenging to assign tasks to SMDs according to their resource capability and usability. An optimised MCC scheduler should be targeting to minimise the overall makespan and response time while maximising the throughput. The scheduler should also target to utilise a minimum number of SMDs to minimise the failure probability and operational overheads and costs. The microtasks may vary in terms of task length, resource requirements, errors sensitivity, and result verification requirements [83]. An ideal scheduler should adopt an intelligent and dynamic mapping mechanism for scheduling the tasks to the most appropriate SMD, considering the above-mentioned decisive factors.

3.4.2.4.2 Energy-aware Scheduling

Due to the limited battery life of the SMDs, an essential criterion of the scheduler is to be energy-aware. The battery level of the SMDs should be included in the scheduling criteria, i.e., an SMD with a lower battery level should be refrained from being assigned tasks frequently. The most energy-consuming attributes are task processing in CPU/GPU, memory transfer and data transmission (between SMD and coordinator) [253]. Inefficient scheduling can significantly increase communication traffic and, in turn, energy consumption. This applies to both global and local MCC. Controlling the power consumption for processing and memory operations is mainly in the hand of the client application. Therefore, the client

application also needs to be energy-efficient, i.e., it should be designed to consume minimum energy in managing the execution of the assigned task. Furthermore, to achieve sustainable computing, the scheduler should ensure that the total energy consumption to complete the task is minimal.

3.4.2.4.3 Balanced and Fair Scheduling

It is commonplace in an opportunistic computing system that a small number of participatory nodes get overburdened most of the time, while the rest, the larger segment of the participants, contributes meagrely or, in some cases, not at all. Also, a typical scheduler might always look for the most competent crowdworkers and schedule the task to them to achieve a better system performance. But this approach would overburden a small number of SMDs. Putting excessive load on an SMD may lead to fast battery drainage and hardware stress, which would coerce the users to drop out of the MCC. For a satisfactory retention ratio, a decent level of QoE should be provided to the crowdworkers by implementing a fair and balanced task allocation scheme [482] [483].

This is true for a P2P MCC also. Some users may be eager to use the resources of other peers without the fair contribution of their own computing resources. These types of selfish peers are called 'free riders'. As a result, the good peers become easily congested and overloaded. To achieve a unbiassed resource utilisation by tackling with these selfish free riders, proper incentive and pricing schemes or penalty policies are required.

3.4.2.4.4 Dynamic Scheduling

The high mobility of SMDs can debilitate the consistency of the system. If the SMDs frequently get in and out of the MCC grid, the job scheduling and allocation would be a real challenge. This factor makes designing an MCC scheduler more challenging than other traditional distributed computing. An efficient MCC scheduler should be adaptive and flexible enough to schedule the task dynamically depending on the availability of the crowdworkers.

3.4.2.5 Resource Scavenging

The MCC client should adopt an efficient, non-intrusive resource scavenging

policy. Suppose a crowdworker is executing an MCC task, and in a while, if the SMD needs to carry out some heavy computational job of its own, the MCC job should be suspended. If the MCC task is time-bound, then the suspended task might need to be offloaded to another crowdworker and continue thereon. As in context switching, this shifting procedure is also costly in terms of time, fault tolerance, and reliability. So, if the application is delay tolerant, it may be allowed to halt for a while and wait at the same device instead of offloading the job. The suspended MCC task is resumed when it finishes its own job and the resource is available once again. The answer to the obvious question, "how long should it wait?" may be decided on different criteria and considered separately.

3.4.2.6 Opportunistic Computing

An efficient MCC should be able to utilise the opportunistic scenario to its fullest advantage. We suppose the opportunistic computing aspect is more pertinent to local MCC. The opportunistic scenario can be pure or quasi, as discussed in the following.

Pure opportunistic scenario: Whenever an SMD enters the range of a Wi-Fi hotspot included in a local MCC infrastructure, it will be considered a probable crowdworker. In this case, the MCC coordinator continuously hunts for a new entrant in the local MCC topology. If the SMD pass the minimum threshold criteria for resources, it is considered for task scheduling. Opportunistic computing does not guarantee continuous resource availability; thus, reliability and fault tolerance are also not guaranteed. This could affect mission-critical applications heavily. This might happen, particularly where the probability of hanging around phone users at and around a particular location during a specific time interval is awfully unpredictable.

Quasi-opportunistic scenario: Instead of depending on the random nature of resource availability as in opportunistic computing, in quasi-opportunistic computing, it is made sure that the resources are available for a certain period [484]. Before that specified time, no crowdworker should leave once it has been hooked to an MCC. This can be assured through some pre-agreement (SLA), economic or other incentives, intelligent prediction of resource availability or by other means.

For mission-critical and real-time applications, every crowdworker should complete the assigned task without any or with minimal interruption. Exiting the MCC topology should be avoided until the assigned task is completed. In case of emergency and with prior reporting, a crowdworker is permitted to leave in the midway only if it is assured that there is a suitable alternative. Quasi-opportunistic computing can be achieved if the MCC is performed within a campus, where there is a certain probability of the presence of a certain number of SMDs during a certain period. For example, institutes (school/college/university), libraries, large govt. offices, corporate houses, production units, shopping malls, cinema halls, public transport (preferably for long distances), etc., are suitable sites for campus-based quasi-opportunistic MCC. In these sorts of places, it is possible to predict the number of accessible SMDs and the time duration of their availability, accordingly to which the MCC tasks should be scheduled.

3.4.2.7 Workflow Management

Ensuring fault tolerance in MCC becomes more complicated if the tasks involve multiple workflows. A workflow system can be defined as a collection of synchronous or asynchronous processing tasks, executed on the same or different nodes and organized to accomplish a bigger goal [485]. A distributed task with workflows is expensive to recreate if not completed successfully. A failed workflow may have severe ripple effects. Therefore, improper workflow management would destabilise the system. An ideal workflow manager should be able to orchestrate the whole operation by synchronising and managing the processes correctly and timely. A workflow can fail due to network failure, device unresponsiveness, unexpected latency, or incorrect intermediary results. Many efforts have been made toward handling workflow management in distributed computing [475], grid computing [486] [487], mobile computing [488], and ubiquitous computing [489]. The uncertainty in the continuous availability of the devices in MCC demands extra efforts in this regard.

3.4.2.8 Result Verification and Aggregation

To make MCC reliable, it is to be ensured that the tasks are correctly executed; in other words, the processed outcome is correct and valid. Since the tasks in MCC

are divided into microtasks executed on different SMDs, the result verification gets a little intricate. First, the results for each microtask received from the assigned crowdworkers must be verified. And then, these results are aggregated to get the final result which needs to be finally validated. However, the complete process of verifying, aggregating and validating the results received from heterogeneous computing nodes is not trivial. For this, a suitable framework and policy are to be adopted [83]. There should be unambiguous and well laid out decision policies in the cases, for example, what to be done with the partial results, what should be the standard for verification and validation, if the results could not be immediately verified how to proceed with the aggregation and validation, etc.

3.5 Advantages of MCC

The unique nature of MCC has spawned several pluses. This section identifies and discusses MCC's potential advantages and benefits.

3.5.1 General advantages of MCC

Following are the advantages of MCC in general, i.e., they apply to all types and models of MCC.

Cost-effective: Institutions and organisations do not have to invest in buying and maintaining extremely pricey HPC systems. They can reap the same service by employing MCC through smart policy adoption. MCC will also substantially reduce the electricity bill by eliminating the electricity expenditure for operation, ventilation, and air-conditioning of the IT systems that may include clusters and servers.

Least overhead in IT infrastructure management: Since the resources are yielded by the public and not owned and managed by the organisation, it should not be worried about maintaining the IT infrastructure support system, and the IT team can ponder on more productive works.

Free from DDoS attack: Since the computing has been distributed over many devices, MCC gives little scope to the DDoS attackers.

Offering computation offloading: Due to several reasons such as insufficient resources, saving energy, and maximising throughput, sometimes it becomes desirable to offload the workloads of the mobile devices [490]. A P2P MCC would

allow other SMDs to offload their workloads, when needed, to other SMDs without needing any other external services such as the cloud.

Scalability and agility: Today's computing workloads are generally dynamic and unpredictable, and since the datasets can often grow unpredictably and, in many cases, exponentially, the systems are required to be able to scale up or down easily and quickly, depending on the workload. Depending on the availability of the SMDs, MCC is dynamically scalable as per workload.

Amply available resources: Because of the massive adoption of SMDs, there is every possibility that a sufficient number of devices would be available for computation even at scantily populated sites. It would ensure the availability of MCC-powered HPC.

3.5.2 Benefits of Local MCC

In addition to the above-mentioned advantages, a local MCC specifically offers the following benefits.

On-premises: MCC can be set up locally utilising the on-premise devices and local network. A local MCC does not require to be connected to the internet. This offers a great advantage for sites where internet bandwidth is scarce or not available.

Lower latency: A local MCC has significantly lower latency than internet-based services such as the cloud. This makes MCC a suitable computing option for improving user experience, especially for interactive applications requiring minimised response time.

Minimized network cost and congestion: Not requiring the data to be sent to external servers for processing saves communication cost and minimises intra- and inter-network congestion.

3.5.3 Ubiquity and Pervasiveness of MCC

Depending on the availability of the SMDs, an ad-hoc MCC can be set up pervasively. Following the advantages of MCC make it an ideal platform for ubiquitous and pervasive computation.

Anywhere HPC: MCC gives us the flexibility to build up a nomadic HPC facility anywhere, irrespective of the fixed architecture and internet connection. Since SMDs are being increasingly used ubiquitously, a collection of such devices connected through pervasively available wireless connectivity will provide a ubiquitous HPC facility.

Location and context awareness: The local nature of MCC allows it to be able to provide location- and context-aware services, including computing, analytics, local points of interest, businesses, events, and many more.

Application specificity: Thanks to its agility, it can deal with a range of business-specific applications which require ubiquitous computations.

Suitable for real-time applications: Since MCC can be set up locally, it is perfectly appropriate to cater for the needs of real-time applications that demand time-constrained responsiveness. Most real-time systems that process real-time data streams require intensive computing resources [451]. The HPC offered by MCC can be suitably utilised for this.

Proximity: Closeness to the data source and sink of an ad-hoc MCC is a crucial enabler for its ubiquity and pervasiveness.

Network flexibility: An ad-hoc MCC offers great networking flexibility since it can be established using short-range communications such as Bluetooth and devices' hotspots in the absence of infrastructure-based networks such as the internet and WLAN [55].

3.5.4 Sustainability of MCC

Compared to other dedicated HPCs, MCC offers significant sustainable advantages, as discussed in the following.

Energy-efficiency: MCC can be a prodigious benefactor in green computing. Usual HPC systems require enormous electric power to run and cool. For example, ASCI Red needed 500,000 watts of power to run with additional 500,000 watts just to cool off the building it was kept in [491], whereas Tegra X1 required less than 15 watts of power [409]. SMD CPUs are more than 20 times more power efficient than desktop CPUs while delivering nearly equivalent computational power [114].

Accordingly, MCC is considerably more energy saver compared to cluster and grid systems. Ubispark (discussed in Section 2.2.2) estimated that when a task is executed on a cluster of nine Samsung Galaxy S4 smartphones, it consumes only 7.2% of the electricity consumed by an HP Proliant server to run the same task. Running applications on SMDs will indeed consume extra energy, but it also implies that the hardware is utilized more. A study suggests that total energy consumed by mobile devices, on account of usage, is only around 25% of the total energy spent on it through its lifetime, while the rest 75% is guzzled during production [492]. Therefore, it will be sensible to do all-out utilization of the device on which already so much energy has been drained. Furthermore, distributing the workload over a large number of separate SMDs would reduce the power consumption of each device [105]. As associate technologies and mobile platforms are advancing continually, it is highly expected that the computation per watt that can be garnered from SMDs, will be even much higher in coming years [93].

Environment-friendly: The production of computers takes a massive toll on the environment. For instance, to make a computer with a 17-inch CRT monitor, on average, 1500 litres of water, 240 kg of fuel, 22 kg of chemicals are needed, which costs approximately a total of 1.8 tons of material [10] [11]. Besides, it generates a significant amount of hazardous e-wastes contaminating the earth. MCC utilises already existing resources, i.e., public's SMDs. Users would buy and use SMDs anyway. Optimal and multipurpose use of these devices will restrain the production and use of new computers. This will certainly minimise the environmental hazards caused by production and e-waste [2]. Moreover, the small size of SMDs also relieves the adverse effects to some extent since they require less material in manufacturing and also, the contribution of e-waste of discarded SMDs will be considerably less. Furthermore, using MCC will not incur additional energy consumption compared to dedicated HPC systems. This will lessen the need for electricity generation and the use of fossil fuels, positively impacting the environment.

3.6 Issues and Challenges

Successful implementation is challenging. Several crucial as well as trivial issues need to be addressed. In this section, we discuss them extensively.

3.6.1 SMD and Communication Issues

Here, we discuss some of the elementary issues associated explicitly with SMDs and networking.

3.6.1.1 Battery Depletion

While portable computers, SMDs, wearable computers, and other mobile devices are growing ever more advanced, technologically and architecturally, they're still limited by power. The battery or energy technology hasn't moved at par in decades, which pulls back the pervasive and ubiquitous computing revolution. In view of that, the most concerning facet regarding MCC is its limited battery power and fast drainage. In a recent survey, nearly 70 percent of respondents stated that battery life is the biggest limitation of their mobile phone, and most are willing to pay more for phones that offer extended power [493]. Present SMD batteries are struggling to keep up with users' active and ever-increasing SMD usage demands. And it will only get worse as next-generation 4G networks come online, giving phones access to high-speed, always-on connections and torrents of data. The major factors that escalate battery drainage in an SMD are shown in Fig. 3.15.

However, the scenario is not that gloomy. Hopefully, we might witness a power revolution very soon [494]. Research groups in universities and organizations are coming up with innovative ideas to either extend battery life or minimize the recharging time or accomplish both. Some are exploring alternate power sources like ambient energies, including light and sound, bio-mechanical, etc. or for charging SMD's battery. Especially the emergence of wireless charging has augmented the prospect of powering mobile devices persistently and ubiquitously. Table 3.7 lists some prospective aspects of SMD battery and charging in which researchers are presently focussing and realising success. Considering these advancements, we are very much optimistic that the days are not far when the users will no further be haunted by the horror of 'low battery' and the full potential of SMD's capabilities will be realised.

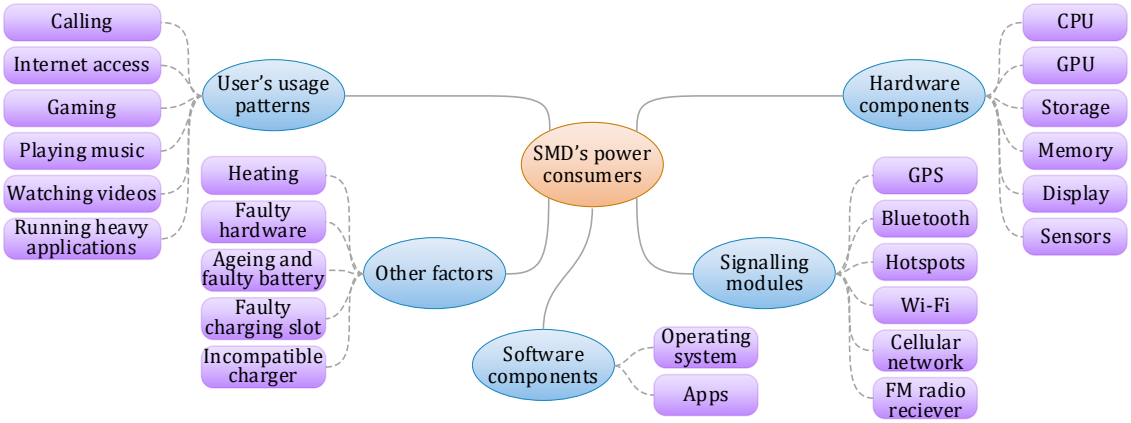


Fig. 3.15. Major power-consuming factors in SMDs

Table 3.7. Advancements in different aspects of SMD battery and charging

Focus area	Representative references
Decelerating discharge rate and stretching operational duration	[495] [496] [497]
Increasing battery capacity	[498] [499]
Shortening charging time	[500] [501] [502]
Increasing energy density	[503] [504] [505]
Extending battery lifespan	[506] [507]
Wireless charging	[508] [509] [510]
Energy harvesting and self-charging	[511] [512] [513]
Power sharing and crowd charging	[514] [515] [516]

3.6.1.2 Heat

"With great power comes great responsibility" - this saying may well be rephrased, in the context of SMD processors, as "with great power comes great heat". For example, NVIDIA's X1, one of the most powerful SoC, can easily overheat the device unless it is supported by a powerful cooling system [517]. Shoving excessive power into a tiny space without an adequate cooler or exhauster is the main cause of overheating mobile devices. Not only the CPU and GPU but the battery and the screen are also equally, if not more, responsible for overheating the SMDs. Some of the key factors responsible for heating the SMDs are listed in Fig. 3.16.

The heat generated by an SMD is largely proportional to the amount of electricity flowing through it, which directly depends on its workload. This suggests that users would tend to be impassive in engaging their SMDs for some additional work as it would make their device more hot. This would certainly hinder the success of MCC.

Fortunately, like batteries, there is a light of hope here also. Research efforts are being made to reduce the heating of the different components of the SMDs without

compromising the performance and quality. Table 3.8 lists some of the worthwhile research directions for tackling the heating issues of SMD. Developments are also witnessed in the commercial market. For example, to prevent heating, a new ICE 10.0 cooling system is incorporated in the Red Magic 7S[31] and Red Magic 7S Pro[32] gaming smartphones. The ICE 10.0 has a nine-layer cooling structure with a cooling material area of up to 43525 mm² and a 4124 mm² large VC cooling plate. The company claims that a temperature drop of 3°C can be achieved.

Gaming for long durations	• High-intensity gaming apps keeps the CPU and GPU highly active, which overheats the phone.
Running HD videos	• Watchng videos also puts enormous strain on the processors and keeps the screen on for long duration. This obviously generates heat.
Non-optimal settings	• Impropoer phone settings may cause inefficinet functioning and, in turn, overheating of the device.
Environmental factors	• Exposure to high temperatures (e.g., sunlight, nearby heat source, hot car dashboard, etc.) also causes the device to be heated.
Buggy apps	• Presence of a bug in an app may cause the phone to overheat by overusing the device's processor.
Not updating software	• The outdated system software or firmware which require updates, or a faulty/ incomplete update may cause malfunctioning, hence, heating the device.
Malware infection	• Malware often keeps the device hardware engaged and consumes RAM and CPU power highly, which causes the phone to overheat.
Faulty charger	• A faulty charger also causes a phone to be abnormally heated while charging.

Fig. 3.16. Causes of SMD overheating

Table 3.8. Research directions to mitigate heating issues of SMDs

Focus area		Representative references
Innovative thermal management	Body/encapsulating material	[518] [519]
	Mobile processors	[520]
	Multicore CPUs	[521] [522]
	Multicore SoCs	[523] [524]
	App-oriented	[525]
	Battery	[526] [527]
Efficient heat transferring		[528]
Alternative energy encoding		[529]
Using effective coolants		[530]

3.6.1.3 Network Connectivity and Bandwidth

Crowd computing, or any distributed system, demands stable and uninterrupted high bandwidth connectivity for smooth communication, low latency, good

[31] https://global.redmagic.gg/pages/redmagic-7
[32] https://global.redmagic.gg/pages/redmagic-7-pro

response time and throughput, and fault tolerance. Since, in most cases of MCC, communication would be through Wi-Fi, the bandwidth, speed, and traffic at a particular Wi-Fi access point play a crucial role. Unfortunately, Wi-Fi networks are disreputable for their instability. Many factors, such as multipath disturbances, degradation of the power signal, inter-cell hand-off, etc., are to blame [117]. A recent study has estimated that 45% of mobile devices experience failures while connecting to a corresponding Wi-Fi access point, and 15% of those that connect successfully require a substantial amount of time (5 seconds or more) to connect [531]. Even in the absence of connectivity issues, communication speed through Wi-Fi cannot match the bandwidth provided by typical wired connections. These shortcomings will certainly hamper the realization of MCC. Nevertheless, due to the sheer pressure of increasing mobile and portable devices, wireless technology is advancing continuously, which is established by introducing new Wi-Fi technologies like 802.11ah and 802.11ad. 802.11ah (also known as low power Wi-Fi) allows short and bursty data to travel long distances consuming much low power, whereas 802.11ad offers a very high data rate but for short-range communications [532]. Both protocols can be tactically used in MCC for different applications with varied computational data. Few tricks such as using a Wi-Fi booster extender and bigger antennas or strategically positioning the router may help in a better Wi-Fi experience.

The biggest hope in wireless communication technology is the Li-Fi, which can transfer 100 times faster than the present average Wi-Fi [533]. In lab conditions, it has attained an unbelievable speed of 224 Gbps, while in real-world trials carried out in offices and industrial environments, it achieved a substantial speed of 1 Gbps [534]. But it has its limitations, such that it can't work across walls, and for data communication, a Li-Fi-enabled device needs to have a direct line of sight to a functional light sensor. However, these limitations make Li-Fi a more secure medium than Wi-Fi and Bluetooth. Though there is little probability that Li-Fi will completely replace Wi-Fi in the near future, they could be used mutually to achieve more efficient and secure local communications. It is expected that future phones should come with a Li-Fi feature that would boost the MCC vision.

3.6.2 Major Challenges

MCC is descended from grid computing which in turn from distributed computing. Naturally, the issues involved in the ancestors have been inherited, perhaps even amplified, in the MCC system as well. In this section, we analyse the major challenges that are needed to be appropriately addressed for MCC to be successful.

3.6.2.1 Ensuring Security, Privacy, and Trust

Security, privacy and trust are very crucial issues for any crowdsourced systems [535] [536] [537]. Specifically, MCC's very dynamic and uncertain nature triggers real security concerns at both the service seeker's and provider's end. Allowing foreign programs from unacquainted sources to run on one's personal device is bound to be sceptical concerning privacy and security. Similarly, sending sensitive tasks to be operated at unknown entities involves high risk. Design flaws in the interaction between the MCC coordinator and the crowdworker may allow the attackers to access the coordinator and other crowdworker if the attacker can breach the client application on any SMD. Similarly, if the attacker can breach the security at the coordinator, a faulty design may compromise all the crowdworkers connected to the coordinator. Different threats potentially involved with MCC are discussed in the following.

- **Threats from MCC:** This is the most concerning perception that thwarts crowdworker participation. The most common and probably the only fear from the end of MCC coordinator is:
    - **Access to private data on SMD:** An ill-conceived MCC application can become a potential avenue for an intruder to access personal information from an SMD. A compromised coordinator can access the stored passwords, private data inputs, and other personal information in the SMD, which might be severely risky.
- **Threats from crowdworker:** Typically, the threats from the crowdworkers are not acknowledged. However, from an MCC system perspective, they can be very severe as they are capable of disrupting the whole system. Some of such threats are:

- **Malicious content injection:** An unscrupulous crowdworker or an intruder may inject malicious content like corrupt files, media content, scripts, etc., to contaminate the MCC server and destabilize the whole MCC system.
- **Attack on MCC client application:** An infected application caused by either malicious script or corrupt content injection to the SMD can potentially paralyze the functioning of the MCC client application, making the application crash or behave unexpectedly.
- **Repudiation issues:** An impostor may attempt to repudiate a signature itself and pose as a legitimate crowdworker to get connected to the MCC system. The repudiation threat may also come from a misplaced SMD that has been seized by a fraudulent user. The issue of unauthorised crowdworker can be mitigated by using digital certificates, which would assert the authenticity of the crowdworker. However, with mobile devices, it is not uncommon for private keys to get leaked, leading to digital forgery.
- **System integrity:** It may also happen that an unscrupulous SMD user manipulates the results with ill intention or tries to disrupt the overall system. The MCC should shield itself from these kinds of threats. One solution is to adopt suitable trust and reputation-building policies and mechanisms [538]. There are other general approaches to maintaining crowd computing systems' integrity, as mentioned in Table 3.9.

- **Threats from network or third party:** Another aspect of concern is that the security is not that strong in wireless protocols compared to wired network protocols. Wireless networks are more vulnerable to attacks like eavesdropping, tracing, spoofing, camouflaging, tampering with data, and others [539] [540]. Though it is a tough ask to vanquish people in the network with evil intents continually, all crowd computing projects should be able to eliminate the apparent risks. Existing network layer (e.g., IPsec) and transport layer (e.g., SSL, TLS, SSH) protocols can provide decent end-to-end security. Many wired security solutions (e.g., WTLS) are also adapted to support wireless networks. Several other regular security measures for the wireless network and mobile devices (e.g., WPA2, 802.11i security, wireless IPS, etc.) can be adapted to

secure MCC. In the context of MCC, the most severe issues from the networking perspective are:

- **Data tampering:** Preventing data tampering in networked systems is always challenging. This can potentially be unsafe if public Wi-Fi or mobile data networks are used for MCC operations (job distribution and result collection). A man-in-the-middle attack is potentially possible, which can manipulate the transmitted data while in transit, obliquely affecting MCC's stability and reliability.
- **DoS attack:** Like other internetworked systems, a global MCC can be susceptible to DoS attacks. However, this is not so perilous for the local and ad-hoc MCCs.

Table 3.9. General approaches to mitigate system integrity in crowd computing **[83]**

Solution	Mechanism	Remark	References
Voting	Each microtask is sent to multiple crowdworkers, and the best result is selected through voting. Two approaches can be adopted: • Majority: The result returned by the most crowdworker is selected as final. • *m*-first: The particular task is continuously executed by different crowdworkers until a threshold (*m*) is reached for the first similar results.	It involves redundant computing, which leads to resource wastage.	[541] [542]
Spot-checking	The MCC coordinator sends a task of which the result is already known to a crowdworker to randomly crosscheck the result returned by the crowdworker. If the result does not match, the crowdworker is tagged as a saboteur and blacklisted for future task assignments.	All the crowdworkers need to be cross-checked for best effects and probably multiple times, which is not very practical.	[543]
Credibility-based voting	The credibility is calculated as the conditional probability of the correctness of the result of a task which is evaluated against a threshold value. If the credibility is lower than the threshold, the task is reassigned to some other crowdworker.	It inherently guarantees that the overall error rate of the MCC system will not exceed a certain limit.	[544]

Encryption and cryptography can solve the problem of privacy and security threat in public and open network communication. But running cryptography and authentication processes on SMDs would be a bit heavier than on desktop computers. Volunteer projects based on BOINC prevent hackers from distributing malware even after breaking into the server by employing code signing.

Since the apprehension of security is often associated with trust, it is essential to establish trust between volunteers and the client project to make crowd computing successful. To check on corrupt crowdworkers, a reputation-based trust [545] scheme can be adopted. To maintain trust, resource seekers and providers should know each other. This can be achieved by pre-registering to a common platform where both parties can check each other. A crowd computing project should be able to earn the trust of a contributor on different accounts:

- The project should not harm or corrupt the contributing SMD by any means.
- The client application should not invade the host's privacy. In other words, the software can't read, access, or share an SMD's personal files.
- The project should adhere to reliable security measures so that contributors' SMDs are not affected caused by any malicious activity by hackers.
- The project should take the contributor into confidence about the reliability and legality of the activities being executed by its applications.
- The project should unambiguously declare the purpose and scope of the project and how the results will be used to attain that.
- The project also should explicitly state the capacity of the resulting intellectual property and how it will be exercised.

Researchers are coming up with novel and interesting ideas and solution approaches to mitigate the above-mentioned issues [145] [546]. Recently, Blockchain-based solutions are popularly being used to mitigate security and privacy issues in crowdsourced systems [547] [548] [549]. Especially, the inherent decentralised security of Blockchain technologies has encouraged researchers to successfully apply it in safeguarding crowdsourcing applications [550] [551]. Researchers have explored applications of Blockchain to enforce trust and reputation [552] [553] [554] [555]. Table 3.10 lists some of the representative research works addressing the security, privacy and trust in crowdsourced systems. Though the mentioned works intended to focus on different crowdsourcing applications with different solution approaches, we believe most of them would also be applicable to MCC.

In summary, the MCC system should ensure that neither the MCC job submitted to an SMD should interfere with local files and data, nor the host system (SMD)

should be able to maltreat the guest processes. The user's personal data on the phone should be transparent to the guest program and its source. The MCC client application should guarantee that the security and privacy of the crowdworker are strictly maintained. In fact, the MCC application should not be able to access the user data by any means.

Table 3.10. Research attempts to mitigate the issues of security, privacy and trust in crowdsourced systems

	Focus area	Target problem	References
Security	Mitigating repudiation issues	Counter measuring false-name attacks	[556] [557]
	Crowd computing system security	Malware identification	[546]
		Mitigating colluding attacks	[558]
Privacy	Privacy preservation in crowdsourced applications	Anonymous crowdworker selection	[559]
		Task assignment	[560] [561]
		Task recommendation	[562]
		Location secrecy	[563] [564] [565]
		Handling crowdworker unavailability	[566]
		Transaction privacy and transparency	[567]
	Code and data privacy vulnerabilities in crowdsourcing	Detecting and solving code and user identity privacy issues	[568]
		User identity and data privacy protection	[569] [570]
Trust and reliability	Crowdworker recruitment and task assignment	Secure task recommendation	[571]
		Trust-based task assignment	[572]
		Trust-based crowdworker recruitment	[573] [574]
		Reliability-driven task assignment	[575]
	Crowdsourced data validation	Reliable data analysis in the mobile ad-hoc cloud	[547]
		Ensuring crowd task's reliability and evaluating crowdworker's data quality	[576]
	Trust management for crowdsourced services	Assessing trust value/worthiness of crowdsourced services	[577] [557] [578] [548]
		Multi-perspective trust management	[579]
		Trust establishment	[580]
		Anonymous authentication on the trust	[581]
		Improving trustworthiness	[582] [559]
		Trusted crowdsourced servicing and payment system	[583]
	Trust-based consensus for crowdsourced services	Automated agreement on crowdworkers' credibility	[584]

As a crowdworker, one should be careful of joining an MCC project. Similar to the usual precautionary measure while accessing the Web, the authenticity and legitimacy of the project source should be validated before downloading the client application for the project. Before starting any business, digital signatures, and digital certificates should be used to authenticate each other.

3.6.2.2 Motivating People to Participate in MCC

The success of crowd computing is absolutely dependent on people's willingness to participate and donate their devices' resources. Most often, due to various reasons, the users can get reluctant to lend their devices as crowdworker [585] [586]. To attain acceptable QoS, a sufficient number of active crowdworkers with stable behaviour need to be maintained, which is challenging to attain. For successful implementation of MCC, as many as competent SMDs need to be attracted, for which the users need to be adequately motivated. Also, not only attracting new crowdworker but also retaining the existing crowdworkers is also equally important.

In volunteer computing, resources are not bought but earned [587]. A volunteered MCC project banks on its public appeal to get volunteers. A research project that has excellent public appeal can get enormous computing power through volunteering. It is also the responsibility of the people associated with the project to persuade and convince the public that their computing resources are being used for a greater cause and are significantly beneficial to society. If MCC needs to be adopted in a wide range of applications, it needs to be popularised and draw all kinds of people from every sector of society and motivate them to lend their devices. People may be tempted to volunteering their computing resources for various motivations such as:

- Many people enjoy a sense of gratification in being a part of citizen science, contributing to research in science, humanities and other greater causes.
- Many crowd computing forums encourage users to share ideas and information in the forum. People feel proud of connecting to a noble online community with shared pursuits.
- Some projects allow crowdworkers to be involved in the project in many aspects, including programming, testing, support, documentation, and even software development via different online modes. In that case, crowdworkers can cultivate skills and gain technical knowledge, which is a great motivation factor for IT-savvy enthusiasts.
- Sometimes involvement of the crowdworkers in the various projects creates

chances for discoveries that lead them to fame.

- Some crowdworkers revel in healthy competition in chalking up the highest processing time for a particular project.

3.6.2.3 Framing Sustainable Economic Model

Crowd computing need not necessarily be driven by free volunteering only. In fact, if MCC is based on free volunteering, then it will not be easy to realise. Availability of resources is entirely up to the volunteers' will. In practicality, the motivational factors mentioned in the previous section have minimal effect. For a more realistic MCC implementation, appropriate reward and incentive mechanisms need to be adopted [588] [589]. Establishments running MCC projects can pull off an individual's SMD resources by presenting something in return. For example, a shopping mall can offer shopping credit values to the visitors for using their SMD's computing cycles. People may also be interested in offering their resources where there is a scope of reciprocal allowances. For example, organizations may allow people to use their Wi-Fi at no cost in return for the user's SMD resources. Authorities can persuasively acquire resources. For example, a firm that hands out free SMDs to its employees can make it obligatory to make their SMDs accessible for crowd computing projects of the organization.

Some people see it as a lucrative option to lend their resources for commercial purposes if they get fair reimbursement in return. The recompense amount can be calculated on every unit of time the resource is utilized [590]. A couple of such initiatives have already hit the market. Zennet[33], a public, distributed, and decentralized supercomputer, offers an open commercial market platform for computation power trading. Anybody can offer his/her hardware for sale, and anybody who needs computation power to run arbitrary computational tasks can rent these offers. Neocortix[34] has developed a viable shared-economy MCC business model in which they rent computer time on public phones and sell the aggregated computing capacity via Neocortix Cloud Services. The smartphone owners get paid for

[33] http://www.zennet.sc/about/index.html
[34] https://neocortix.com/phone-paycheck

renting out their phones' computing capacity. Neocortix claims that in return for 8 hours a day of service a user can earn up to $80 a year, and if the user can offer her spare phone for 24 hours, the earning can be up to $240 a year. The number of phones per user is limited to five. Users can redeem their earnings through a PayPal account. Smartphones are categorised as per their computing abilities. For example, devices like Galaxy S9, S10, or S20, are tagged as 'gold device performance category'. Earning rate depends on the phone model. To participate in this business model, the phones should meet certain requirements[35] and obey some restrictions, such as emulated and rooted devices are not permitted.

To attract mass crowdworker and retain them for the long-term, a sustainable business model that would be beneficial for both sides (MCC aggregator and crowdworker) need to be framed and implemented [82]. Various formulas and techniques for incentivising and pricing crowdsourced services are proposed in the literature. The most common techniques and approaches, along with some representative references indicating their applications in crowdsourced systems, are listed in Table 3.11. Furthermore, different criteria such as truthful [591] [592] [593] [594], reputation [595] [596] [597], quality [598] [599] [600], etc. can also be applicable as incentivising policy. Adopting the right incentivising or pricing technique and criteria would depend on the application requirement, users' behaviour or preferences and organisational strategy.

Table 3.11. Common incentive mechanism techniques for crowdsourcing

Incentivising and pricing technique/approach	Reference
Game theory	[601] [602] [603]
Stackelberg theory	[604] [605] [606]
Contract theory	[607] [608]
Tournament model	[609] [610]
Auction theory	[611] [612] [613]
Reverse auction	[614] [615] [616]
Rewarding	[617] [618]
Cooperative incentive	[619] [620]
Dynamic pricing	[621] [622] [623]

[35] https://neocortix.com/device-requirements

3.7 Potential Applications of MCC

Crowd computing is an effective resolution to fulfil the huge computing requirements of those who are unable to afford expensive HPC systems. Likewise, MCC can provide an inexpensive means to carry out compute-intensive tasks. Not limiting to large-scale scientific computing, this pragmatic concept can be utilized in many other real-life applications. The ability to realise an HPC facility on an ad-hoc basis as per requirement widens the application horizon of MCC. In the following, we explore some interesting application areas of MCC.

Computing facility for research in universities/research institutions: Universities and research institutions can frame MCC facilities by combining the SMDs of students, researchers, faculties, and staff. This setup will be used to meet the computing requirements of the research problems carried out within the campus. Enormous computing power can be achieved if MCC is coupled with the campus-wide grid encompassing in-campus PCs (lab computers, desktops, and laptops belonging to faculty, staff, and students). This massive computing facility will entice potential faculty and researchers, empowering them to realize computing resource-demanding research works [624].

Organizational computing: Organisations spend a large share of expenditure on IT infrastructure. Lately, cloud computing has emerged as a cost-effective solution for data storage and processing. But as cloud computing need to be subscribed, it is not a wise option for those organisations which have extensive and 24/7 requirement for high-performance computing. For SMBs, although cloud can provide a significant cost reduction it still involves a considerable amount of money. In a large business's case, the savings' margin scale is not that exciting. Organisations can go for setting up private clouds for cost minimization. But again, this will involve certain upfront infrastructure costs as well as regular operational and maintenance costs. Older organisations can utilize their existing desktop PCs and build an organizational grid of computing resources. But as we have discussed earlier, PCs' popularity is declining rapidly, and business organizations can exploit employees' SMDs for MCC, not spending on excessive IT infrastructure budget, for processing and analysing business and financial big data. Many organisations

adopt the BYOD policy nowadays, which permits and often obliges employees to use their own SMDs for carrying out company business. This allows the organisation to have the following benefits:

- Significant cost savings.
- Flexibility of doing the office work regardless of location, device, or time of day.
- Ease of use that BYOD provides.

Computing resource for remote research lab: MCC can be very handy to carry out computing-intensive field research at isolated and remote locations where no wide-area communication is available. SMD's Wi-Fi hotspot facility will be used to construct an MCC environment.

Train/transport navigation and collision avoidance: Nowadays, trains (usually high-speed and premier trains) are getting equipped with a Wi-Fi facility with adequate charging points for the passengers. Passenger's SMDs can be exploited to establish an ad-hoc computing facility that will be required for effectively implementing collision avoidance methods like RCAS[36], TCAS[37] and others [625] [626]. These systems use multi-sensor navigation, and integrate and calculate the sensed data with complex algorithms for data fusion and situation analysis, which require considerable computing resources. The ad-hoc computing system can also be utilized for other complex calculations like forecasting the local weather. The same idea can be employed on buses and metro trains having the same facilities. That will yield a very lucrative opportunity for MCC in a crowded country or city.

Aircraft control and navigation: Many airlines offer in-flight Wi-Fi facilities to the flyers [627] [628]. Out of them, some provide this facility without any extra charge [629]. In return, these airlines can use passengers' SMDs to process real-time flight data. Though connections are often slow and unreliable, we can expect faster and more reliable connections in the near future, thanks to the ongoing development of the core technologies behind in-flight Wi-Fi [630]. Availability of the

[36] http://www.collision-avoidance.org/rcas/
[37] http://www.hbl.in/product-view-52-engineering-solutions-railways-train-collision-avoidance-system.html

passenger's SMDs is guaranteed during the flight (provided the battery is not a hurdle). Typically, a jet aircraft is fitted with nearly 5000 sensors that produce nearly 10 GB of data per second [631]. A 12-hour flight would averagely, produce 844 TB of data. Processing this huge amount of data would require a good amount of on-board computing. Transmitting this data to the cloud for processing and obtaining the response back would waste time and is a bandwidth constraint. A local MCC utilising the passenger's SMDs would be an effective solution to process the in-flight data in near real-time while saving time, bandwidth, and the requirement of installing an HPC in the aircraft.

Field data processing: Collection and processing of open field data, e.g., agricultural land, river basin, forest, desert, barren land, hills and mountain, etc., become challenging due to non-existing computing infrastructure and network [105]. An ad-hoc MCC would provide the required computing facility and be a handful in collecting those data distributedly with a slight improvisation in the MCC client application.

On-field military data processing: Owing to the technical advancements, soldiers are equipped with multi-sensor devices. Sensors are also installed in the surroundings of their base camps and outposts. These sensors generate a great amount of data, some of them being real-time data, which require secured HPC systems to be processed and analysed. If the soldiers are furnished with high-end SMDs, then an insular MCC can be materialized, which will be secured because it will require only a low-range communication network, not a WAN. Furthermore, in war situations, it is essential to monitor and assess a soldier's psychological and physical health. Also, injured soldiers on the battlefield need to be continuously monitored for their vitals and physical state to take appropriate medical action. The biomedical non-invasive sensors help in such situations by assessing the heartbeat, heart condition, blood flow and pressure, oxygen level, and body sugar. These sensors produce and transmit a huge amount of data, which need to be processed and analysed as per real-time health monitoring models. However, this involves a good amount of computation which is difficult to have in a computer and network infrastructure-less scenario. An ad-hoc MCC would be helpful utilising the SMDs

that the soldiers carry to attain an aggregated computing service.

Disaster management: In the case of large-scale disasters, communication is generally severely affected, due to which cloud and remote servers become inaccessible. In these cases, MCC can be very handy to form an ad-hoc computing grid using SMD's NFC options such as hotspots, Bluetooth, etc., which can be used to analyse the image, audio, and video data to assess and manage the situation by helping in locating and rescuing the victims.

Video and image analysis: Image and video processing, editing, rendering, and analysis require hefty computation. Instead of going for high-end computers and cloud services, MCC can be utilised. Using MCC for video and image applications will cut costs and can be availed at any time and place. Local MCC can also be used for collaborative image [632] and video [75] processing and analysis.

Biometric verification: For security measures, personal identification and verification through biometrics like fingerprints, palmprints, eye scans, and face detection has been very effective and popularised recently. Biometric verification needs a good amount of computing resources. An MCC is a suitable solution to increase the speed and flexibility of data processing as it has two-fold advantages. Besides saving energy and resources, distributed processing in MCC would add more security [105]. In addition to regular in-campus systems, this will be very much applicable for occasional large gatherings where installing permanent computing is not viable. Attendees' own SMDs can be utilised to set up a temporary MCC.

Distributed key generation and agreement: In cryptography, private and public keys are used for secure communication. Normally, these keys are generated by the communicating parties. In a distributed key generation, these keys are generated by multiple parties. This eliminates the possibility of accidental exposure or misplacement of the keys by precluding the access of the private key by a single party. Distributed key generation ensures secrecy in the presence of malicious contributions to the key calculation [633]. MCC can be useful for computing such distributed keys. MCC can also be useful in distributed key agreement in securing crucial files in an organisation by distributing the access key over different trusted SMDs within the organisation [105].

Cooperative encryption cracking: Encryption cracking (e.g., WPA/WPA2, WEP, SHA256/512, DES, MD5, Blowfish hash, etc.) involved in brute-force-based intrusion is a high-intensity computing task which might take weeks to months and even years depending on the computer's capacity [634]. To crack within an agreeable time, a cluster of high-end computers is usually used where the tasks are executed in a distributed fashion [635]. In an ethical hacking scenario, the power of MCC can be harnessed for this purpose [93].

Business processing: Business processing involves office management, ERP, statistical data analysis, and decision making. Office management involves employee, email, and other business data management. The ERP includes optimizing business operations and processes, which enhances business output. Likewise, statistical analysis of business includes finance, market, inventory and other data analysis for better decision making. The use of AI and computation automatizes business processing. Due to the huge amount of data and, accordingly, requirement for massive computing resources, most business houses invest a lot either in in-house computing infrastructure or getting cloud services. Businesses with large and medium offices having a number of employees can save a lot by exploiting the available in-house SMDs to form an in-campus MCC and carry out the business processing tasks in real-time.

Enhancing business and sales in shopping malls: Shopping malls encounter a large number of footfalls every day. Every customer has different shopping behaviour and inclination. A smart store can make its servicing smart by analysing and predicting customer shopping preferences and purchase patterns, organising the store, and planning its business accordingly. Along with, heat maps on customer foot traffic, peak period, customer movements, dwell time, etc. can also be very useful to take various decisions like an advertisement of products, availability of products, the display positions of the products, arranging product for a controlling and channelise customer movement and increase sales. All these analyses require substantial computing resources. It is preferred to have HPC within the mall premises for real-time analytics rather than going for the cloud. MCC can be an ideal option, taking advantage of a large number of customers' SMDs available on the

premise along with the permanently available SMDs of to the mall staff. Most of the customers tend to spend a decent amount of time inside. However, their hanging around stint inside the premise for a certain duration is neither guaranteed nor can be predicted precisely. Hence the customer's SMDs may be engaged in fully opportunistic MCC, whereas the mall staff's SMDs can contribute to a quasi-opportunistic MCC. Big shopping malls also often feature multiplexes where the availability of resources for a fixed period is more or less guaranteed. That gives an excellent opportunity for quasi-opportunistic MCC. The vendors, retailers, or stores can make use of MCC for the computing requirement in increasing their business by performing real-time data analytics and taking informed and better decisions.

MCC-based edge computing for processing IoT data: One of the most notable practical applications of MCC is to be considered edge computing for processing IoT data. The majority of the IoT applications are time-bound. Due to transmission latency, cloud computing is not favourable. Edge computing provides a computing facility near the data source [74] [636]. A local MCC will be a suitable edge computing solution for processing organizational and industrial real-time IoT data, saving a lot on proprietary edge solutions [143]. In practice, there are different forms of edge computing, such as fog computing, cloudlets and MEC [637]. A detailed discussion on MCC-edge is presented in Chapter 9.

Smart building and smart city: Smart buildings and smart cities bring various kinds of automation to make people's life easier. To achieve this, intelligent decisions are made based on the sensed data. Thousands of sensors are employed, which produces enormous data which need to be processed and analysed to make impromptu and smart decisions to take effective actions. For processing and analysing this huge data, a local HPC or cloud services are required. Using localised HPC systems would increase cost, overhead, and carbon footprint, whereas using remote cloud services not only requires enough bandwidth for data transmission but also computing data far from the point of source and sink delays the automated action, nullifying its smartness. Utilizing MCC would help in overcoming these issues. The SMDs available with people in buildings or open areas (roads, parks,

grounds) can be utilised to set up a local MCC for processing the data generated from the nearby data sources.

Public security and policing: Nowadays, in many public places, such as bus stands, railway stations, etc., criminal face recogniser applications are installed. Considering the probability of a criminal fleeing by boarding a bus or a train, a real-time face recognition process runs to search for a probable match of the criminal with the faces captured by the CCTVs installed in that area. Since there will be abundant SMDs available in these crowded public places, MCC will be a feasible and better alternative to cloud computing in terms of the latency that is crucial for real-time applications.

3.8 Limitations and Further Scopes

In this chapter, we concentrated more on the design considerations of MCC. We did not cover the development, deployment and operational aspects in length. It would have been complete by including a thorough exploration and assessment on the required and suitable tools, languages and libraries for developing a generalised, platform-independent (including mobile hardware and OS) and lightweight MCC. In future, we wish to come up with an article on this.

3.9 Summary

In this chapter, we presented an in-depth study on the potential and feasibility of achieving computing power by utilising the public's (crowd's) SMDs. Since in this computing paradigm, a crowd of SMD's resources is collectively used to provide computing resources, it is called mobile crowd computing (MCC). MCC gives us an economical and sustainable alternative to other HPC systems such as grid, cloud, clusters and supercomputers to carry out compute-intensive tasks. Thanks to the ubiquity and dense availability of the SMDs, an ad-hoc HPC facility can be provided, leading to attaining a truly ubiquitous HPC.

Besides the advancements of SMD hardware and its wide adoption, MCC has been fuelled by several other factors such as denser Wi-Fi zones, low-cost and highspeed mobile data, energy-efficient and highspeed short-range communication technologies, etc.

MCC is not an out-of-the-blue concept. It stands on several other established computing paradigms such as distributed computing, parallel computing, grid computing, volunteer and crowd computing, and opportunistic computing, to name a few. However, there are certain distinctions between MCC and these foundation computing systems. We also saw how MCC differs from other similar mobile computing systems such as mobile grid, mobile cloud, ad-hoc mobile cloud and mobile crowdsourcing.

Primarily, a typical MCC can either follow a centralised or a P2P architecture. However, hybrid architectures can also be adopted for a large and complex MCC. Also, depending on the infrastructure and the deployed application, MCC can either be global (connected through the internet), local (connected through WLAN), or ad-hoc (connected through Bluetooth, hotspots, etc.). Furthermore, an MCC can be purely opportunistic or quasi-opportunistic in utilising the crowd resources.

For a successful and efficient MCC design, several aspects need to be considered and tackled properly. The hardware, software and network heterogeneities need to be abstracted for better interoperability. Ideally, an MCC platform should be generalised to support any MCC application irrespective of its type and requirements. It should be adaptable to various internal and external changes without affecting the system's normal functionalities. Being a networked distributed system, in MCC reliability, fault tolerance, and QoS need to be ensured. Also, an MCC system should be scalable and elastic to dynamically acclimate to the resource requirements. Other design goals include user-friendliness and non-intrusiveness, i.e., the usual operations of the crowdworkers should not be hampered anyway. Bearing the energy limitations of the SMDs in mind, executing the microtasks should be energy efficient. For a commercial MCC, a well-defined SLA should cover the liabilities and legalities. A suitable architectural model should be adopted based on different aspects for effective deployment. Since the spine of MCC is the crowdworker base, they need to be managed very efficiently. Management of crowdworkers includes discovery, profiling, selection, ensuring availability, and monitoring. A large MCC task is divided into multiple microtasks that are sent to the crowdworker for execution. The task farming process should be efficient

because they are created in such a way that they can be executed by the available crowdworker resources and as far as possible independently. Scheduling these microtasks to the appropriate crowdworker is another crucial aspect because an MCC's overall performance depends on it greatly. While scheduling, various factors should be considered, such as best resource utilization, energy efficiency, load balance and fairness, and dynamicity. If an MCC task involves multiple workflows, it should be handled appropriately, keeping in mind that the crowdworker are not static and dedicated resources. Another important consideration should be to verify and aggregate the results received from the crowdworkers. The correctness and usefulness of the final outcome would very much depend on this.

MCC has several advantages to offer. Cost-effective, least overhead in IT infrastructure management, DDoS attack-free, offering computation offloading, scalability and agility, and amply available resources are a few of them. Additionally, local MCC offers on-premises computing service, which brings on added advantages of lower latency and minimized network cost and congestion. Besides these, two major advantages of MCC are ubiquity and pervasiveness and sustainability.

However, realising MCC is not challenge-free. Two major concerns from user points of view are battery constraint of the SMD and getting the SMDs heated due to executing heavy processing tasks for a long duration. Recent studies console us that these factors will not be much worrisome in the near future. Additionally, since MCC is a networked system, its performance very much depends on the data transfer rate. Since the operation and quality of the network are not always guaranteed, it remains a challenge to overcome it. Ensuring security, privacy, and trust always remains one of the most challenging factors for any networked system. MCC is no exception. Perhaps in MCC, the privacy issue is far more crucial than in other systems because here, the SMDs that carry out the foreign tasks are very much personal to their users. Another hurdle in implementing MCC is to motivate the public to lend their SMDs. They need to be sufficiently motivated to draw into participation in MCC. One straight option might be offering them lucrative financial or other financial incentives. Sustainable economic models need to be framed for this, which would depend on the MCC application type and the user base.

Intensive research is going on in all the above-mentioned areas, and many innovative and feasible solutions are coming out. Therefore, we believe these challenges will not seriously threaten realising MCC.

Appreciating the potential of MCC, it can be leveraged for many real-world applications. For organisational and scientific computing, MCC can be a suitable HPC alternative to costly supercomputers and cloud services. The HPC capability of MCC can also be utilised in the education and healthcare sectors. Even small and medium businesses can utilise MCC to meet their daily computing demands, including for data analysis and predictions. An ad-hoc MCC can be useful for infrastructure-less use case scenarios such as disaster management, military base and war fields, and many more. Local MCC can be a feasible alternative to commercial edge computing.

4
Resource Profiling in MCC

"It's important to determine which surroundings work best for you, and then build that environment to suit your needs." --- Marilu Henner

4.1 Introduction

The efficacy of MCC largely depends on the capability and reliability of the incorporated SMDs. It is apparent that SMDs with superior computing resources would offer better throughput. Therefore, to achieve better performance (e.g., maximum throughput and minimum turnaround time) of an MCC, it is important to schedule the MCC jobs to the most potential SMDs available in the network. The first step to achieve this is to recognise the presently available resources and assess them meritoriously.

To evaluate the capability of an SMD, it is imperative to estimate the capacity and the present usability of its resource parameters such as CPU, GPU, battery, etc. Here, by the capacity of an SMD's resource parameters, we mean a) the fixed parameters such as the clock frequencies and no. of cores in the CPU and GPU, RAM size, battery capacity, etc. and b) the present usability that denotes the present status of variable parameters such as present CPU and GPU load, available RAM and battery, device temperature, signal strength, etc. The fixed parameters never change their values, but the values of the variable parameters change dynamically.

Considering the heterogeneity and dynamicity of the resource, the selection of the best resources might be inaccurate in the absence of a proper resource assessment machinery. And to assess the resources correctly, they need to be profiled efficiently. For instance, a suitable logger program might be helpful, which would assess the resource parameters of the SMDs correctly and return their persistent and instantaneous values whenever asked by the MCC coordinator for job scheduling. For accurate resource assessment, precise values of all these parameters are needed to be profiled, which is not trivial because capturing and storing these parameters' values require different policies and implementations. For this, it is important to

have a structured framework and policy for resource profiling to appropriately and applicably profile the SMDs' various resources that are significant for an SMD to be considered as a computing device in MCC.

In this chapter, we aim to achieve the followings:

- A methodological approach for profiling various resource parameters that would be necessary for selecting the most potent SMD as computing resource.
- Ascertain and classify the considered resource parameters with apposite reasoning.
- Apply a benchmarking scheme to assess the actual performance of the SMDs, in addition to other specifications.
- Design a prototype application that incorporates the resource profiling and selection modules of an MCC.

Fig. 4.1. summarises the resource profiling and selection process, covered in this chapter.

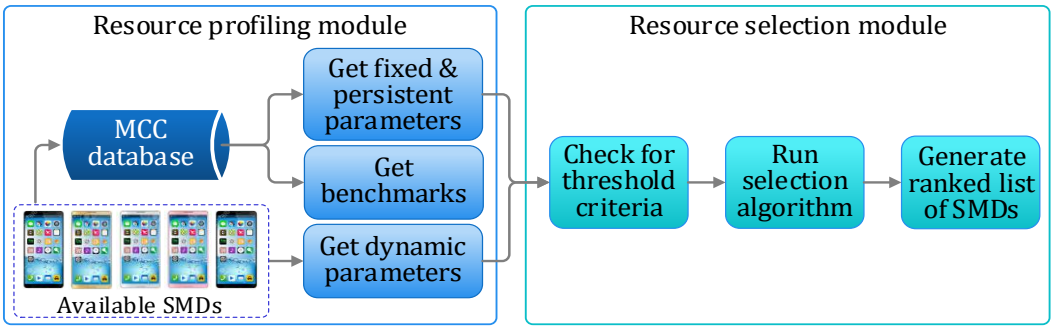


Fig. 4.1. The pictorial summary of resource profiling and selection

4.2 System Model and Hypotheses

Here, we considered a local MCC scenario in which the SMDs are connected to the MCC coordinator through a Wi-Fi network [36]. Within a particular Wi-Fi network, there is a single coordinator that manages the crowdworkers connected to that Wi-Fi AP. SMDs that are connected to the MCC coordinator and are willing to share their device resources are considered as crowdworkers.

An MCC client is installed on each SMD that is supposed to take part in MCC. The MCC client service estimates the computational capability (e.g., CPU and GPU power, etc.) and other vital statistics (RAM and battery capacity, etc.) of the SMD and shares them with the coordinator. It also assesses the present resource usage

status of the SMD when requested by the coordinator. The client application on the SMD manages the execution by CPU stealing and, upon completion, sends the result to the server. The coordinator assembles the results received from different clients.

There is a middleware for job management. The coordinator hosts the MCC middleware, which manages the operations of MCC that includes searching and selecting SMDs, job scheduling, result collecting, and fault handling. The middleware is also responsible for a few other tasks such as SMD profiling, execution time estimation, resource presence time prediction, and so on.

We assume that all SMDs, which have the MCC client installed, are crowdworker and willing to share their resources, either voluntarily or in return for some incentives profit or non-profit basis. We also assume that each crowdworker completes the assigned subtask within a finite time and sends back the results before leaving the MCC network. Fig. 4.2 lists the general components of a typical local MCC system, while Fig. 4.3 presents an abstract model of a local MCC.

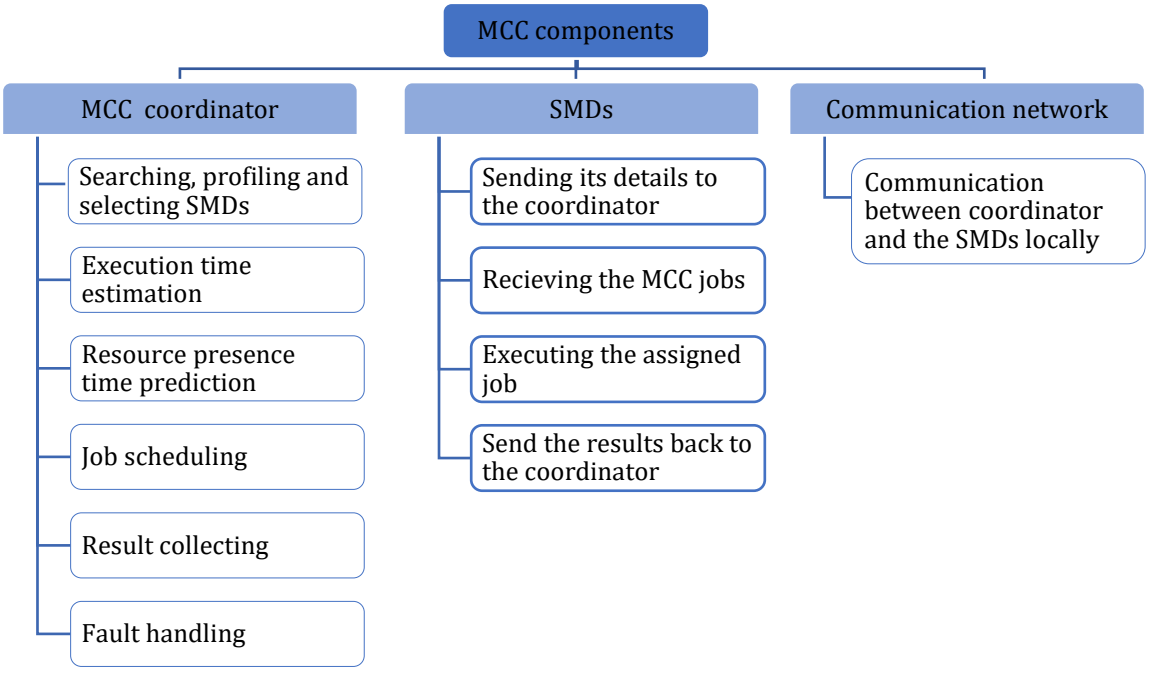


Fig. 4.2. General components of a local MCC

4.3 Resource Profiling and Assessment

To assess the goodness of an SMD as a computing resource, the most straightforward approach is to check its hardware specification. However, this approach

would be insufficient to assess the overall competency of an SMD, as it may not reflect the actual usable status of its resources. Furthermore, an SMD may seem fittest in the present context, but it may happen that this fitness may not be consistent over a longer period. Therefore, we need to consider the consistent behaviour of the SMDs to truly assess their suitability for selection.

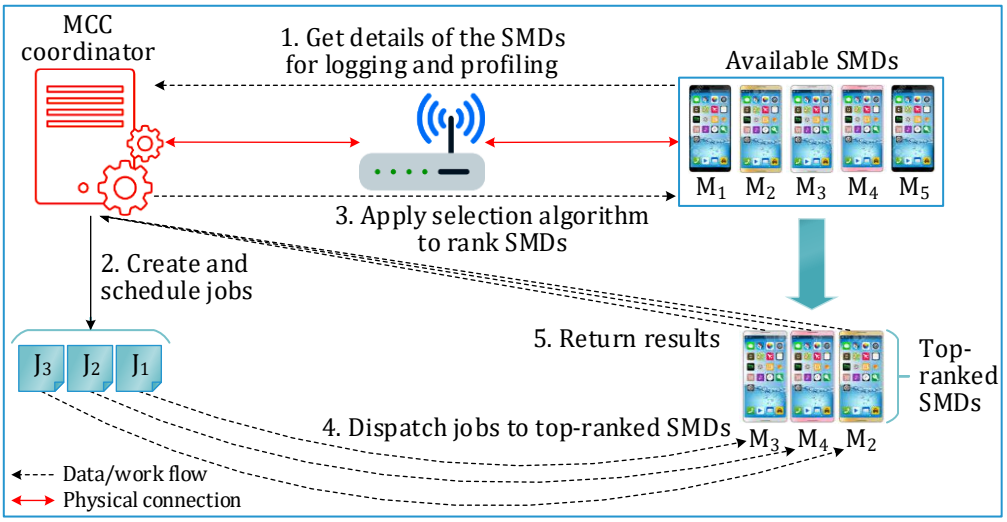


Fig. 4.3. An abstract model of a local MCC

To accurately assess the capabilities of the SMDs as potential crowdworkers in an MCC, we considered several parameters. Some of these parameters are read on the fly, while some are retrieved and calculated from the logged data. We characterized these parameters into three categories, as discussed in the following subsections. Fig. 4.4 present the taxonomy of the considered parameters for resource profiling.

4.3.1 Fixed Parameters

These information are fixed for any particular SMD and typically never change in its usable lifetime. That is why these parameters were logged only once, i.e., when an SMD connects to the MCC for the first time and were permanently stored in a database. For the subsequent connections of the re-entrant SMDs, profiling of these parameters is not required. The following parameters fall in this category:

UID: Though this parameter is not considered as a resource parameter, it is used to identify the SMDs uniquely in the database. In real SMDs, each device can be recognized uniquely using some identifiers such as Android ID (for Android phones), MAC, IMEI, etc. Among these, IMEI is simple and easy to implement and serves our purpose well. Therefore, we considered the IMEI number of the SMDs

as the UID (unique identity) to identify the SMDs.

Maximum CPU clock frequency: The capability of a CPU largely depends on its clock frequency. Though in a multicore CPU, the cores might have different clock frequencies, we followed the general convention, i.e., considering the highest frequency.

Number of CPU cores: Though the CPUs in SMDs are optimised for serial operations, the efficiency of an SMD CPU is supposed to increase with the number of cores in it, which can execute multiple ALU operations in parallel.

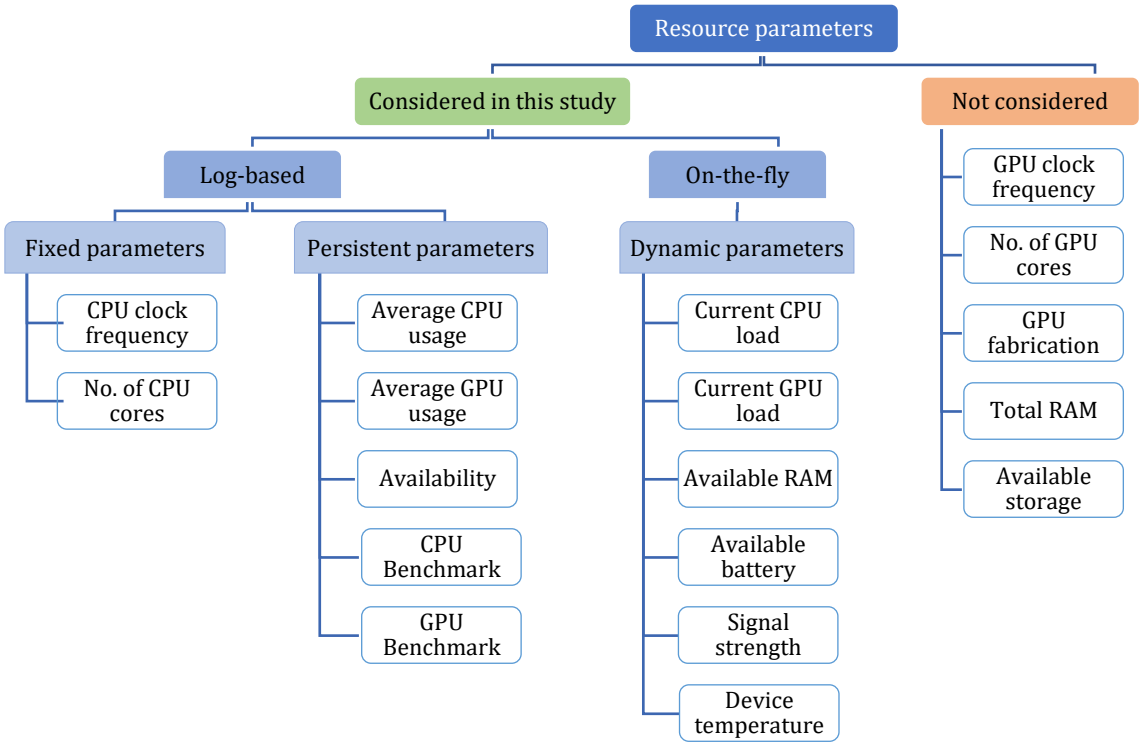


Fig. 4.4. Parameters considered for resource profiling

4.3.2 Dynamic Parameters

The values of these parameters in an SMD vary dynamically depending on the usage of the SMD. For example, if the usage is high, i.e., the SMD is running several apps or some resource-intensive apps, some of the parameters would have much higher while some would have lower values (depending on the resource type), which would make the SMD an unsuitable crowdworker at that moment. Therefore, to assess the SMD truly, it is crucial to measure not only the fixed hardware resources mentioned in the previous section but also their usability status at the present instance (i.e., at the time of job submission). In our experiment, we

considered the following parameters in this category:

Current CPU load: It denotes how much CPU is loaded in terms of running applications presently. A higher CPU load % denotes that at present, the CPU usage is high, i.e., the SMD is either running several apps simultaneously or running a CPU-intensive app. In either case, the SMD is not an ideal crowdworker at that moment. Hence, always a lower CPU load is preferred.

Current GPU load: A GPU-intensive job (generally a large number of small-scale but highly parallel tasks) would increase the GPU load. Like CPU, a lower GPU load is preferred.

Available RAM: Since the MCC client application and the MCC tasks and the related data need to be in the RAM for efficient execution, there should be enough free space in the SMD's RAM. The minimum RAM requirement would, however, depend on the particular MCC task. In this chapter, we considered an arbitrary RAM requirement for the experiment. The amount of available RAM is checked at the time of job submission.

Available battery: SMD battery is a sensitive parameter from the user's perspective. Users will generally decline to take part in MCC if they do not have enough charging left in their devices. Even if they initially agree to lend their device for MCC, they might suddenly go off if they realise that the task is juicing up the energy of their device fast. Hence, it is important to weigh up the available charging in the SMD battery before considering it as a crowdworker. In our experiment, we used a straightforward policy for this, i.e., to consider a certain cut-off % of the remaining charge. For example, the SMDs having below 75% charge would not be considered. It might be noted that this % calculation is relative. For example, the charge of 75% is of an SMD with 12000 mA of the battery will have much higher than a phone that has a battery of 6000 mA. However, we did not consider this factor because to a normal user, the distress of lower battery % is more psychological than practical.

Device temperature: Like the case of the battery, users also get concerned when their devices get heated up. People would not like to allow their SMD for MCC if it

is already hot. An SMD might get heated up due to several reasons. Though it is possible to retrieve temperatures of individual components such as processing unit, signal module, battery, etc., in this chapter, we considered the overall device temperature. As we did for the available battery, we considered a certain cut-off (^{o}C) for device temperature also.

Wi-Fi signal strength: For efficient communication, satisfactory signal strength is absolutely important. It becomes crucial in the case of mobile computing because weak signal strength creates a hindrance for data transfer which, in turn, affects the QoS. Though many factors may be responsible for weak signal strength, in a mobile computing environment, it is mainly due to the mobility of the users. The signal gets weaker as the user goes away from the AP. We considered the signal strength on a scale of 1-5.

4.3.3 Persistent Parameters

These parameters are calculated based on an extended assessment, reflecting the long-term behaviour of the SMD resources. The values of these parameters might not change significantly in a short duration, but they vary in a longer duration. In other words, these information of an SMD typically remain persistent for an average period of time. We considered the following parameters in this category:

Average CPU usage: In Section 4.3.2, we considered the current CPU load as one of the resource parameters because a higher CPU power does not guarantee enough free CPU cycles. However, the current CPU load also does not truly portray the CPU of the considered SMD. For instance, it might happen that the current CPU load of an SMD is low for the time being, but usually, it remains high. The reason for this might be, for example, that the user plays online games frequently. A large CPU-intensive job requires SMDs which have steady free CPU cycles for a long duration. Hence, an SMD with less CPU power but having unceasingly free CPU cycles for a long duration would be preferred than an SMD with higher CPU power but exhibiting frequently and randomly fluctuating CPU load. Moreover, the CPU usage pattern can reflect the battery consumption pattern also. If the CPU usage is to be high in the coming period, it is expected that the battery will also drain faster. To assess the CPU load of an SMD more practically, we took the

average CPU load of each device. During each connection session of an SMD s_i, defined by Eq. 4.1, the current CPU load of each SMD is recorded periodically. When s_i exits the network, its average CPU load for the immediate previous session N_i^{prev} is calculated using Eq. 4.2, which gives us the average CPU usage for a single session. We calculate the overall average CPU across all the sessions (i.e., since when the SMD was connected to the MCC coordinator for the first time) using Eq. 4.3.

$$N_i^{prev}(n) = s_i^{out\_time}(n) - s_i^{in\_time}(n) \tag{4.1}$$

where, $s_i^{in\_time}(n)$ and $s_i^{out\_time}(n)$ are the timestamps of s_i when it entered and exited the network for the n^{th} session, respectively.

$$s_i^{CL_{avg}}(N_i^{prev}) = \begin{cases} s_i^{CL}(t), & t < 2 \\ \frac{1}{t}\sum_1^t s_i^{CL}(t), & t \geq 2 \end{cases} \tag{4.2}$$

where, t is the instances (count) of the current CPU load record of s_i and $s_i^{CL}(t)$ is the t^{th} record of the current CPU load of s_i.

$$s_i^{CL_{avg}}(N_{all}) = \begin{cases} s_i^{CL_{avg}}(N_i^{prev}), & n < 2 \\ \frac{s_i^{CL_{avg}}(N_{all}) + s_i^{CL_{avg}}(N_i^{prev})}{2}, & n \geq 2 \end{cases} \tag{4.3}$$

Average GPU usage: Similar to average CPU load, for a more realistic assessment, we estimated the average GPU load of the SMDs using Eq. 4.4 and Eq. 4.5.

$$s_i^{GL_{avg}}(N_i^{prev}) = \begin{cases} s_i^{GL}(t), & t < 2 \\ \frac{1}{t}\sum_1^t s_i^{GL}(t), & t \geq 2 \end{cases} \tag{4.4}$$

$$s_i^{GL_{avg}}(N_{all}) = \begin{cases} s_i^{GL_{avg}}(N_i^{prev}), & N < 2 \\ \frac{s_i^{GL_{avg}}(N_{all}) + s_i^{GL_{avg}}(N_i^{prev})}{2}, & N \geq 2 \end{cases} \tag{4.5}$$

Availability: In MCC, users' mobility is an important criterion for considering the corresponding SMDs for job scheduling. If an SMD leaves the network without finishing the assigned job, it needs to be reassigned to another SMD. If it is not handled properly, the result (unfinished) and the job itself might be lost. Frequent disconnection would hamper the performance and the QoS of MCC. Therefore, before submitting a task to an SMD, it is better to be assured that it will not leave

until the job is finished. Since the MCC model presented in this chapter is campus-based, most of the users would have a certain mobility/availability pattern. It is possible to predict the availability of an SMD till the execution completion of the assigned task before dispatching the task to that SMD [36] [481]. In our selection policy, we considered the availability of the SMDs as a pre-selection criterion, i.e., if the SMD might not be available till the completion of the job, it would not be considered for selection at all. The predicted out-time c_O of an SMD s_i for the current session c is given by Eq. 4.6.

$$s_i^{c_O} = s_i^{c_I} + s_i^{a_p} \tag{4.6}$$

where, $s_i^{c_I}$ is the in-time of s_i for the current session and $s_i^{a_p}$ is the predicted availability duration of s_i from $s_i^{c_I}$.

The SMD s_i should be considered for job submission only if Eq. 4.7 satisfies.

$$s_i^{c_O} \geq j_{t_c} + k_1 + k_2 \tag{4.7}$$

where k_1 is the runtime of the prediction algorithm, k_2 is the lagging time between decision making and job dispatching, and j_{t_c} is completion time of the job j as defined by Eq. 4.8.

$$j_{t_c} = j_{t_s} + j_{t_e} \tag{4.8}$$

where, j_{t_s} is the submission instance of job j and j_{t_e} is its expected execution time. An idea of predicting execution time can be found in [35].

CPU Benchmark: Besides the core hardware, we wanted to check the actual performance of the device. Therefore, we benchmarked the CPU performance of each SMD individually. The details of benchmarking are discussed in the next section.

GPU Benchmark: Similar to CPU, we calculated the performance scores for GPUs of the SMDs. The details are given in the next section.

4.3.4 Customized Benchmarking

The raw hardware specifications may not reflect the device performance always correctly. For example, it is observed, on occasions, that an SMD with a higher clock frequency does not always offer better performance compared to an SMD

with a lower clock frequency. This is due to the fact that the computational capability of a device does not vary in direct proportion to the CPU and GPU power, rather some other factors such as internal data transferring delay, size and speed of the cache and its caching scheme, etc. also significantly influence the computation time. However, determining and predicting these irregularities are not trivial.

To mitigate this issue, i.e., to compare SMDs uniformly, benchmarks are popularly used. For smartphones, there are a number of benchmarking apps available such as Antutu[38], Geekbench[39], 3Dmark[40], etc., which assess the performances of various components of the device and compare with other similar devices. These apps run a series of pre-programmed operations (e.g., rendering some game scenes, stress testing of the processor and memory, and so on) on the device to test the respective components' performances.

However, since these apps are built and optimised differently, they are not particularly helpful for comparing the performance of different SMDs in executing similar tasks. Most of these apps focus on individual components of the SMD. Furthermore, they do not provide special features to describe how a device would perform during particular job execution. Also, the comparative assessment by these benchmarking apps may not be correct for all types of applications and processes. A general benchmarking scheme may not scale all SMDs equally for every type of processing task as these tasks might have varied resource demands.

Considering the above-mentioned issues, we preferred to have our customized benchmarking to assess the competency of the SMDs. In our MCC application, instead of individual components, we needed to assess the overall performance of the SMDs for a certain task. Customized benchmarking would allow assessing the device performance for particular job execution. Furthermore, the customized benchmarking would address the inconsistency in the standards of the SMD component specification. For example, let us consider the case of fabrication

[38] https://www.antutu.com/en/index.htm
[39] https://www.geekbench.com/index.html
[40] https://www.3dmark.com/

technology (nanometer (nm) values) that is used in GPU manufacturing which significantly determines its performance. A 7 nm chipset may deliver 20% better performance with a 40% reduction in energy consumption as compared to a 10 nm chipset. However, there is no universal standard to calculate the nm value. Different GPU makers calculate it differently. For instance, the 10 nm from TSMC is not equivalent to the 10 nm from Samsung, and Intel's 10 nm is equivalent to TSMC's 7 nm.

We considered the overall time taken to complete the job as the benchmark. Nevertheless, we calculated the CPU and GPU benchmarks separately. As a template, we evaluated the CPU and GPU benchmarks by performing a matrix addition and multiplication, respectively, as shown in Fig. 4.5. However, depending on the characteristics and requirements of the MCC task, different benchmarking schemes can be adopted.

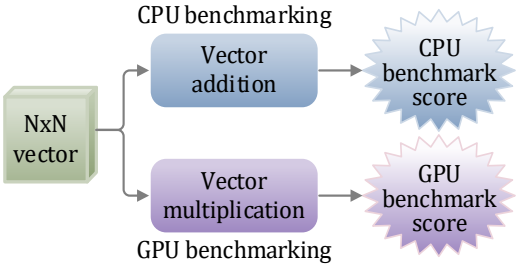


Fig. 4.5. Customized benchmarking scheme

Table 4.1. Specification and benchmark comparison of two sample SMD models

SMD model		Oppo A53	Redmi Y3
CPU	No. of cores	8	8
	Highest clock frequency	4x1.8 GHz Kryo 240	4x1.8 GHz Kryo 250
	Lower clock frequency	4x1.6 GHz Kryo 240	4x1.8 GHz Kryo 250 Silver
GPU	Model	Adreno 610	Adreno 506
	No. of ALUs	128	96
	Highest clock frequency	950	650
	Fabrication (nm)	11	14
	64-bit GFLOPS	28.8 - 31.2	28.8 - 68
Average CPU benchmark score		5689.308	4315.077
Average GPU benchmark score		17632.31	14817.75

Table 4.1 shows a sample comparison between two SMDs of different models but with nearly close specifications. It can be observed that though the first device is more 'powerful' compared to the second one, it took more time to execute the benchmark tasks both for CPU and GPU. This justifies the necessity of customised

benchmarking in assessing the SMDs practically.

The benchmark is calculated when an SMD connects the MCC coordinator as a crowdworker for the first time. The benchmarking tasks were sent to the SMD, and the execution time was noted.

The performance of a device varies from time to time. Generally, it degrades over time. Therefore, it is required to assess the SMDs periodically. To comply with, the benchmarking can be calculated after every k^{th} entry of each SMD. In our case, we took k=10; i.e., the benchmark was calculated at the 1st, 11th, 21st, ... in-time entries from the connection log of the corresponding SMD. Each benchmark score of a particular SMD was cumulatively averaged separately for the CPU and GPU. The average CPU and GPU benchmarks were calculated using Eq. 4.9 and Eq. 4.10, respectively.

$$s_i^{CB_{avg}}(k) = \begin{cases} s_i^{CB}(t), & k < 2 \\ \frac{s_i^{CB_{avg}}(t-1) + s_i^{CB}(t)}{2}, & k \geq 2 \end{cases} \tag{4.9}$$

$$s_i^{GB_{avg}}(k) = \begin{cases} s_i^{GB}(t), & k < 2 \\ \frac{s_i^{GB_{avg}}(t-1) + s_i^{GB}(t)}{2}, & k \geq 2 \end{cases} \tag{4.10}$$

4.3.5 Parameters that are not Profiled

Besides the above-mentioned resource parameters, there are a few more such as GPU clock frequency, number of GPU shader cores, GPU fabrication/architecture (nm), total RAM, available storage that we purposefully excluded from profiling. Though the GPU specifications are very crucial to determine the GPU capability of an SMD, all these information are not available uniformly for every SMD. In this experiment, we avoided the complexity in the resource selection process that might be introduced by this sparsity in the decision matrix. Regarding the other two parameters (i.e., total RAM and available storage), todays' SMDs are furnished with abundant RAM and storage. For example, Vivo Nex Dual Display Edition, Xiaomi Black Shark Helo, OnePlus 6T McLaren Edition, etc., have 10 GB RAM [638]. In general, most of the standard SMDs have 6 – 8 GB RAM. The same can be said for SMD storage. Hence, we felt that these two parameters could be excluded from

the list of selection criteria. However, if an MCC application necessitates assessing these parameters, they can always be included in the criteria list.

4.4 System Design

Our proposed MCC system, in principle, follows a two-tier client/server architecture, as shown in Fig. 4.6. Here, the MCC coordinator represents the server module, and the client module is the SMD. The server module should be able to take care of retrieving and storing the resource details of the connected SMDs efficiently such that the required data is available at the moment of SMD selection. Whereas the client module is supposed to carry out the instruction received from the server module at the SMDs. In terms of connections, there may be two types of SMDs:

- New: The SMD which a new entrant to the network and has no history in the system log.
- Returning: The SMD has been to the network before and has information recorded in the system log.

The connection details (e.g., the in- and out-times) are logged for all the SMDs. Since some data of the devices need to be stored for future usage, a database is required that would be connected to the server.

The details of each component of the system are discussed in the following subsections.

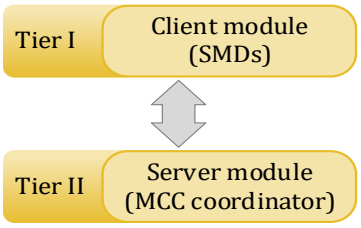


Fig. 4.6. Two-tier MCC

4.4.1 Server Module

Some of the key functionalities that a typical MCC server generally has are:

- Data communication: Communicates with the SMDs for data exchange.
- Resource profiling: Keeps record of various resources of the SMDs
- Resource availability prediction: Predicts the availability of an SMD in the

MCC network for uninterrupted job assignment and execution.

- Resource selection: Selects the most suitable SMD(s) for job distribution.
- Job distribution: Distributes MCC tasks to the selected SMDs
- Result collection and fault mitigation: Collects the processed output from the SMDs and takes correction measures in case of faults.
- Data storage: Stores information regarding SMDs and MCC tasks.

However, in this chapter, we primarily focus on data communication, resource profiling, and resource selection, as discussed in the following subsections.

4.4.1.1 Data Communication

The data exchange between the coordinator and an SMD is categorized into the following four types:

a) SMD registration: When an SMD having the MCC client application gets connected with the MCC coordinator for the first time, the coordinator creates a log entry for the SMD and collects other resource information, as discussed in Section 0.
b) Monitoring device usage: The coordinator periodically collects CPU and GPU usage details of the presently connected SMDs for calculating the persistent parameters, as discussed in Section 4.4.1.2.2.
c) Present resource status collection: The coordinator fetches the present status of the dynamic resource parameters when it has to perform SMD selection, as discussed in Section 0.
d) Task dispatching and result collection: The coordinator sends the MCC tasks and related data to the SMDs and gets back the result from them.

In this chapter, we considered the first three aspects of data communication, i.e., SMD registration and monitoring device usage and present resource status collection.

The server module connects to clients using sockets, as shown in Fig. 4.7. The required steps for initiating the communication between client and server are given in the following, while the process of setting the communication interface at the server and client ends is shown in Procedure 4.1.

1. The coordinator opens the server socket at a specified port.
2. It listens for client requests at the opened socket.
3. An SMD opens a client socket to communicate with the coordinator.
4. The client connects to the server at the static IP: PORT.
5. Once connected, the server creates a separate dedicated channel for the client to communicate.
6. The coordinator and the SMD communicate through this channel until the channel is terminated (intentionally or unintentionally).

Procedure 4.1: Communication Interface

```
//Server module interface
ServerSocket = openServerSockect(PORT) //server opens socket at a given port for communication
while(TRUE)
{
   NewConnection = ServerSocket.listen()//server listens to incoming connection request through the socket
   String UID = NewConnection.Read() //server module reads the data – UID, received through socket
}
//Client module interface
ClientSocket = openClientSocket(IP, PORT) //client opens a socket for communication with server at the specified server port
ClientSocket.Connect() //client connects to server through the socket
ClientSocket.Send(UID) //on successful connection, client sends its UID to server
```

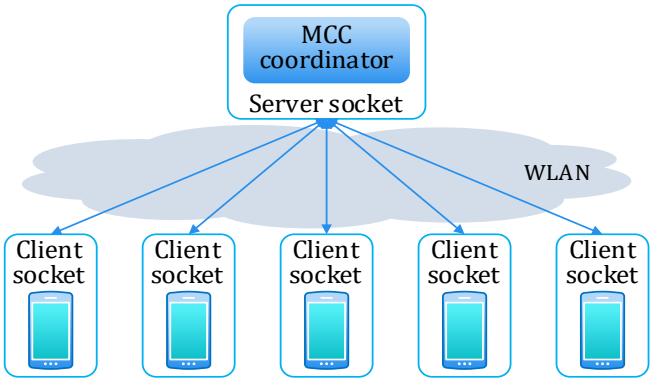


Fig. 4.7. The schematic diagram for client-server connection

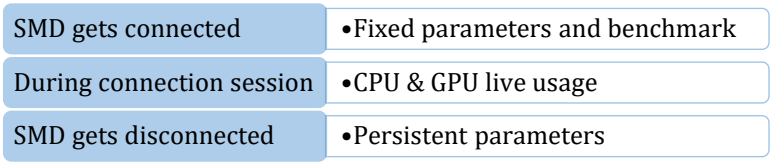


Fig. 4.8. Resource profiling phases

4.4.1.2 Resource Profiling

Resource profiling is a continuous process involving three phases, as shown in Fig. 4.8. Each phase is discussed in detail in the following. The process flow for resource

profiling is shown in Fig. 4.9.

Procedure 4.2: Resource Profiling - SMD Connected

```
ServerSocket = openServerSockect(PORT)//server opens socket at a given port for communication with the clients SMDs
db = openDatabase("SMD_profile") //database is opened
NewConnection = ServerSocket.listen() //server listens to incoming connection request through the socket
UID = NewConnection.Read() //server module reads the data – UID received through socket
login_time = getCurrentTime() //obtains the current system time as the login time of client SMD
dt = getCurrentDate() //obtains the current system date as the login date of client SMD
db.executeUpdate("insert into Connection_log(UID, login_time, login_date) values('"+UID+"',"+ login_time+","+ dt)  //UID, login time, and login date of the client SMD are stored in database
find = db.executeQuery("select UID from SMD_fixed_par") //the client SMD is searched in SMD_fixed_par table whether its specification are stored or not
if (find==null) then  //if the client SMD is not found in SMD_fixed_par table
{
   //Reading fixed parameters
   NewConnection.send("max_cpu_clk") //request made for maximum CPU clock frequency
   max_CPU_clk= NewConnection.read()//server reads the max CPU clock sent by the client SMD
   NewConnection.send("cpu_core") //request made for number of CPU cores
   CPU_core= NewConnection.read() //server reads the number of CPU core sent by the client SMD
   db.executeUpdate("insert into SMD_fixed_par(UID, max_CPU_clk,  CPU_core, con_freq) values('"+ UID + "',"+ max_CPU_clk + ","+ CPU_core + "," + 1) //fixed parameters of client SMD thus obtained is saved in database at server site
getBenchmark (db, NewConnection, UID, 1)
}
else
{
   frequency = db.executeQuery("select con_freq from SMD_fixed_par where  UID = '"+UID+"'") //obtain number of times the client SMD had been connected to server previously
   frequency++ //increase the number of times the client SMD gets connected to server by one
   db.executeUpdate("update SMD_fixed_par set con_freq = "+ frequency + "where UID = '"+ UID +"'") //update the connection frequency count in the SMD_fixed_par table for the newly connected SMD
}
```

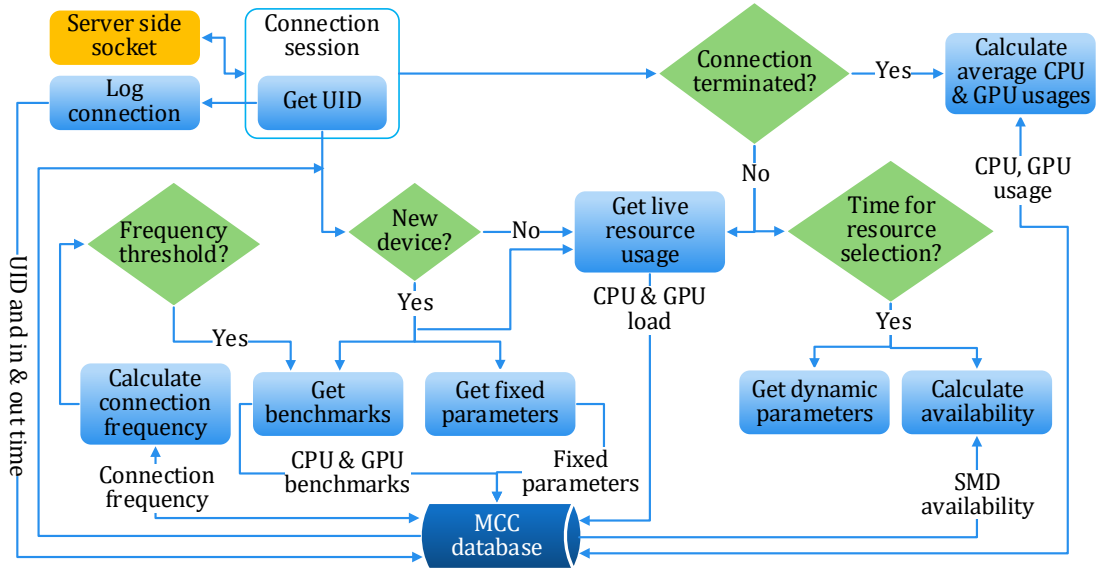


Fig. 4.9. Process flow for resource profiling

Procedure 4.3: Benchmark Calculation

```
getBenchmark(db, NewConnection, UID, frequency) //benchmark calculation for the newly connected SMD
{
  for i=0 to 5
    for j=0 to 5
      A[i][j] = random() //random numerical value is generated for matrix A
  for i=0 to 5
    for j=0 to 5
      B[i][j] = random() //random numerical value is generated for matrix B
  NewConnection.send("device benchmark")
  NewConnection.send(A, B) //matrices are sent to SMD
  CPU_benchmark_current = NewConnection.read() //CPU benchmark is received from the requested SMD
  GPU_benchmark_current = NewConnection.read() //GPU benchmark is received from the requested SMD
  if (frequency == 1) then
  {
//If the SMD is connected for first time its UID and CPU and GPU benchmarks are stored in Persist_par table
    db.executeUpdate("insert into Persist_par (UID, CPU_benchmark, GPU_benchmark) values ('"+ UID + "'," + CPU_benchmark + "," + GPU_benchmark)
  }
  else
  {
//If the SMD is connected for second time or later its previous and present CPU and GPU benchmarks are averaged individually and stored in Persist_par table
    CPU_benchmark_old = db.executeQuery("select CPU_benchmark from Persist_par where UID ='"+ UID + "'")
    GPU_benchmark_old = db.executeQuery("select GPU_benchmark from Persist_par where UID ='"+ UID + "'")
    avg_CPU_benchmark = (CPU_benchmark_old + CPU_benchmark_current)/2
    avg_GPU_benchmark = (GPU_benchmark_old + GPU_benchmark_current)/2
    db.executeUpdate("update Persist_par set CPU_benchmark ="+ avg_CPU_benchmark +", GPU_benchmark =" + avg_GPU_benchmark + " where UID = '"+ UID + "'")
  }
}
```

4.4.1.2.1 SMD Connected

When an SMD gets connected to the MCC coordinator for the first time ever, the coordinator acquires the fixed resource parameters and the benchmark of the device to be stored in the database permanently. The SMD sends its UID to the coordinator as soon as it connects to the coordinator. The coordinator logs the SMD by its UID and its login time. If the UID is already existing, it suggests that the SMD has previously joined this MCC system, and hence its information are already stored in the database. Otherwise, the SMD is newly connected, and it is required to store the required initial information, including the benchmark scores. The process of a new SMD connection is illustrated through the following steps, while Procedure 4.2 presents the programmatical procedure:

1. The SMD, using the client socket, connects to the server socket at the static IP:

PORT.

2. The coordinator reads the UID sent by the client and makes a record entry for the UID, login time and date in the *Connection_log* table.
3. The coordinator searches the UID in the *SMD_fixed_par* table.
4. If UID is not found, the coordinator requests the SMD for the fixed parameters and the CPU and GPU benchmarks.
5. The coordinator reads fixed parameters sent from the client connection and stores them in the *SMD_fixed_par* table.
6. The coordinator reads the benchmarks sent from the client connection and stores them in the *Persist_par* table, as per Procedure 4.3.
7. If UID is found, the frequency is incremented by one and updated in *SMD_fixed_par* table.

4.4.1.2.2 During Connection Session

The connection session denotes the period during which the coordinator and the SMD communicate to each other uninterruptedly through the established connection. This continues until either the coordinator or the SMD disconnects either intentionally or unintentionally. The coordinator periodically gathers CPU and GPU usage of the SMDs during the entire session and stores them in the *Live_usage_log* table. The interval for this data collection might vary depending on the design and implementation policy of the MCC system. The benchmarks are calculated periodically, i.e., after a certain value of *frequency*. The benchmarks are updated in *Persist_par*. The process is illustrated through the programmatical procedure in Procedure 4.4.

4.4.1.2.3 SMD Disconnected

The coordinator keeps track of when an SMD leaves the MCC network by logging out or due to connection loss or any other reasons. In either case, the current session is closed. While closing the session, the coordinator takes the CPU and GPU usage data from *Live_usage_log* table and calculates the CPU and GPU usage for all the logged sessions. Accordingly, these data are updated in the *Persist_par* table. The process is illustrated through the programmatical procedure in Procedure 4.5.

Procedure 4.4: Resource Profiling - During Connection Session

```
db = openDatabase("SMD_profile") //open database
frequency = db.executeQuery("select con_freq from SMD_fixed_par where UID = '"+ UID +"'")
//obtain number of times the SMD has been connected to LC
if (frequency%10==0) then
{
   getBenchmark (db, NewConnection, UID, frequency) //every 10th time the SMD is connected, its
CPU and GPU benchmarks are calculated
}
while (NewConnection)
{
   if (timeCounter(3000) == TRUE) //time interval considered as 5 minutes
   {
    //For a SMD is connected to LC its CPU and GPU usage are obtained
      at_time = getCurrentTime()
      at_date = getCurrentDate()
      NewConnection.send("CPU usage")
      cpu_usage= NewConnection.read()
      NewConnection.send("GPU usage")
      gpu_usage= NewConnection.read()
      db.executeUpdate("insert into Live_usage_log  values('"+ UID "',"+ cpu_usage + "," + gpu_us-
age + "," + at_time + "," + at_date) //SMD's CPU and GPU usage, UID, time, and date are saved in
Live_usage_log table
   }
}
```

Procedure 4.5: Resource Profiling - SMD Disconnected

```
db = openDatabase("SMD_profile")  //open database
dt = getDate()
lin = getLogInTime(UID)  //obtain the login time of the SMD
lout = getLogOutTime(UID)  //obtain the current logout time of the SMD
if (NewConnection==FALSE) then
//When session closes the average of current and old CPU and GPU usage is calculated and are
saved in Persist_par table
{
   //Obtain the average CPU and GPU usage between login and logout time for current session from
Live_usage_log table
   Avg_cur_CPU=db.executeQuery("select AVG(CPU_usage) from Live_usage_log where UID = '"+
UID +"' AND atDate = "+ dt +" AND atTime between" + lin +" AND "+ lout +")
   Avg_ cur_GPU=db.executeQuery("select AVG(GPU_usage) from Live_usage_log where UID =
'"+ UID +"' AND atDate = "+ dt +" AND atTime between" + lin +" and" + lout +")
   //Obtain the average CPU and GPU usage for previous login sessions from Persist_par table
   Avg_old_CPU= db.executeQuery("select avg_CPU from Persist_par where UID = '"+ UID +"'")
   Avg_old_GPU= db.executeQuery("select avg_GPU from Persist_par where UID = '"+ UID +"'")
   //Average CPU and GPU is calculated
   Avg_CPU = (Avg_old_CPU + Avg_cur_CPU)/2
   Avg_GPU = (Avg_old_GPU + Avg_cur_GPU)/2
   db.executeUpdate("update Persist_par set avg_CPU ="+ Avg_CPU + ", avg_GPU =" + Avg_GPU
+ " where UID = '"+ UID +"'")  //average CPU and GPU usage is saved in table Persist_par
   db.executeUpdate("update Connection_log set selected = 'FALSE'  where UID = '"+ UID +"' AND
logout_time = lout")  //connection termination is recorded in Connection_log table
}
```

4.4.1.3 Resource Selection

To achieve the optimal effectivity (e.g., response time, throughput, turnaround time, etc.) and reliability (e.g., fault tolerance, ensuring resource availability, minimized device mobility, minimized hands-off, etc.) from the MCC, it is very crucial

to select the most suitable SMDs among the currently available ones as per the requirement of the MCC task and the application type [156].

Table 4.2. Parameters that are not a part of the selection process but set as threshold criteria

Parameters	Threshold	Remark
Available RAM	50 MB	The required RAM size is to be specified by the MCC application. However, for experimental purposes, we considered the threshold value of 50 MB, which might be sufficient for running general computing-intensive tasks [639].
Available battery	>70%	We have optimistically assumed that people will not be so concerned if the remaining battery charge is greater than equal to 70%.
Device temperature	<45°C	40°-45°C is treated as normal device temperature, running usual apps.
Availability	1	The availability would have a binary evaluation, i.e., whether the SMD would be available or not, and calculated using Eq. 4.7.

Table 4.3. Considered parameters for crowdworker selection

Parameters	Optimized value
CPU frequency	Maximum
No. of CPU cores	Maximum
Current CPU load	Minimum
Current GPU load	Minimum
Signal strength	Maximum
Average CPU usage	Minimum
Average GPU usage	Minimum
CPU benchmark	Minimum
GPU benchmark	Minimum

When needed (i.e., resource required for MCC task execution), the coordinator calls the resource selection module. The selection module gets the instantaneous values of different parameters, as discussed in Section 4.3 and based on which generates a list of top-ranked SMDs. The selection module works in two phases. In the first phase, it filters the SMDs based on the threshold criteria for the parameters listed in Table 4.2. The SMDs that pass through this filtering are considered for the next level of the selection process, in which the COPRAS method is used to select the best SMD based on the parameters listed in Table 4.3. We used a flag to distinguish between the already selected SMDs that are executing some MCC tasks and the other available SMDs. If an SMD is already engaged with an MCC task, it would not be considered for the selection. The selection process is outlined in Procedure 4.6.

Remark 4.1. In this chapter, we added this resource selection module for the shake of completeness. The details of MCDM-based resource selection are separately

discussed in Chapter 5, Adopting COPRAS method in Procedure 4.6 is influenced by the outcome of Chapter 5.

Procedure 4.6: Resource Selection

```
db = openDatabase("SMD_profile") //open database
SMD_UIDs[] = db.executeQuery("select UID from Connecttion_log where login_date = "+ date+"
AND logout_time = null AND selected = 'FALSE'")
//SMDs currently logged in to LC but not yet selected for job processing are identified and stored in
array
Decision_matrix[][] //decision matrix is built
i=0
for each UID in SMD_UIDs //considering each identified SMD from the array
{
   connection = getSessionConnection(UID)
   //Obtain dynamic parameters from the presently selected SMD
   connection.send("Available RAM")
   available_RAM = connection.read()
   connection.send("Available Battery")
   available_battery = connection.read()
   connection.send("Device Temperature")
   device_temp = connection.read()
   connection.send("CPU Load")
   CPU_load = connection.read()
   connection.send("GPU Load")
   GPU_load = connection.read()
   connection.send("Wifi Strength")
   wifi_strength = connection.read()
   available = calculateAvailability() //calculate the availability of the present SMD for job pro-
cessing
   if (available_RAM >= R AND available_Battery > 70 AND device_temp < 45 AND available) then
//check the SMD's available RAM, battery, device temperature and availability is beyond threshold
value
   {
      max_CPU_clk, CPU_core, = db.executeQuery("select max_CPU_clk, CPU_core, from
SMD_fixed_par where UID = '" + UID+ "'") //max CPU clock speed and CPU cores are obtained
from SMD_fixed_par table
avg_CPU, avg_GPU, CPU_benchmark, GPU_benchmarks = db.executeQuery( "select avg_CPU,
avg_GPU, CPU_benchmark, GPU_benchmarks from Persist_par where UID = '" + UID+ "'") //the
average CPU, GPU and benchmarks are obtained from Persist_par table
      Decision_matrix[i][] = {max_CPU_clk, CPU_core, wifi_strength, CPU_load, GPU_load, avg_CPU,
avg_GPU, CPU_benchmark, GPU_benchmarks} //max CPU clock speed, CPU cores, signal strength,
CPU load, average CPU and GPU usage, and CPU and GPU benchmarks of the SMD are stored in
decision matrix
      i++
   }
}
SMD_Rank[] = executeCOPRAS(Decision_matrix) //apply COPRAS method to find the most suitable
SMDs for job processing
```

4.4.2 Client Module

The client module is implemented in SMD. The functionalities of the client module include communicating with the server module and executing the assigned MCC tasks. Since in this chapter, we do not consider any job processing scenario but only resource profiling, we focus only on the data communication aspect. From the

client's perspective, the data exchange between the client and the server is categorized as follows:

a) Device information: An SMD sends its UID to the coordinator as soon as it joins the MCC network. Afterward, it provides other information as asked by the server module.
b) Job execution: It receives the MCC tasks and related data from the coordinator and sends back the outputs after execution.

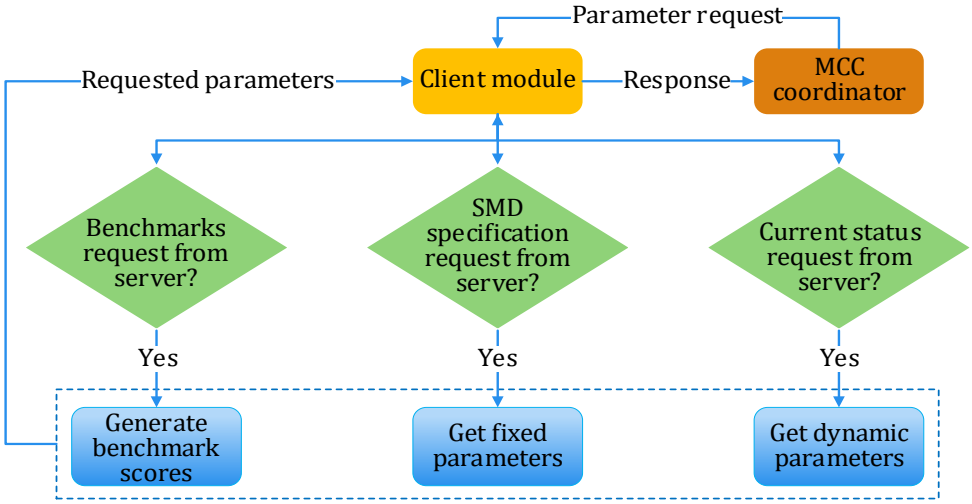


Fig. 4.10. Process flow for the client-server communication

As mentioned earlier, if the SMD connects for the first time, along with its UID, the fixed parameters and the benchmark values are sent to the coordinator. And if the SMD is re-entrant, then it sends the dynamic parameters and live resource usage data as requested by the coordinator. The entire communication process is illustrated in the following steps, while the process flow and the corresponding sequence diagram are shown in Fig. 4.10 and Fig. 4.11, respectively.

1. The SMD opens a client socket and searches for the coordinator (server module) by its IP in the MCC network. If the coordinator is found, the SMD connects to it at a specified port.
2. The SMD sends its UID to the coordinator and waits for a response.
3. The SMD, on receiving a request for fixed parameters from the coordinator, sends back the fixed parameters' values.
4. The SMD, on receiving a request for benchmark, accompanying the related assessment data from the coordinator, calculates the benchmark as described

in Section 4.3.4 and sends it back to the coordinator.

5. The SMD, on receiving a request for CPU & GPU live usage from the coordinator, sends the required data back to the coordinator.
6. The SMD, on receiving a request for dynamic parameters from the coordinator, sends back the dynamic parameters' present values.

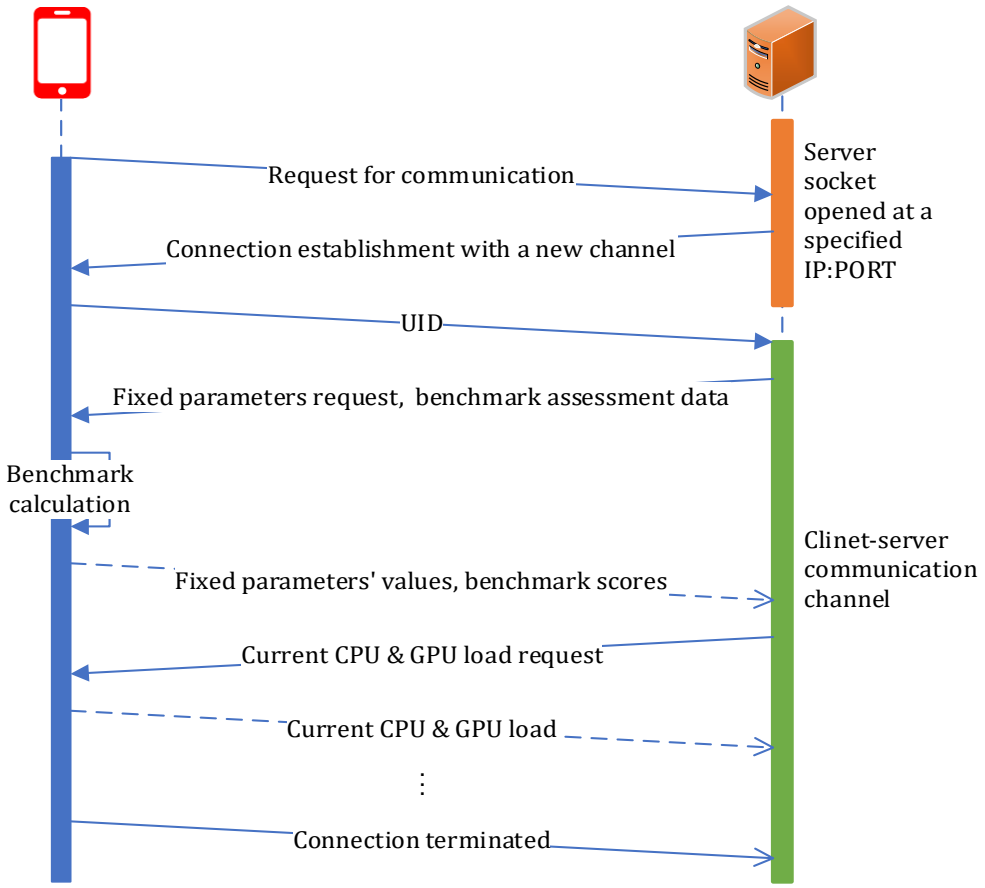


Fig. 4.11. Sequence diagram for client-server communication

4.5 System Development

As per the system design, as discussed in the previous section, we present the details of the development in this section. The details of the devices and tools used for system development are mentioned in Table 4.4.

4.5.1 Server Module

Our goal is to design a low resource demanding MCC coordinator. Hence, we developed a lightweight server module, comprising only the necessary components. For this, we preferred Java, considering its versatility. Also, Android has inbuild compatibility for Java.

Table 4.4. Developmental environment specifications for the resource profiling and selection system

	Device/tool	Specification details	Purpose
Hardware	Workstation (laptop)	Processor: AMD Ryzen 3 3250U quadra core (64 Bit) dual-core CPU (2.6 GHz, 64 bit) RAM: 4 GB OS: Windows 11, 64 Bit	Developing the server and client software modules and testing the system
	Wireless router	D-Link DIR-600M wireless router	For connecting MCC coordinator and SMDs wirelessly
	Smartphone	Model: Vivo Y20A CPU: 1.95 GHz Snapdragon439 (12 nm) octa-core (4 × 1.95 GHz Cortex-A53 and 4 × 1.45 GHz Cortex A53) GPU: Qualcomm Adreno 505 RAM: 3.00 GB OS: Android 11	Developing and testing the client module
Software	Notepad++	Version 7.5.7, 32 Bit	As code editor
	Java/JDK	Version 8	Developing the server module
	Android Studio	Version 3.3.1	Developing the client module
	Database application	MySQL 8.0 Community Server	Storing information related to SMDs and MCC tasks

4.5.2 Client Module

Considering the popularity and wide availability of Android devices, we preferred to develop the client application targeting the Android-based SMDs. We used Android 4.4 KitKat as client API. We deliberately considered this version of Android to include the prospective users having both newer and older SMDs. The client socket connects to the server socket at server port 1026.

We developed the client module as a service that would run in the background. A service is an application component generally used for longer operations running in the background continuously, even if the user switches to other applications. Usually, a service runs with a higher priority than other inactive or background applications and processes, and hence there is little possibility of it getting terminated by the Android. However, if, in any case, it is terminated by the Android, it can be configured to be restarted once sufficient system resources are regained.

In our client module, the user has control over enabling or disabling the service explicitly. However, the service would be disabled for that particular session only. It is automatedly enabled whenever the SMD gets disconnected from the network. Automated resuming of the service would ensure the continuous availability of the

SMDs for MCC when they are in the network. The service is deliberately enabled because it is impractical for a user to remember enabling the service every time whenever she joins the MCC network.

4.5.3 Database

We used MySQL server for designing the database named *SMD_profile* to store the required SMD information. Different aspects of SMDs' profiles are stored in four different tables, the details of which are given in Table 4.5.

4.6 Implementation

Before implementation, we tested the system for its proper working. We performed unit testing for all the functionalities of each module individually. Initially, for system testing, we considered a small number of SMDs of different configurations.

After satisfying testing, we implemented the developed resource profiling and selection for an MCC system in a real environment. We deployed the system at the Data Engineering Lab of the Department of Computer Science & Engineering at National Institute of Technology, Durgapur. The lab is generally accessed by the institute's research scholars, the project students, faculty members, and the technical staff. The coordinator was connected to the Wi-Fi router installed in the lab. The SMDs that are interested in crowd computing would connect to the coordinator through the same router. The implementational details are discussed in the following, while Table 4.6 lists the details of hardware and software used for implementing the resource profiling and selection system.

4.6.1 Server

We aim to deploy the MCC in an environment which would not require an extensive infrastructure-based computing system. Hence, we used a small, low-powered Linux-based SoC as the MCC coordinator that can be set up anywhere without much burden to IT infrastructure and IT budget. Particularly, we used Raspberry Pi to deploy the server module that would act as the coordinator. It not only has sufficient computing capacity, but it supports small user-created databases as well.

Table 4.5. Database schema for SMD profiling

Table	Purpose	Attributes	Data type
Connection_log	Log in and out time of all the connected SMD	UID	Text
		Login_time	Time
		logout_time	Time
		login_date	Date
		selected	Boolean
SMD_fixed_par	Fixed parameter and device benchmark of a SMD	UID	Text
		max_CPU_clk	Number
		CPU_core	Number
		con_freq	Number
Live_usage_log	SMD's live CPU and GPU usage data at an instant of time	UID	Text
		CPU_usage	Number
		GPU_usage	Number
		atTime	Time
		atDate	Date
Persist_par	SMD's persistent parameters based on its historical behavior	UID	Text
		avg_CPU	Number
		avg_GPU	Number
		CPU_benchmark	Number
		GPU_benchmark	Number

Table 4.6. Implementational environment specifications for the resource profiling and selection system

Category	Entity	Specifications	Purpose
Hardware	Raspberry Pi 3 Model B[41]	Processor: 1.2 GHz (ARM Cortex-A53) RAM: 1 GB Storage: 8 GB OS support: Linux Communication network: Wi-Fi, Ethernet USB ports: 4 Ethernet port: 1 HDMI port: 1	MCC coordinator (server)
	SMDs	Varies for different devices	Crowdworker (client)
Software	Server module	JRE/JDK version 8	Runtime environment for running server module
	Android	Version 4.4. (KitKat) and above	The SMD OS that hosts the client module and runs the MCC tasks on the SMD.

4.6.2 Client

The client module was installed on the SMDs of the students, staff, and faculty members who regularly accessed the lab. All the SMDs were Android devices with KitKat version and above. We logged the SMD information for nearly eight months

[41] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

of several users (whoever connected to the AP during this period).

4.6.3 Networking

In an organisational MCC setup, there might be several small-scale MCC installed in different buildings or floors. These MCCs might be isolated silos or interconnected through a backbone network. In our MCC model, we considered an interconnected network. However, each MCC functions independently, i.e., they do not share any MCC-related information. The SMDs would be connected to the coordinator through a Wi-Fi router which is connected to a network switch. The local coordinator is also connected to the same Wi-Fi router. In a local MCC, the coordinator and the SMDs connected to that coordinator are in the same address space. The coordinator always has a fixed address in this address space so that when an SMD connects to a particular Wi-Fi router, it can easily locate the coordinator and initiate the communication for MCC. We made this address common to every coordinator, wrapped as a subnet mask. This would allow the client module to connect different MCCs seamlessly. In our setup, every Wi-Fi AP would connect maximum 2^6 devices, and the coordinator was configured by assigning the fixed IP 192.168.x.y2, where 'x' denotes the third octet and 'y' denotes the first two bits of the fourth octet. The server socket was configured at port number 1026 for listening to the client connections. A typical layout for an organisation MCC is shown in Fig. 4.12.

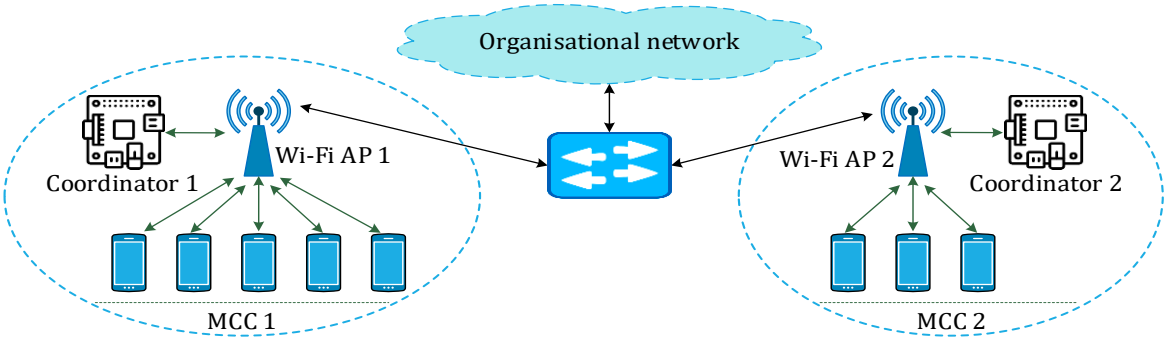


Fig. 4.12. Network layout of a local MCC

4.7 Limitations and Further Scopes

The benchmarking scheme proposed in this chapter is generalised. But for different types of jobs, different competencies are required. Hence, it would be more efficient if benchmarking could be done for every particular computing task which

is expected to be run through MCC. Furthermore, we followed a single predefined benchmarking calculation approach to assess the SMDs, which suggests that the MCC tasks are somewhat determined. This makes the benchmarking inflexible, i.e., it would be impracticable to change the benchmarking scheme dynamically as per varied requirements of different MCC tasks. The benchmarking mechanism could be made more accommodating if this was implemented as a lightweight, portable service that could be invoked by the coordinator at the client module whenever required.

To avoid inconsistency, we shunned profiling some crucial GPU parameters such as clock frequency, no. of shader cores, and fabrication. These might be crucial for a GPU-intensive job. The proposed system can be updated with profiling these information along with a way out to handle the sparsity in the decision matrix, i.e. if some of the information are not available from all SMD. We also did not consider the case if the SMD is in the charging state. Without considering this, we probably would miss out on some potential SMDs that were superior in all aspects but filtered out due to low battery.

Though we considered a local MCC scenario, the proposed resource profiling model can certainly be extended to an intra- (multiple local MCCs within a single network) and inter-MCC system (multiple local MCCs across different networks), with some added complexity. The distributed resource management solutions could be quite helpful in this regard.

4.8 Summary

To achieve the best out of MCC, it is highly desirable to consider the most potent SMDs for MCC task execution. For that, it is necessary to assess the SMDs' resources accurately and justifiably. And for assessing the SMDs, their various resource specifications need to be profiled appropriately.

In this chapter, we presented a framework for profiling the SMD resources in a methodological approach. We considered different categories of resource parameters such as fixed (e.g., clock frequency and cores of the CPU) and variable (e.g., present CPU and GPU load, remaining battery, etc.) parameters. Besides these two

usual types of parameters, we further considered some behavioural parameters (e.g., average CPU and GPU load, availability) that would suggest the resource stability pattern of a certain SMD. Not relying on the straightforward assumption that the SMD, which has the best quantifiable hardware specifications, would be the most competent computing device to assess an SMD practically, we opted for a customised benchmarking scheme to estimate its computing capability with respect to the CPU and GPU.

We considered a local MCC scenario where the SMDs are connected to an organisational Wi-Fi AP, and most of them connect to the same network on a regular basis. The profiling of the resources would be done by the MCC coordinator. We presented the minute details of designing, developing, and implementing the resource profiling and selection for such a local MCC.

5
Resource Selection in MCC

"The choice to make good choices is the best choice you can choose. Fail to make that choice and on most choices you will lose." --- Ryan Lilly

5.1 Introduction

Resource selection is a primary step for task scheduling, having a high impact on the overall performance of the MCC. The effectivity (e.g., response time, throughput, turnaround time, etc.) and reliability (e.g., fault tolerance, ensuring resource availability, device mobility handling, minimized hands-off, etc.) of MCC largely depend on selecting the right resources for scheduling the MCC tasks.

At any instance, there might be quite many SMDs available at a certain place (local MCC, connected through a WLAN or other short-range communication means) [463] [464] or for a certain application (global MCC, connected through the internet) to be considered as computing resources [462] [145] [461]. All of these SMDs might have diverse values of each resource parameter, which means each SMD will have different capability as a computing resource.

Now, among this sizable pool of resources, which of them would be chosen as crowdworker? In other words, to which SMDs the MCC jobs are preferred to be scheduled? Obviously, it is expected that for a given MCC task, the best SMDs would be preferred.

Also, for a given task, to minimise the overhead, it is desirable to use a minimum number of SMDs. This can be achieved by selecting the most capable SMDs for scheduling and execution of any MCC task. It would not only maximise the throughput but also improve the overall performance and QoS of the MCC system. An inefficient SMD selection method will have a negative impact on the QoS of MCC. Therefore, it is crucial to have an efficient mechanism to select the most suitable SMDs from the available pool of SMDs as per the requirement of an MCC task [156].

In this chapter, we aim to achieve the followings:

- Establish the problem of resource selection in MCC as an MCDM problem.
- Find out the most suitable MCDM approach that provides consistent and considerably accurate SMD selection decisions, balancing various parameters at a reasonable time complexity for real-time resource selection in a dynamic environment like MCC.
- Assess and compare the performance of different MCDM methods belonging to disparate families in terms of the correctness and robustness of the SMD rankings given by each method and their precise run-times.

5.2 Resource Selection and MCDM

In this section, we formally define the resource selection problem in MCC and establish it as an MCDM problem.

5.2.1 Challenge in Resource Selection in MCC

Resource discovery and selection in a distributed system has always been challenging [640] [641] [642]. In MCC, it has been aggravated by the fact that the MCC implicates an absolutely dynamic and heterogeneous environment. The heterogeneity is not only due to the assorted characteristics of the SMDs (physical diversity) but also because of users' divergent device usage behaviours (operational diversity).

A number of SMD makers regularly launch plentiful devices with a variety of hardware and software features. People also tend to upgrade (replace) their SMDs regularly [2]. Hence, in most of the cases, the available SMDs in an MCC would be vastly heterogeneous in terms of hardware (e.g., CPU & GPU clock frequency, number of cores, primary memory size, secondary memory size, battery capacity, etc.); and with different specifications, the SMDs boast varying computing capacities [138].

The computing capability is one of the most important selection criteria as this would eventually influence the response time, throughput, and turnaround time for any given task. Therefore, the straightforward solution would have been to pick the SMDs that have the best-quantified hardware resources. But this does not

always hold true, i.e., the SMD with best hardware assets may not be the optimal computing resource at every instance. Due to users' SMD usage behaviour and the applications running on the SMDs the usable resources would vary time to time. At different instances, the SMDs would have different values of their dynamic resource parameters.

Table 5.1 lists three such cases and their respective corollaries, considering two SMDs S_1 and S_2 belonging to the users U_1 and U_2, respectively. It is obvious that though S_1 has better resources than S_2, S_1 should not be the most preferred candidate for MCC. This indicates that the greater resources on paper should not be the sole measure for considering an SMD to be the fittest as a crowdworker. To compare the competency of the SMDs, they need to be weighed multidimensionally, i.e., based on different types of information. The other parameters and external conditions also should be taken into consideration. Furthermore, at the time of job submission, the actual statuses (which varies dynamically) of the resources (e.g., CPU & GPU load, available memory, available battery, signal strength, etc.) need to be considered. The values of these variable parameters change depending on the SMD usage by its user. For a better QoS of MCC, it is crucial to select the most suitable SMDs with the best usable resources to offer at the moment of job submission and during its execution.

Furthermore, besides the static and dynamic resource parameters, to assess the SMDs more accurately, some other persistent parameters such as average CPU and GPU usage, users' mobility pattern, etc., should also be considered. As discussed in Chapter 4, these parameters would suggest the pattern of the SMD's usability behaviour.

It is apparent that taking into account the diversity in the considerable parameters, selecting the most suitable SMD as a candidate for crowdworker is a difficult task. Moreover, while making the selection decision, these diversified parameters need to be considered in a unified way. This is really challenging and it makes the selection process complex. Furthermore, several of these parameters are conflicting in nature. It poses further challenge to have an optimized selection balancing between the conflicting assessment criteria.

Correspondingly, devising an efficient, dynamic, and time-efficient resource selection method that would perfectly consider all these diverse resource parameters is non-trivial.

Table 5.1. Examples of resource selection impasses

Case	Respective corollary
S_1 and S_2 have CPU power c_1 and c_2, respectively; where $c_1 > c_2$ but the user U_1 continuously plays games on his/her SMD.	The CPU of S_1 is busy most of the time, which would hamper the execution of the task assigned to it, whereas S_2 can provide slower but steady CPU cycles.
S_1 has considerably larger battery capacity than S_2, but S_1 is extensively used compared to S_2.	The battery drainage of S_1 is much faster than S_2. Hence, it may happen that even the job is submitted to S_1, after a short while, u1 withdraws from MCC due to low battery concern. If the job was submitted to S_2, the overhead of job offloading, and reassigning could be avoided.
S_1 has far better hardware than S_2 in all respects, but it has a considerably weaker wireless signal strength than S_2 at the time of job submission.	The weak wireless signal may result in disrupted and unreliable communication. Hence, due to this poor communication channel, the computing strength of S_1 is not utilized properly, whereas S_2 could provide sluggish but reliable service.

5.2.2 Defining the Resource Selection Problem in MCC

The resource selection problem can be formally presented as following. Assume that there is a set of m number of heterogeneous SMDs as $S^{(t)} = \{s_1, s_2, ..., s_m\}$ connected with the MCC coordinator at time t. Here, each SMD (s_i) is characterized by n number of resource parameters as $R = \{{r_1}^{w_1}, {r_2}^{w_2}, ..., {r_n}^{w_n}\}$, where w_j is the associated weight of resource r_j. R consists of different parameters types, as discussed in Section 4.3. Typically, $\forall s_i$, i = [1,m], except the fixed resource parameters (r^f), the dynamic resource parameters (r^d), persistent resource parameters (r^p) and the benchmarks (r^b) vary at times, depending on various factors, where $r^f \cup r^d \cup r^p \cup r^b = R$. With this variations, it is challenging to select the best s_i in S at any point of time t, based on the current value of $r_j^{x(t)}(s_i)$ and respective w_j, $\forall r_j$, $\forall w_j$ and $\forall s_i$, where j = [1,n], i = [1,m], and $x = f|d|p|b$.

For some parameters, as mentioned earlier, instead of including them in the selection process we used them for pre-selection filtering. Some threshold criteria were set for these parameters. If n' is the total number of such parameters then $({r_1}^t \wedge {r_2}^t \wedge ... \wedge {r_{n'}}^t)$, where ${r_1}^t, {r_2}^t, ..., {r_{n'}}^t \in R$, needs to be true for $\forall s_i$ to be qualified for the next phase of the selection process. Let us assume, without these parameters having threshold criteria, the resource set is $R' = R - \{{r_1}^t, {r_2}^t, ..., {r_{n'}}^t\}$,

and after these filtering conditions, the updated set of SMDs, $S'^{(t)} = \{s'_1, s'_2, ..., s'_{m'}\}$

If we observe the considered resource parameters, they are conflicting in nature, i.e., for some parameters ($r_{j^{max}}$), we would prefer the greater values (maximum), whereas for the rest ($r_{j^{min}}$), the smaller values (minimum) would be ideal, where, $r_{j^{max}} \cup r_{j^{min}} = R$. Our goal is to select an s_i such that it comprises an overall balance of $r_{j^{max}}$ and $r_{j^{min}}$, considering all r_j. In fact, we need a resource selection method that would evaluate all s_i considering the present values of $r_{j^{max}}$ and $r_{j^{min}}$ and their corresponding weights w_j, $j = [1, n - n']$, and rank them accordingly.

Based on the value of $r_j^{x(t)}(s'_i)$ and taking into account the maximization and minimization criteria and w_j, a utility value ($s_i'^{u(t)}$) is to be calculated of all s'_i with respect to $\forall r_j$, $j = [1, n - n']$ at t. And finally, comparing all $s_i'^{u(t)}$, $i = [1, m - m']$, a resource selection method should return an updated $S'^{(t)}$ as $S''^{(t)} = \{s''_1, s''_2, ..., s''_{m'}\}$ where, $s_i''^{u(t)} \geq s_{i+1}''^{u(t)}$.

An ideal selection method should comply the above-mentioned considerations and return a ranked list of the fittest SMDs.

5.2.3 MCDM

In most real applications, decision-making is not based on a simple if-else pattern; rather, due to the presence of multiple potentially conflicting requirements, balanced decision-making becomes nontrivial. Traditional optimization-based solutions try to mitigate this by considering the most important requirement(s) as the objective function(s) and the remainder as constraints. But it still might have the problem of potentially irreconcilable requirements, which can be handled to an extent by relaxing the thresholds of the constraints until a feasible solution is obtained [643].

MCDM solutions are suitable alternatives for solving these kinds of problems. MCDM methods, in general, adopt an interactive approach to deal with the selection problems with multiple criteria by utilising a variety of processes that clarify the implications of the underlying trade-offs between the considered criteria in

constituting the alternative solutions [174].

An MCDM problem can be described as weighing a set of alternatives based on multiple pre-set decision criteria. Let us suppose, $A = \{a_1, a_2, \ldots, a_m\}$ denotes a finite set of distinct alternatives and $C = \{c_1, c_2, \ldots, c_n\}$ is the set of criteria that evaluates A. A performance score p_{ij} is calculated for each a_i, $i = 1, 2, \ldots, m$ with respect to the criteria C. Based on p_{ij}, A is ordered such that a_k is better than a_{k+1}.

Though the alternatives are homogeneous, the interrelationship between the decision criteria may be complex in nature. They can be expressed in different units which might not have any apparent interrelationship. However, the decision-making gets complicated when some of the criteria conflict with each other; i.e., some are profit (maximization) criteria, whereas some are loss (minimization) criteria. Usually, each criterion has some weight as per their significance in the decision-making in the context of a particular application [644]. The common stages of a typical MCDM method are shown in Fig. 5.1.

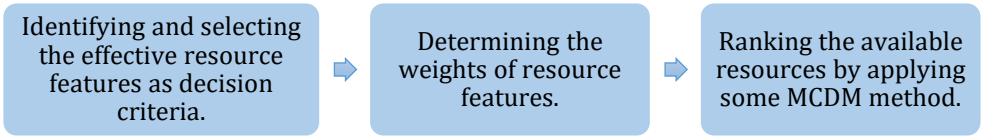


Fig. 5.1. Typical MCDM stages

MCDM methods have been used for decision-making in numerous applications and problems [153] [154]. Over the years, several algorithms have been developed or augmented, targeting the applicability and suitability of the problems, which significantly contributed to the evolution of the expanding field of MCDM [645] [153] [646]. These methods differ in terms of their computational logic and assumption, applicability, calculation complexities, and ability to withstand variations in the given conditions. Table 5.2 lists some of the popular MCDM approaches and the most noteworthy representatives of each approach. Some MCDM methods work better for a particular decision-making problem, while others may not perform well. That is why it is important to decide the most suitable MCDM method for a given problem scenario.

5.2.4 Resource Selection as an MCDM Problem

In the stated problem, we wish to find out a rank-based selection method for listing

the best SMDs uniquely with tolerable runtime, fulfilling the demand for real-time dynamic resource-provisioning. For this, we present the resource selection problem as an MCDM problem, which is expected to assess the SMDs in a balanced way based on their various resource parameters.

Table 5.2. The popular MCDM approaches and their respective popular representatives

MCDM approach	Representative example	Reference
Distance-based method	TOPSIS	[647] [648]
	EDAS	[649]
Area-based comparison and approximation method	MABAC	[650] [651]
Ratio-based additive method	ARAS	[652] [653]
	SAW	[654]
	COPRAS	[655] [656]
Algorithms that work under compromising situations	VIKOR	[657] [658]
	CoCoSo	[659]
	MARCOS	[660]
	RAFSI	[644]

Witnessing the wide-scale applications of MCDM, especially in decision-making problems, we believe that it can also offer promising solutions for resource selection in MCC and other similar computing systems, which is not explored so far [480].

In our SMD selection problem, the alternatives are the available SMDs at the time of job submission, and the criteria are different parameters considered for SMD selection (e.g., CPU frequency, RAM, CPU load, etc.). The MCDM solutions provide a ranking of the available SMDs based on the selection criteria. From this ranked list, the resource management module of the MCC selects the top-ranked SMD(s) for job scheduling. The pictorial representation of the SMD ranking process is shown in Fig. 5.2.

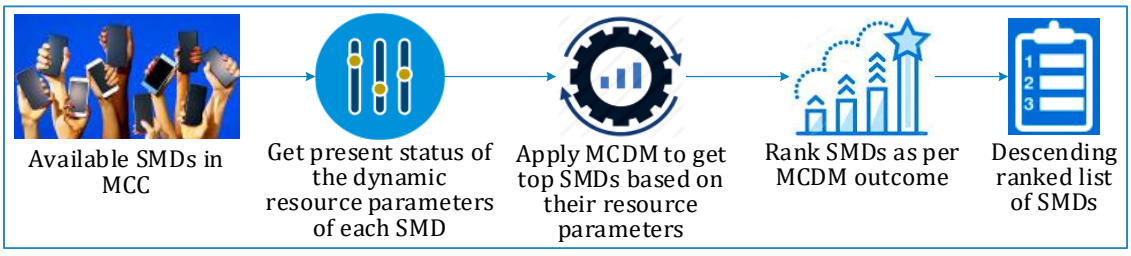


Fig. 5.2. SMD ranking using MCDM

5.2.5 MCDM Methods Considered for the Comparative Study

This section briefly describes five MCDM methods considered for the comparative analysis along with their computation algorithms. In this chapter, we derived the preferential order of the alternatives based on the following aspects:

- Separation from average solution (EDAS method).
- The relative positioning of the alternatives with respect to the best one (ARAS method).
- Utility-based classification and preferential ordering on the proportional scale (COPRAS method).
- Approximation of the positions of the alternatives to the average solution area (MABAC method).
- Compromise solution while trading of the effects of the criteria on the alternatives (MARKOS method).

Table 5.3. Merits and demerits of the MCDM methods considered in this study

MCDM method	Merits	Demerits
EDAS	• Useful when there are conflicting criteria and decision-making fluctuations • Provides realistic solutions as it does not consider extreme ideal points • Operates with a difference from average solution instead of distance • Free from rank reversal issue	• In many real-life cases, the average point does not reveal the true picture • This method is more suited for risk-neutral cases
ARAS	• Simple computational steps with lesser complexity • Can operate under the compromising situation • A relative measurement in terms of the ratio	• ARAS works reasonably well only when the number of alternatives is limited
MABAC	• Stability in result • Systematic computation with a precise and rational solution • Free from rank reversal • Can work with large criteria set	• Does not consider non-compensation of criteria
COPRAS	• Evaluates influence of maximizing and minimizing criteria separately • Simple calculation • Free from rank reversal	• Provides unstable results in case of data variation, and the results may not reveal the true nature of the data
MARCOS	• Consideration of the anti-ideal and ideal solution at the very beginning of the formation of the decision matrix • Determination of utility degree for both solutions, • Can work with a large set of criteria and alternatives • Stability in solution	• Works on compromising results

We considered the widely used MCDM methods as a representation of each above-mentioned class. In Table 5.3, we present a comparative analysis of the merits and demerits of the considered MCDM methods. Since the calculation time is vital in our problem (resource selection in MCC) and subjective bias might affect the final solution, we avoided considering the pairwise comparison methods such as AHP, ANP, ELECTRE, MACBETH, REMBRANDT (multiplicative AHP), PAPRIKA, etc.

5.2.5.1 EDAS Method

EDAS is a recently developed distance-based algorithm that considers the average solution as a reference point [649]. The alternative with a higher favourable deviation, i.e., the positive distance from average (PDA), is preferred compared to non-favourable deviation, i.e., the negative distance from average (NDA). As a result, EDAS provides a reasonably robust solution, free from outlier effect and rank reversal problem, and decision-making fluctuations [661]. However, the EDAS method does not portray a favourable result. Therefore, this method is more suited in the case of risk aversion considerations. The procedural steps of EDAS are described below.

Step 1: Calculation of the average solution

The average solution is the midpoint for all alternatives in the solution space with respect to a particular criterion and is calculated using Eq. 5.1.

$$AV_j = \frac{\sum_{i=1}^{m} x_{ij}}{m}; j = 1, 2, ..., n \tag{5.1}$$

Step 2: Calculation of PDA and NDA

PDA and NDA are the dispersion measures for each possible solution with respect to the average point. An alternative with higher PDA and lower NDA is treated as better than the average one. The PDA and NDA matrices are defined by Eq. 5.2 and Eq. 5.3, respectively.

$$PDA = [PDA_{ij}]_{m\times n} \tag{5.2}$$

$$NDA = [NDA_{ij}]_{m\times n} \tag{5.3}$$

Where, PDA_{ij} and NDA_{ij} are defined by Eq. 5.4 and Eq. 5.5, respectively.

$$PDA_{ij} = \begin{cases} \frac{\max\left(0,(x_{ij} - AV_j)\right)}{AV_j}, \text{if } j^{th} \text{ criterion is profit type} \\ \frac{\max\left(0,(AV_j - x_{ij})\right)}{AV_j}, \text{if } j^{th} \text{ criterion is cost type} \end{cases} \tag{5.4}$$

and

$$NDA_{ij} = \begin{cases} \frac{\max\left(0,(AV_j - x_{ij})\right)}{AV_j}, \text{if } j^{th} \text{ criterion is profit type} \\ \frac{\max\left(0,(x_{ij} - AV_j)\right)}{AV_j}, \text{if } j^{th} \text{ criterion is cost type} \end{cases} \tag{5.5}$$

It can be inferred that if PDA > 0, then the corresponding NDA = 0, and if NDA > 0, then the PDA = 0 for an alternative with respect to a particular criterion.

Step 3: Determine the weighted sum of PDA and NDA for all alternatives

Calculate the weighted sum of PDA and NDA using Eq. 5.6 and Eq. 5.7, respectively.

$$SP_i = \sum_{j=1}^{n} w_j \, PDA_{ij} \tag{5.6}$$

$$SN_i = \sum_{j=1}^{n} w_j \, NDA_{ij} \tag{5.7}$$

where, w_j is the weight of j^{th} criterion.

Step 4: Normalization of the values of SP and SN for all the alternatives

The normalization of linear form for SP and SN values are obtained by using Eq. 5.8 and Eq. 5.9, respectively.

$$NSP_i = \frac{SP_i}{\max_i(SP_i)} \tag{5.8}$$

$$NSN_i = 1 - \frac{SN_i}{\max_i(SN_i)} \tag{5.9}$$

Step 5: Calculation of the appraisal score (AS) for all alternatives

Here the appraisal score denotes the performance score of the alternatives and is calculated using Eq. 5.10.

$$AS_i = \frac{1}{2}(NSP_i + NSN_i) \tag{5.10}$$

where, $0 \leq AS_i \leq 1$. The alternative having the highest AS_i is ranked first and so on.

5.2.5.2 ARAS Method

ARAS method uses the concept of utility values for comparing the alternatives. In this method, a relative scale (i.e., ratio) is used to compare the alternatives with respect to the optimal solution [652] [662] [663]. This method uses a simple additive approach while working under compromising situations effectively and with lesser computational complexities [664] [665]. However, it is observed that ARAS works reasonably well only when the number of alternatives is limited [666]. The procedural steps of ARAS are described below.

Step 1: Formation of the decision matrix

The decision matrix is constructed using Eq. 5.11.

$$X = [x_{ij}]_{m\times n} \tag{5.11}$$

Step 2: Determination of the optimal value

The optimal value for j^{th} criterion is determined using Eq. 5.12.

$$x_{ij} = \begin{cases} \max\limits_{i} x_{ij}, \text{for profit type} \\ \min\limits_{i} x_{ij}, \text{for cost type} \end{cases} \tag{5.12}$$

Step 3: Formation of the normalized decision matrix

The criteria have different dimensions. Normalization is carried out to achieve dimensionless weighted performance values for all alternatives under the influences of the criteria. In this case, we follow a linear ratio approach for normalization. However, we consider the optimum point as the base level. Therefore, in the normalized decision matrix, we include the optimum value, and the order of the matrix is $(m + 1) \times n$. In the ARAS method, a two-stage normalization is followed for the cost type of criteria. The normalized decision matrix is given by Eq. 5.13 where r_{ij} is defined by Eq. 5.14.

$$R = [r_{ij}]_{(m+1)\times n} \tag{5.13}$$

$$r_{ij} = \begin{cases} \frac{x_{ij}}{\sum_{i=0}^{m} x_{ij}}, \text{for profit type criteria} \\ \frac{1/x_{ij}}{\sum_{i=0}^{m} 1/x_{ij}}, \text{for cost type criteria} \end{cases} \tag{5.14}$$

If in case of cost type criteria $x_{ij} = 0$, we consider $r_{ij} = 0$.

Step 4: Derive the weighted normalized decision matrix

The weighted normalized decision matrix is calculated using Eq. 5.15.

$$V = [v_{ij}]_{(m+1)\times n} \tag{5.15}$$

Where v_{ij} is definded by Eq. 5.16 and $i = \overline{0, m}$.

$$v_{ij} = r_{ij} \times w_j \tag{5.16}$$

Step 5: Calculation of the optimality function value for each alternative

The optimality function value is calculated using Eq. 5.17.

$$\emptyset_i = \sum_{j=1}^{n} v_{ij} \tag{5.17}$$

where, $i = \overline{0, m}$.

Higher is the value of $\emptyset_i$, better is the alternative.

Step 6: Find out the priority order of the alternatives based on utility degree with respect to the ideal solution

The priority order is calculated using Eq. 5.18.

$$\partial_i = \frac{\emptyset_i}{\emptyset_0} \tag{5.18}$$

where, $i = \overline{0, m}$ and $\partial_i \in [0,1]$.

Obviously, the bigger value of ∂_i is preferable. It is pretty certain that the optimality function $\emptyset_i$ maintains a direct and proportional relationship with the performance values of the alternatives and weights of the criteria. Hence, the greater the value of $\emptyset_i$, more is the effectiveness of the corresponding solution. The degree of utility is essentially the usefulness of the corresponding alternative with respect to the optimal one.

5.2.5.3 MABAC Method

MABAC uses two areas: an upper approximation area (UAA) for favourable or ideal solutions and a lower approximation area (LAA) for non-favourable or anti-ideal solutions for performance-based classifications of the solutions. This method provides lesser computational complexities compared to the EDAS and ARAS methods. Further, since this method does not involve distance-based separation

measures, it generates stable results [650]. MABAC compares the alternatives based on relative strength and weakness [667]. Because of its simplicity and usefulness, MABAC has been a widely popular method in various applications, for example, social media efficiency measurement [668], health tourism [669], supply chain performance assessment [219], portfolio selection [670], railway management [671], medical tourism site selection [672], and selection of hotels [673]. The procedural steps of MABAC are described below.

Step 1: Normalization of the criteria values

Here, a linear max-min type scheme is used, as given in Eq. 5.19.

$$r_{ij} = \begin{cases} \frac{(x_{ij} - x_i^-)}{(x_i^+ - x_i^-)}, \text{for beneficial criteria} \\ \frac{(x_{ij} - x_i^+)}{(x_i^- - x_i^+)}, \text{for nonbeneficial criteria} \end{cases} \tag{5.19}$$

where, x_i^+ and x_i^- are the maximum and minimum criteria values, respectively.

Step 2: Formulate the weighted normalization matrix (Y)

Elements of Y are given by Eq. 5.20.

$$y_{ij} = w_j(r_{ij} + 1) \tag{5.20}$$

where, w_j is the criteria weight.

Step 3: Determination of the Border Approximation Area (BAA)

The elements of the BAA (T) are denoted by Eq. 5.21 where t_j is given by Eq. 5.22.

$$T = [t_j]_{1\times n} \tag{5.21}$$

$$t_j = \left(\prod_{i=1}^{m} y_{ij}\right)^{1/m} \tag{5.22}$$

where, *m* is the total number of alternatives and t_j corresponds to each criterion.

Step 4: Calculation of the matrix Q related to the separation of the alternatives from BAA

Q is calculated using Eq. 5.23.

$$Q = Y - T \tag{5.23}$$

A particular alternative a_i is said to be belonging to the UAA (i.e., T^+) if $q_{ij} > 0$ or

LAA (i.e., T^-) if $q_{ij} < 0$ or BAA (i.e., T) if $q_{ij} = 0$. The alternative a_i is considered to be the best among the others if more numbers of criteria pertaining to it possibly belong to T^+.

Step 5: Ranking of the alternatives

The ranking is done according to the final values of the criterion functions as given by Eq. 5.24.

$$S_i = \sum_{j=1}^{n} q_{ij} \text{ for } j = 1,2, \dots, n \text{ and } i = 1,2, \dots, m \tag{5.24}$$

The higher the value is, more is the preference.

5.2.5.4 COPRAS Method

The COPRAS method calculates the utility values of the alternatives under the direct and proportional dependencies of the influencing criteria for carrying out preferential ranking [655] [674] [675]. The procedural steps for finding out the utility values of the alternatives using the COPRAS method are discussed in the following. The alternatives are ordered in descending order based on the obtained utility values.

Step 1: Construct the normalized decision matrix using the simple proportional approach

The normalised decision matrix is calculated using Eq. 5.25.

$$\widetilde{d_{ij}} = \frac{d_{ij}}{\sum_{i=1}^{m} d_{ij}} \tag{5.25}$$

where, d_{ij} is the performance value of the i^{th} alternative with respect to j^{th} criterion ($i = 1, 2, \dots, m; j = 1, 2, \dots, n$).

Step 2: Calculation of the sums of the weighted normalized values for optimization in ideal and anti-ideal effects

The ideal and anti-ideal effects are calculated by Eq. 5.26 and Eq. 5.27, respectively.

$$\mathcal{g}_{+i} = \sum_{j=1}^{k} \widetilde{d_{ij}} \cdot \mathcal{E}_j \tag{5.26}$$

$$\mathcal{g}_{-i} = \sum_{j=k+1}^{n} \widetilde{d_{ij}} \cdot \mathcal{E}_j \tag{5.27}$$

where, k is the number of maximizing (i.e., profit type) criteria and $\mathcal{E}_j$ is the

significance of the j^{th} criterion.

In case of g_{+i}, all $\widetilde{d_{ij}}$ values are corresponding to the beneficial or profit type criteria, and for g_{-i}, we take the performance values of the alternatives related to cost type criteria.

Step 3: Calculation of the relative weights of the alternatives

The relative weight for any alternative (i^{th}) is given by Eq. 5.28.

$$\Omega_i = g_{+i} + \frac{\min_i g_{-i} \sum_{i=1}^{m} g_{-i}}{g_{-i} \sum_{i=1}^{m} \frac{\min_i g_{-i}}{g_{-i}}} \cong g_{+i} + \frac{\sum_{i=1}^{m} g_{-i}}{g_{-i} \sum_{i=1}^{m} (\frac{1}{g_{-i}})} \tag{5.28}$$

The Ω_i value corresponding to the i^{th} alternative signifies the degree of satisfaction of that with respect to the given conditions. The greater is the value of Ω_i better is the relative performance of the concerned alternative, and hence, higher is the position. Therefore, the most rational and efficient DMU should have $\Omega_{i\,max}$ i.e., the optimum value. The relative utility of a particular DMU or alternative is determined by comparing the Ω_i value of any DMU with respect to the $\Omega_{i\,max}$ value, corresponding to the most effective one.

The utility for each alternative is given by Eq. 5.29. Needless to say, the U_i value for the most preferred choice is 100%.

$$U_i = \frac{\Omega_i}{\Omega_{i\,max}} \times 100\% \tag{5.29}$$

5.2.5.5 MARCOS Method

MARCOS belongs to a strand of MCDM algorithms that derives solutions under compromise situations. However, unlike the previous versions, MARCOS starts with including ideal and anti-ideal solutions in the fundamental decision matrix at the very beginning. Likewise, COPRAS also finds out the utility values. However, here the decision-maker can make a trade-off among the ideal and anti-ideal solutions to arrive at the utility values of the alternatives. The MARCOS method is also capable of handling a large set of alternatives and criteria [660] [676] [677]. The procedural steps of MARCOS are described below.

Step 1: Formation of the extended decision matrix (D) by including the anti-ideal solution (S^-) values in the first row and the ideal solution (S^+)values in the last row*

S^- and $S^+$are defined by Eq. 5.30 and Eq. 5.31, respectively.

$$S^- = \begin{cases} \min\limits_i x_{ij}, \text{when } j \in \text{profit type} \\ \max\limits_i x_{ij}, \text{when } j \in \text{cost type} \end{cases} \tag{5.30}$$

$$S^+ = \begin{cases} \max\limits_i x_{ij}, \text{when } j \in \text{profit type} \\ \min\limits_i x_{ij}, \text{when } j \in \text{cost type} \end{cases} \tag{5.31}$$

The anti-ideal solution represents the worst choice, whereas the ideal solution is the reference point that shows the best possible characteristics given the set of constraints, i.e., criteria.

*Step 2: Normalization of D**

The normalized values are given by Eq. 5.32.

$$r_{ij} = \begin{cases} \frac{x_{S^+}}{x_{ij}}, \text{when } j \in \text{cost type} \\ \frac{x_{ij}}{x_{S^+}}, \text{when } j \in \text{profit type} \end{cases} \tag{5.32}$$

Since it is preferred to set apart from the anti-ideal reference point, in MARCOS, the normalization is carried out using a linear ratio approach with respect to the anti-ideal solution.

*Step 3: Formation of weighted D**

After normalization, the weighted normalized matrix with elements v_{ij} is formulated by multiplying the normalized value of each alternative with the corresponding weight of the criteria, as given in Eq. 5.33.

$$v_{ij} = w_j r_{ij} \tag{5.33}$$

Step 4: Calculation of utility degrees of the alternatives for S^+ and S^-

The utility degree of a particular alternative with respect to given conditions represents its relative attractiveness of the same. The utility degrees are calculated using Eq. 5.34 and Eq. 5.35 where γ_i is given as Eq. 5.36.

$$K_i^- = \frac{\gamma_i}{\gamma_{S^-}} \tag{5.34}$$

$$K_i^+ = \frac{\gamma_i}{\gamma_{S^+}} \tag{5.35}$$

$$\gamma_i = \sum_{j=1}^{n} v_{ij} \tag{5.36}$$

Step 5: Calculation of values of utility functions for S^+ and S^-

The utility function resembles the trade-off that the observed or considered alternatives make vis-à-vis the ideal and anti-ideal reference points, and are given by Eq. 5.37 and Eq. 5.38.

$$f(K_i^-) = \frac{K_i^+}{K_i^+ + K_i^-} \tag{5.37}$$

$$f(K_i^+) = \frac{K_i^-}{K_i^+ + K_i^-} \tag{5.38}$$

The decision is made related to the selection of a particular alternative is based on utility functional values. The utility function exhibits the relative position of the concerned alternative with respect to the reference points. The best alternative is closest to the ideal reference and, subsequently, distant from the anti-ideal one compared to other available choices.

Step 6: Calculation of the utility function values for the alternatives

The utility function value for i^{th} alternative is calculated by Eq. 5.39.

$$f(K_i) = \frac{K_i^+ + K_i^-}{1 + \frac{1 - f(K_i^+)}{f(K_i^+)} + \frac{1 - f(K_i^-)}{f(K_i^-)}} \tag{5.39}$$

The alternative having the highest utility function value is ranked first over the others.

5.2.6 Entropy Method for Criteria Weight Calculation

Each selection criterion carries some weight. The weights define the importance of the criteria in the decision-making. To determine the criteria weights, we applied the most popularly used entropy method. The entropy method works on objective information following the concept of the probabilistic information theory [678]. The objective weighting approach can mitigate the man-made instabilities in the subjective weighting approach and gives more realistic results [679]. The Entropy method shows its efficacy in dealing with imprecise information and dispersions while offsetting the subjective bias [680] [681]. Extant literature shows a colossal number of applications of Entropy method for determining criteria weights in various situations (for example, [670] [682] [683] [684] [685] [686]). The steps of the

entropy method are given below:

Suppose, $X = [x_{ij}]_{m\times n}$ represents the decision matrix where *m* is the number of alternatives and *n* is the number of criteria.

Step 1: Normalization of the decision matrix

Normalization is carried out to bring the performance values of all alternatives subject to different criteria to a common unitless form having scale values $\epsilon(0,1)$. Here we follow the linear normalization scheme.

Entropy value signifies the level of disorder. In the case of criteria weight determination, a criterion with a higher Entropy value indicates that that particular criterion contains more information.

The normalization matrix is represented as $(R)_{m\times n}$ where the elements r_{ij} are given by Eq. 5.40.

$$r_{ij} = \begin{cases} \frac{(x_{ij} - x_{j_{min}})}{(x_{j_{max}} - x_{j_{min}})}, \text{for profit type criteria} \\ \frac{(x_{j_{max}} - x_{ij})}{(x_{j_{max}} - x_{j_{min}})}, \text{for cost type criteria} \end{cases} \tag{5.40}$$

Step 2: Calculation of Entropy values

The Entropy value for i^{th} alternative for j^{th} criterion is given by Eq. 5.41.

$$H_j = -k\sum_{i=1}^{m} f_{ij} \ln(f_{ij}) \tag{5.41}$$

where, *k* (a constant value) and f_{ij} are defined by Eq. 5.42. and Eq. 5.43, respectively. If $f_{ij} = 0$ then, $f_{ij} \ln(f_{ij}) = 0$.

$$k = 1/\ln(m) \tag{5.42}$$

$$f_{ij} = \frac{r_{ij}}{\sum_{i=1}^{m} r_{ij}} \tag{5.43}$$

Step 3: Calculation of criteria weight

The weight for each criterion is given by Eq. 5.44.

$$w_j = \frac{1 - H_j}{n - \sum_{j=1}^{n} H_j} \tag{5.44}$$

Here, the higher the value of w_j is , more is the information contained in the j^{th} criterion.

5.3 Research Methodology

This section discusses the research framework used in this chapter and provides the computational steps of the MCDM algorithms applied for carrying out the comparative analysis in a dynamic environment. Fig. 5.3 depicts the steps followed in this research work.

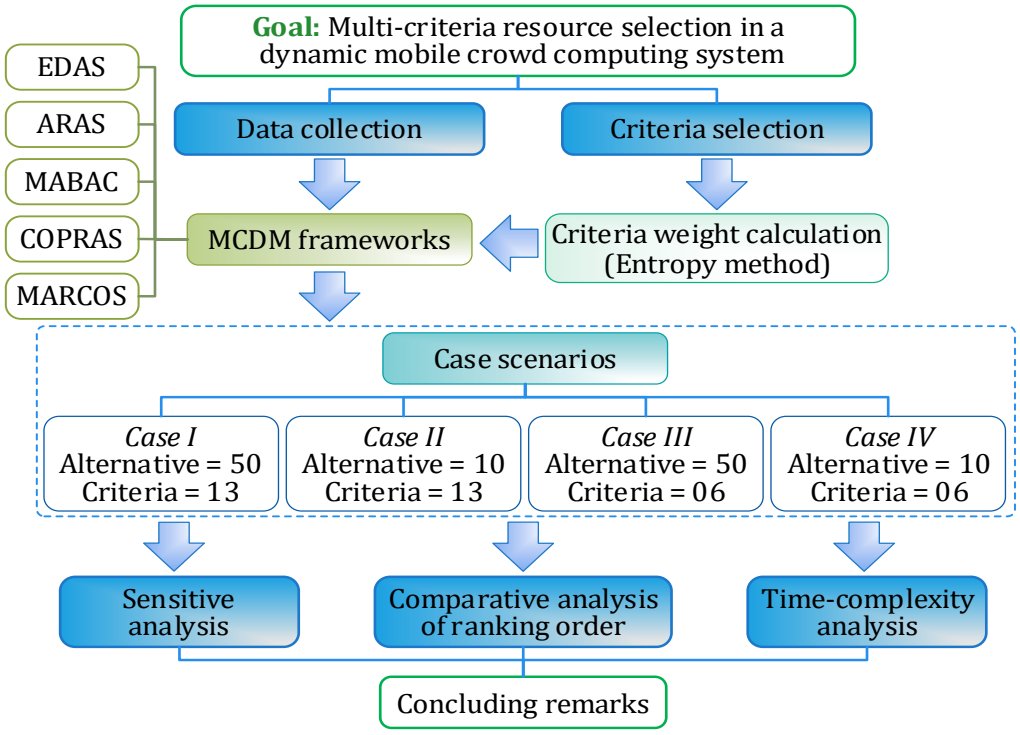


Fig. 5.3. Research framework

5.3.1 Resource Selection Criteria

Here, we considered thirteen criteria for SMD selection, as shown in Table 5.4. Out of these, eight are profit criteria, i.e., their maximized values would be ideal for selection, whereas five are cost criteria, i.e., their minimized values should be ideal. Depending on specific applications and specific job types, the criteria and their weights (significance) would vary. For example, a CPU-bound job may not use GPU cores, while some highly computing-intensive jobs (such as image and video analysis, complex scientific calculations, etc.) would use GPU more than the CPU. Similarly, the RAM size would be a decisive factor for a data-intensive job that might not be so important for a CPU-intensive job. Here, we chose the criteria that would, in general and overall, be considered for selecting an SMD as a computing resource.

To keep the experiment minimalistic, we ignored several resource parameters such as persistent parameters and benchmarking that are discussed in Section 4.3. However, in practical implementation, they can be incorporated as selection criteria.

Table 5.4. List of selection criteria

	Profit type								Cost type				
Criteria	CPU frequency (GHz)	CPU cores (in numbers)	GPU frequency (GHz)	Total RAM (GB)	Available memory (MB)	Battery capacity (mAh)	Battery available (%)	Wi-Fi strength (1-5)	CPU load (%)	GPU load (%)	CPU temp (°C)	Battery temp (°C)	GPU Architecture (nm)
Code	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}
Effect direction	(+)	(+)	(+)	(+)	(+)	(+)	(+)	(+)	(–)	(–)	(–)	(–)	(–)

5.3.2 Data Collection

The data collection for this experiment was done through the resource profiling, as discussed in Chapter 4. From the logged user data, we picked the users who were more consistent with high presence frequency and less sparsity. For this study, we considered such 50 SMDs, selected randomly. Specifically for this experiment, we considered a total of thirteen resource parameters that are important in the decision-making process for selecting an SMD as a suitable resource in MCC, as shown in Table 5.4. It can be seen from the table that some resource parameters are fixed, i.e., they would not change their values in their lifetime (e.g., C_1, C_2, C_3, C_4, C_6, and C_{13}), while some parameters' values are changed dynamically (e.g., C_5, C_7, C_8, C_9, C_{10}, C_{11}, and C_{12}). We considered some instantaneous values of all the parameters and used the same for all experimental illustrations for the experimental purpose.

5.3.3 Experiment Cases

Since, in this study, we wanted to assess the effect of the number of criteria and alternatives in the selection outcome and computational complexity; we considered different variations of the selection criteria and alternatives for comparison. Accordingly, we generated four case scenarios, as discussed in the following subsections. Each case has a different number of alternatives (SMDs) and criteria. The reason behind choosing four datasets of different sizes is to assess the performance

of the MCDM methods under different MCC scenarios.

Table 5.5. Decision matrix (Case 1)

SMD	Profit criteria								Cost criteria				
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}
M_1	2.2	2	650	8	895	2700	15	4	92	27	43	45	14
M_2	1.5	4	450	4	3831	4000	39	4	16	76	39	40	10
M_3	1.5	2	650	6	2694	2700	12	3	44	67	38	40	28
M_4	1.3	8	650	8	518	4000	11	5	89	78	42	42	10
M_5	1.3	8	650	8	1807	3000	10	4	13	8	31	38	10
M_6	1.7	8	450	8	1982	3000	68	5	64	32	32	35	14
M_7	2.5	2	400	6	3857	3500	18	1	60	16	38	36	10
M_8	2.5	4	624	8	558	4000	56	5	99	87	50	48	10
M_9	1.7	2	450	8	1908	2700	57	4	26	4	30	34	28
M_{10}	2.5	2	450	6	1767	4000	24	2	53	93	45	44	10
M_{11}	2.5	2	400	4	2853	4000	94	3	53	47	40	40	10
M_{12}	2.2	2	624	6	3535	2700	24	3	26	67	37	39	28
M_{13}	2.2	8	710	4	1734	3500	50	1	19	63	34	38	28
M_{14}	1.5	8	650	4	2954	3000	59	5	15	3	34	33	10
M_{15}	2.2	8	650	6	1916	3000	11	1	19	77	32	39	14
M_{16}	1.3	2	400	6	870	2700	90	5	44	89	35	43	10
M_{17}	1.5	4	400	4	2911	3500	17	2	18	96	36	47	10
M_{18}	1.7	8	450	6	3876	4000	63	4	4	0	45	42	10
M_{19}	1.3	4	650	6	944	2700	75	1	2	72	30	43	14
M_{20}	1.7	2	450	6	2855	4000	22	5	62	9	32	40	10
M_{21}	1.3	4	450	6	2973	3500	18	1	78	92	40	45	14
M_{22}	1.5	8	624	8	3521	4000	22	1	42	44	38	37	10
M_{23}	1.3	4	400	6	1734	3500	84	4	95	24	43	39	28
M_{24}	2.5	2	710	4	3986	3000	16	1	8	57	36	40	28
M_{25}	1.5	4	624	6	2851	3500	31	4	71	2	39	42	10
M_{26}	1.7	4	710	6	2983	3000	50	1	61	58	38	45	10
M_{27}	2.2	2	710	8	1932	4000	87	3	57	21	39	43	14
M_{28}	2.5	2	624	6	972	4000	87	5	77	80	43	46	28
M_{29}	1.3	2	710	6	2579	4000	16	2	69	0	41	40	14
M_{30}	1.3	4	710	6	3537	3500	37	2	4	16	37	37	28
M_{31}	2.5	2	650	4	809	2700	89	5	70	3	41	39	14
M_{32}	1.3	4	450	4	3769	3500	56	2	5	35	33	40	28
M_{33}	1.3	8	400	4	799	3000	39	1	65	47	35	44	10
M_{34}	2.2	4	710	4	1938	4000	17	5	48	11	36	40	28
M_{35}	1.3	8	710	6	2755	3000	92	4	1	48	34	39	14
M_{36}	1.3	2	450	4	2663	2700	30	1	56	46	37	41	10
M_{37}	2.5	8	450	4	1789	2700	12	2	4	15	32	36	14
M_{38}	1.3	4	710	6	759	3500	44	2	66	0	34	35	28
M_{39}	2.2	4	400	4	1748	3000	58	5	99	22	45	44	10
M_{40}	1.3	8	450	8	2690	4000	56	4	22	13	33	34	28
M_{41}	1.5	8	624	8	898	3500	82	4	47	22	34	36	10
M_{42}	2.5	2	450	8	3681	3000	62	5	26	68	35	37	28
M_{43}	1.3	8	624	8	2790	4000	16	3	84	15	37	39	14
M_{44}	1.3	8	400	4	1582	3000	26	4	18	0	32	33	14
M_{45}	2.5	8	650	4	2628	3500	69	4	94	11	42	40	28
M_{46}	2.5	2	400	6	619	3000	52	2	40	52	41	39	14
M_{47}	1.3	2	400	6	2760	2700	69	1	31	38	37	38	10
M_{48}	2.5	8	624	8	1673	2700	29	5	26	7	35	36	28
M_{49}	1.7	4	650	4	1647	3000	48	3	43	0	34	37	10
M_{50}	1.3	8	450	6	1753	4000	29	3	91	64	39	45	28

5.3.3.1 Case 1: Full List of Alternatives and Full Criteria Set

This scenario considers the full list of alternatives under comparison (i.e., 50) subject to the influence of full criteria set consisting of 13 different criteria, as shown in Table 5.4. Accordingly, the decision matrix (50 × 13) is given in Table 5.5.

5.3.3.2 Case 2: Lesser Number of Alternatives and Full Criteria Set

In this minimized dataset, we assume that only ten SMDs available for crowd computing (typically in a small-scale MCC). In this case, we shortened the number of alternatives. Here, the decision-maker would be able to compare the MCDM methods on a limited number of alternatives for the full list of criteria. For simplicity, we selected one SMD model out of each group of five starting from the beginning, i.e., M_5, M_{10}, M_{15}, and so on. The decision matrix (10 × 13) is given in Table 5.6.

Table 5.6. Decision matrix (Case 2)

SMD	Profit criteria								Cost criteria				
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}
M_5	1.3	8	650	8	1807	3000	10	4	13	8	31	38	10
M_{10}	2.5	2	450	6	1767	4000	24	2	53	93	45	44	10
M_{15}	2.2	8	650	6	1916	3000	11	1	19	77	32	39	14
M_{20}	1.7	2	450	6	2855	4000	22	5	62	9	32	40	10
M_{25}	1.5	4	624	6	2851	3500	31	4	71	2	39	42	10
M_{30}	1.3	4	710	6	3537	3500	37	2	4	16	37	37	28
M_{35}	1.3	8	710	6	2755	3000	92	4	1	48	34	39	14
M_{40}	1.3	8	450	8	2690	4000	56	4	22	13	33	34	28
M_{45}	2.5	8	650	4	2628	3500	69	4	94	11	42	40	28
M_{50}	1.3	8	450	6	1753	4000	29	3	91	64	39	45	28

5.3.3.3 Case 3: Full List of Alternatives and a Smaller Number of Criteria

In a situation, depending on the MCC application requirement, the full criteria set may not need to be considered. For these cases, only a small number of crucial criteria may be defined. To represent such a scenario, in this case, we considered a minimized dataset by eliminated some criteria from the original dataset. We assumed that some criteria (e.g., CPU and battery temperature and signal strength) could be kept out of the selection matrix and, if required, could be set as threshold criteria straightforwardly. For example, suppose the threshold for temperature is set at 40°C. In that case, all the SMDs having a temperature more than this would be filtered out and would not be considered for the selection, irrespective of other resource specifications. We also removed information of GPU, assuming that the tasks are CPU bound only and they do not require to exploit the power of GPU,

i.e., the jobs are sequential and not parallel. It can also be vice versa, i.e., we could consider GPU where the MCC job involves mostly parallel processing. Table 5.7 shows the criteria considered, and in Table 5.8, the decision matrix (50 × 6) is presented.

Table 5.7. Minimized selection criteria

	Profit					Cost
Criteria	CPU frequency (GHz)	CPU cores (in numbers)	Total RAM (GB)	Battery capacity (mAh)	Battery available (%)	CPU load (%)
Code	C_1	C_2	C_4	C_6	C_7	C_9
Effect direction	(+)	(+)	(+)	(+)	(+)	(–)

Table 5.8. Decision matrix (Case 3)

SMD	Profit					Cost
	C_1	C_2	C_4	C_6	C_7	C_9
M_1	2.2	2	895	2700	15	92
M_2	1.5	4	3831	4000	39	16
M_3	1.5	2	2694	2700	12	44
M_4	1.3	8	518	4000	11	89
M_5	1.3	8	1807	3000	10	13
M_6	1.7	8	1982	3000	68	64
M_7	2.5	2	3857	3500	18	60
M_8	2.5	4	558	4000	56	99
M_9	1.7	2	1908	2700	57	26
M_{10}	2.5	2	1767	4000	24	53
M_{11}	2.5	2	2853	4000	94	53
M_{12}	2.2	2	3535	2700	24	26
M_{13}	2.2	8	1734	3500	50	19
M_{14}	1.5	8	2954	3000	59	15
M_{15}	2.2	8	1916	3000	11	19
M_{16}	1.3	2	870	2700	90	44
M_{17}	1.5	4	2911	3500	17	18
M_{18}	1.7	8	3876	4000	63	4
M_{19}	1.3	4	944	2700	75	2
M_{20}	1.7	2	2855	4000	22	62
M_{21}	1.3	4	2973	3500	18	78
M_{22}	1.5	8	3521	4000	22	42
M_{23}	1.3	4	1734	3500	84	95
M_{24}	2.5	2	3986	3000	16	8
M_{25}	1.5	4	2851	3500	31	71

SMD	Profit					Cost
	C_1	C_2	C_4	C_6	C_7	C_9
M_{26}	1.7	4	2983	3000	50	61
M_{27}	2.2	2	1932	4000	87	57
M_{28}	2.5	2	972	4000	87	77
M_{29}	1.3	2	2579	4000	16	69
M_{30}	1.3	4	3537	3500	37	4
M_{31}	2.5	2	809	2700	89	70
M_{32}	1.3	4	3769	3500	56	5
M_{33}	1.3	8	799	3000	39	65
M_{34}	2.2	4	1938	4000	17	48
M_{35}	1.3	8	2755	3000	92	1
M_{36}	1.3	2	2663	2700	30	56
M_{37}	2.5	8	1789	2700	12	4
M_{38}	1.3	4	759	3500	44	66
M_{39}	2.2	4	1748	3000	58	99
M_{40}	1.3	8	2690	4000	56	22
M_{41}	1.5	8	898	3500	82	47
M_{42}	2.5	2	3681	3000	62	26
M_{43}	1.3	8	2790	4000	16	84
M_{44}	1.3	8	1582	3000	26	18
M_{45}	2.5	8	2628	3500	69	94
M_{46}	2.5	2	619	3000	52	40
M_{47}	1.3	2	2760	2700	69	31
M_{48}	2.5	8	1673	2700	29	26
M_{49}	1.7	4	1647	3000	48	43
M_{50}	1.3	8	1753	4000	29	91

5.3.3.4 Case 4: Lesser Number of Alternatives and Criteria

In this case, we considered the combination of a minimized set of alternatives and criteria. This scenario considers a limited number of choices and the influence of

a limited number of criteria. We considered the alternatives as selected in Case 2 and the criteria as listed in Table 5.7. Hence, in this case, our decision matrix is of dimension 10 × 6, as shown in Table 5.9.

Table 5.9. Decision matrix (Case 4)

SMD	Profit					Cost
	C_1	C_2	C_4	C_6	C_7	C_9
M_1	1.3	8	1807	3000	10	13
M_{10}	2.5	2	1767	4000	24	53
M_{15}	2.2	8	1916	3000	11	19
M_{20}	1.7	2	2855	4000	22	62
M_{25}	1.5	4	2851	3500	31	71
M_{30}	1.3	4	3537	3500	37	4
M_{35}	1.3	8	2755	3000	92	1
M_{40}	1.3	8	2690	4000	56	22
M_{45}	2.5	8	2628	3500	69	94
M_{50}	1.3	8	1753	4000	29	91

5.4 Experiment, Results, and Comparative Analysis

In this section, we present the details of the experiment for the comparative study, including the results and critical discussion. The experiment focuses on the comparative ranking for the SMD selection using five distinct MCDM methods and to find their time complexities under different scenarios by varying the criteria and/or alternative sets.

5.4.1 Experiment

We applied the entropy method and the five MCDM methods (i.e., EDAS, ARAS, MABAC, COPRAS, and MARCOS) on four datasets, as discussed in Section 5.3.3. The algorithms were implemented using a spreadsheet (MS Excel) as well as through hand-coded programming (using Java). However, for ranking and sensitivity analysis, we used the spreadsheet calculation, and to estimate the runtime, we considered the programming outturn. The details of the programmatical implementation are discussed in Section 5.4.4. The aggregate rankings of the SMDs were derived from each MCDM method for each dataset. We checked the consistencies among the results of the individual MCDM methods and the final aggregate ranks. We also compared the robustness and stability in the performance of the MCDM methods applied in this work. Finally, the actual runtimes of each method under different scenarios were calculated.

5.4.2 Results

In this section, we report the details of the experimental results of SMD rankings using the considered MCDM methods, obtained through the spreadsheet

calculation.

Table 5.10 shows the criteria weights calculated for Case 1 using the Entropy method where $\sum w_j = 1$ and C_j represents the criteria, where j = 1, 2, 3, ..., 13. It is seen that the weights of the criteria are reasonably distributed. However, based on the values of the decision matrix, the Entropy method calculates higher weights (>10%) for C_1, C_2, and C_4 while assigning the least weights to C_{11} and C_{12}.

Table 5.10. Criteria weights (Case 1)

Criteria	(+) C_1	(+) C_2	(+) C_3	(+) C_4	(+) C_5	(+) C_6	(+) C_7	(+) C_8	(–) C_9	(–) C_{10}	(–) C_{11}	(–) C_{12}	(–) C_{13}
H_j	0.8436	0.8556	0.8985	0.8862	0.9456	0.8998	0.9178	0.9128	0.9498	0.9552	0.9816	0.9696	0.8996
w_j	0.1442	0.1332	0.0936	0.1050	0.0501	0.0924	0.0758	0.0804	0.0463	0.0414	0.0170	0.0281	0.0926

Table 5.11. Ranking results of EDAS method (Case 1)

SMD	SP	SN	NSP	NSN	AS	Rank	SMD	SP	SN	NSP	NSN	AS	Rank
M_1	0.137	0.227	0.423	0.256	0.340	35	M_{26}	0.092	0.131	0.283	0.569	0.426	29
M_2	0.145	0.146	0.446	0.521	0.484	25	M_{27}	0.221	0.100	0.680	0.672	0.676	10
M_3	0.031	0.269	0.096	0.117	0.106	50	M_{28}	0.209	0.249	0.644	0.184	0.414	31
M_4	0.251	0.224	0.771	0.266	0.518	21	M_{29}	0.111	0.218	0.343	0.284	0.314	41
M_5	0.277	0.117	0.852	0.616	0.734	7	M_{30}	0.131	0.164	0.403	0.464	0.433	28
M_6	0.246	0.057	0.758	0.811	0.785	5	M_{31}	0.251	0.185	0.772	0.392	0.582	14
M_7	0.165	0.217	0.508	0.289	0.398	32	M_{32}	0.105	0.202	0.324	0.339	0.331	36
M_8	0.230	0.174	0.708	0.429	0.568	15	M_{33}	0.131	0.236	0.403	0.226	0.315	40
M_9	0.146	0.188	0.450	0.383	0.416	30	M_{34}	0.156	0.171	0.480	0.440	0.460	27
M_{10}	0.115	0.241	0.354	0.211	0.283	44	M_{35}	0.298	0.059	0.919	0.806	0.862	3
M_{11}	0.210	0.157	0.648	0.486	0.567	16	M_{36}	0.048	0.283	0.146	0.070	0.108	49
M_{12}	0.098	0.225	0.300	0.261	0.281	45	M_{37}	0.238	0.163	0.732	0.465	0.599	13
M_{13}	0.195	0.187	0.601	0.386	0.493	23	M_{38}	0.079	0.204	0.243	0.330	0.287	43
M_{14}	0.311	0.066	0.957	0.782	0.870	2	M_{39}	0.159	0.159	0.490	0.478	0.484	24
M_{15}	0.190	0.170	0.583	0.444	0.514	22	M_{40}	0.259	0.119	0.796	0.610	0.703	8
M_{16}	0.168	0.247	0.517	0.189	0.353	34	M_{41}	0.292	0.054	0.897	0.823	0.860	4
M_{17}	0.086	0.246	0.265	0.193	0.229	47	M_{42}	0.229	0.197	0.705	0.353	0.529	19
M_{18}	0.325	0.030	1.000	0.902	0.951	1	M_{43}	0.214	0.129	0.660	0.577	0.619	12
M_{19}	0.132	0.199	0.408	0.346	0.377	33	M_{44}	0.208	0.155	0.639	0.492	0.566	17
M_{20}	0.155	0.156	0.476	0.489	0.482	26	M_{45}	0.273	0.145	0.839	0.524	0.682	9
M_{21}	0.039	0.272	0.120	0.110	0.115	48	M_{46}	0.094	0.194	0.289	0.365	0.327	38
M_{22}	0.233	0.123	0.718	0.597	0.658	11	M_{47}	0.110	0.215	0.339	0.296	0.317	39
M_{23}	0.112	0.210	0.344	0.312	0.328	37	M_{48}	0.306	0.119	0.941	0.611	0.776	6
M_{24}	0.162	0.305	0.499	0.000	0.250	46	M_{49}	0.107	0.087	0.330	0.716	0.523	20
M_{25}	0.132	0.094	0.406	0.692	0.549	18	M_{50}	0.113	0.236	0.347	0.227	0.287	42

We used these criteria weights to rank the alternatives based on the decision matrix of Table 5.5, applying the five MCDM methods considered in this work. Table 5.11 to Table 5.15 present the rankings of the alternatives based on the final score values as derived by using the five MCDM algorithms. From Table 5.11, we observe that considering the average solution point as the reference, M_{18}, M_{14}, M_{35}, M_{41}, and

M_6 are the top performers while proportional assessment methods such as ARAS and COPRAS respectively yield M_{35}, M_{14}, M_{25}, M_{18}, M_{31} and M_{18}, M_{14}, M_{41}, M_{35}, M_5 as better performers (see Table 5.12 and Table 5.14). It is observed that the top-performing DMUs show reasonable consistency. However, Table 5.13 and Table 5.15 show that the relative ranking results derived by MABAC and MARCOS are weekly consistent with previous rankings.

To find out the aggregate ranking, we used the final score values of the alternatives as obtained using different algorithms and applied the SAW method [687] for objective evaluation as adopted in [219]. Table 5.16 exhibits the relative positioning of the alternatives by different MCDM methods and their aggregate ranks derived by using SAW. In this context, Table 5.17 shows the findings of the rank correlation tests among the results obtained by using different methods and the final rank obtained by SAW. For this, we derived the following two correlation coefficients:

Table 5.12. Ranking results of ARAS method (Case 1)

SMD	Ø	∂	Rank	SMD	Ø	∂	Rank
M_1	0.01697	0.46824	38	M_{26}	0.01802	0.49730	30
M_2	0.01865	0.51477	29	M_{27}	0.02042	0.56363	15
M_3	0.01437	0.39669	49	M_{28}	0.01901	0.52454	25
M_4	0.02016	0.55639	18	M_{29}	0.01512	0.41734	47
M_5	0.02201	0.60750	9	M_{30}	0.01941	0.53561	22
M_6	0.02186	0.60323	10	M_{31}	0.02316	0.63903	5
M_7	0.01752	0.48363	33	M_{32}	0.01761	0.48596	32
M_8	0.02111	0.58266	12	M_{33}	0.01664	0.45911	40
M_9	0.01995	0.55052	19	M_{34}	0.01865	0.51482	28
M_{10}	0.01689	0.46600	39	M_{35}	0.03144	0.86779	1
M_{11}	0.01950	0.53823	20	M_{36}	0.01393	0.38446	50
M_{12}	0.01632	0.45037	42	M_{37}	0.02093	0.57773	14
M_{13}	0.01887	0.52081	26	M_{38}	0.01531	0.42259	46
M_{14}	0.02638	0.72790	2	M_{39}	0.01910	0.52707	23
M_{15}	0.01877	0.51807	27	M_{40}	0.02123	0.58593	11
M_{16}	0.01752	0.48362	34	M_{41}	0.02280	0.62921	7
M_{17}	0.01594	0.43998	45	M_{42}	0.01942	0.53585	21
M_{18}	0.02423	0.66882	4	M_{43}	0.02035	0.56156	17
M_{19}	0.02105	0.58100	13	M_{44}	0.01763	0.48661	31
M_{20}	0.01907	0.52618	24	M_{45}	0.02218	0.61221	8
M_{21}	0.01491	0.41141	48	M_{46}	0.01614	0.44552	43
M_{22}	0.02042	0.56351	16	M_{47}	0.01602	0.44206	44
M_{23}	0.01741	0.48050	36	M_{48}	0.02295	0.63345	6
M_{24}	0.01636	0.45145	41	M_{49}	0.01745	0.48152	35
M_{25}	0.02523	0.69635	3	M_{50}	0.01735	0.47875	37

Table 5.13. Ranking results of MABAC method (Case 1)

SMD	Sum (S_i)	Rank	SMD	Sum (S_i)	Rank
M_1	0.03195	27	M_{26}	0.03081	30
M_2	0.03147	28	M_{27}	0.22863	5
M_3	-0.15444	49	M_{28}	0.09664	17
M_4	0.16694	13	M_{29}	0.00047	33
M_5	0.17633	10	M_{30}	0.00290	32
M_6	0.18871	8	M_{31}	0.08230	21
M_7	0.04362	25	M_{32}	-0.11850	46
M_8	0.22907	3	M_{33}	-0.10883	44
M_9	-0.03533	36	M_{34}	0.08986	19
M_{10}	0.03880	26	M_{35}	0.19310	7
M_{11}	0.10172	16	M_{36}	-0.22082	50
M_{12}	-0.04397	39	M_{37}	0.07870	23
M_{13}	0.08626	20	M_{38}	-0.04703	40
M_{14}	0.18429	9	M_{39}	0.00801	31
M_{15}	0.11832	15	M_{40}	0.14808	14
M_{16}	-0.08972	43	M_{41}	0.25900	1
M_{17}	-0.11263	45	M_{42}	0.09494	18
M_{18}	0.24866	2	M_{43}	0.17503	11
M_{19}	-0.05184	41	M_{44}	-0.00397	34
M_{20}	0.06734	24	M_{45}	0.17100	12
M_{21}	-0.13421	48	M_{46}	-0.02276	35
M_{22}	0.20566	6	M_{47}	-0.12598	47
M_{23}	-0.08945	42	M_{48}	0.22869	4
M_{24}	-0.04221	37	M_{49}	0.03112	29
M_{25}	0.08176	22	M_{50}	-0.04263	38

Table 5.14. Ranking results of COPRAS method (Case 1)

SMD	Q	U	Rank	SMD	Q	U	Rank
M_1	0.01794	64.91172	37	M_{26}	0.01886	68.24418	30
M_2	0.01965	71.09344	27	M_{27}	0.02207	79.85614	10
M_3	0.01551	56.09728	48	M_{28}	0.02011	72.76279	23
M_4	0.02044	73.93551	21	M_{29}	0.01756	63.53210	41
M_5	0.02449	88.60817	5	M_{30}	0.01899	68.70255	29
M_6	0.02342	84.72597	6	M_{31}	0.02099	75.94223	16
M_7	0.01880	68.01316	32	M_{32}	0.01776	64.24743	39
M_8	0.02126	76.91708	15	M_{33}	0.01755	63.47324	42
M_9	0.01880	68.01529	31	M_{34}	0.01948	70.48546	28
M_{10}	0.01728	62.49777	45	M_{35}	0.02465	89.15783	4
M_{11}	0.02073	74.99014	18	M_{36}	0.01493	54.02594	50
M_{12}	0.01752	63.39446	43	M_{37}	0.02205	79.76002	11
M_{13}	0.02011	72.76578	22	M_{38}	0.01728	62.52820	44
M_{14}	0.02647	95.76985	2	M_{39}	0.01965	71.09748	26
M_{15}	0.02004	72.50354	24	M_{40}	0.02245	81.21784	9
M_{16}	0.01811	65.51145	36	M_{41}	0.02469	89.32687	3
M_{17}	0.01645	59.51517	47	M_{42}	0.02070	74.87158	19
M_{18}	0.02764	100	1	M_{43}	0.02136	77.25686	14
M_{19}	0.01833	66.30935	35	M_{44}	0.02174	78.65256	13
M_{20}	0.01987	71.89452	25	M_{45}	0.02273	82.22710	8
M_{21}	0.01549	56.05236	49	M_{46}	0.01765	63.84701	40
M_{22}	0.02194	79.37010	12	M_{47}	0.01779	64.37003	38
M_{23}	0.01837	66.47119	34	M_{48}	0.02340	84.64196	7
M_{24}	0.01699	61.45021	46	M_{49}	0.02090	75.59566	17
M_{25}	0.02059	74.47980	20	M_{50}	0.01838	66.47730	33

Table 5.15. Ranking results of MARCOS method (Case 1)

SMD	$f(K_i^-)$	$f(K_i^+)$	$f(K_i)$	Rank	SMD	$f(K_i^-)$	$f(K_i^+)$	$f(K_i)$	Rank
M_1	0.22525	0.77475	0.56639	21	M_{26}	0.22525	0.77475	0.50176	31
M_2	0.22525	0.77475	0.44928	36	M_{27}	0.22525	0.77475	0.74105	5
M_3	0.22525	0.77475	0.46898	34	M_{28}	0.22525	0.77475	0.86193	2
M_4	0.22525	0.77475	0.71421	8	M_{29}	0.22525	0.77475	0.44699	37
M_5	0.22525	0.77475	0.52483	27	M_{30}	0.22525	0.77475	0.54493	26
M_6	0.22525	0.77475	0.66153	14	M_{31}	0.22525	0.77475	0.54586	25
M_7	0.22525	0.77475	0.43151	40	M_{32}	0.22525	0.77475	0.42421	41
M_8	0.22525	0.77475	0.85395	3	M_{33}	0.22525	0.77475	0.31499	48
M_9	0.22525	0.77475	0.48326	33	M_{34}	0.22525	0.77475	0.70693	10
M_{10}	0.22525	0.77475	0.54869	23	M_{35}	0.22525	0.77475	0.63373	17
M_{11}	0.22525	0.77475	0.54848	24	M_{36}	0.22525	0.77475	0.15851	50
M_{12}	0.22525	0.77475	0.57561	19	M_{37}	0.22525	0.77475	0.44642	38
M_{13}	0.22525	0.77475	0.71049	9	M_{38}	0.22525	0.77475	0.52343	28
M_{14}	0.22525	0.77475	0.51506	29	M_{39}	0.22525	0.77475	0.48990	32
M_{15}	0.22525	0.77475	0.58988	18	M_{40}	0.22525	0.77475	0.71645	7
M_{16}	0.22525	0.77475	0.35342	45	M_{41}	0.22525	0.77475	0.67559	13
M_{17}	0.22525	0.77475	0.32342	47	M_{42}	0.22525	0.77475	0.73176	6
M_{18}	0.22525	0.77475	0.64073	16	M_{43}	0.22525	0.77475	0.67850	12
M_{19}	0.22525	0.77475	0.37309	44	M_{44}	0.22525	0.77475	0.33304	46
M_{20}	0.22525	0.77475	0.46101	35	M_{45}	0.22525	0.77475	0.87019	1
M_{21}	0.22525	0.77475	0.41076	42	M_{46}	0.22525	0.77475	0.43541	39
M_{22}	0.22525	0.77475	0.64097	15	M_{47}	0.22525	0.77475	0.22286	49
M_{23}	0.22525	0.77475	0.56692	20	M_{48}	0.22525	0.77475	0.82558	4
M_{24}	0.22525	0.77475	0.54920	22	M_{49}	0.22525	0.77475	0.37653	43
M_{25}	0.22525	0.77475	0.50493	30	M_{50}	0.22525	0.77475	0.67977	11

***Kendall's τ*:** Let, $\{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ is a set of observations for two random variables A and B where all a_i and b_i ($i = 1, 2, \ldots, n$) values are unique. Then, Kendall's τ is calculated by Eq. 5.45.

$$\tau = \frac{(No.\ of\ concordent\ pairs) - (No.\ of\ discordent\ pairs)}{\binom{n}{2}} \tag{5.45}$$

***Spearman's ρ*:** Any pair (a_i, b_i) and (a_j, b_j) where $i < j$ is said to be concordant if either both $a_i > a_j$ and $b_i > b_j$ or $a_i < a_j$ and $b_i < b_j$ hold good. The Spearman's ρ is calculated by Eq. 5.46.

$$\rho = 1 - \frac{6 \sum d_i^2}{n\,(n^2 - 1)} \tag{5.46}$$

here, d_i is the difference between two ranks of each observation, and n is the number of observations.

The aggregated final rank in terms of consistency (of both Kendall's τ and Spearman's ρ) for Case 1 is: MABAC > COPRAS > EDAS > ARAS > MARCOS, as shown in Table 5.17. Similarly, we derived the ranking of alternatives subject to the influence of the criteria for the other cases (Case 2 to 4).

Table 5.16. Comparative analysis of the rankings by different MCDM methods (Case 1)

SMD	Ranking results					Final rank (SAW)
	EDAS	ARAS	MABAC	COPRAS	MARCOS	
M_1	35	38	27	37	21	33
M_2	25	29	28	27	36	27
M_3	50	49	49	48	34	48
M_4	21	18	13	21	8	14
M_5	7	9	10	5	27	10
M_6	5	10	8	6	14	7
M_7	32	33	25	32	40	32
M_8	15	12	3	15	3	8
M_9	30	19	36	31	33	31
M_{10}	44	39	26	45	23	35
M_{11}	16	20	16	18	24	21
M_{12}	45	42	39	43	19	38
M_{13}	23	26	20	22	9	20
M_{14}	2	2	9	2	29	4
M_{15}	22	27	15	24	18	22
M_{16}	34	34	43	36	45	43
M_{17}	47	45	45	47	47	47
M_{18}	1	4	2	1	16	1
M_{19}	33	13	41	35	44	36
M_{20}	26	24	24	25	35	24
M_{21}	48	48	48	49	42	49
M_{22}	11	16	6	12	15	12
M_{23}	37	36	42	34	20	37
M_{24}	46	41	37	46	22	40
M_{25}	18	3	22	20	30	16
M_{26}	29	30	30	30	31	30
M_{27}	10	15	5	10	5	9
M_{28}	31	25	17	23	2	18
M_{29}	41	47	33	41	37	41
M_{30}	28	22	32	29	26	26
M_{31}	14	5	21	16	25	15
M_{32}	36	32	46	39	41	44
M_{33}	40	40	44	42	48	45
M_{34}	27	28	19	28	10	23
M_{35}	3	1	7	4	17	2
M_{36}	49	50	50	50	50	50
M_{37}	13	14	23	11	38	19
M_{38}	43	46	40	44	28	42
M_{39}	24	23	31	26	32	25
M_{40}	8	11	14	9	7	11
M_{41}	4	7	1	3	13	3
M_{42}	19	21	18	19	6	17
M_{43}	12	17	11	14	12	13
M_{44}	17	31	34	13	46	29
M_{45}	9	8	12	8	1	6
M_{46}	38	43	35	40	39	39
M_{47}	39	44	47	38	49	46
M_{48}	6	6	4	7	4	5
M_{49}	20	35	29	17	43	28
M_{50}	42	37	38	33	11	34

Table 5.17. Correlation test I (Case 1)

Coefficient	Final rank	EDAS rank	ARAS rank	MABAC rank	COPRAS rank	MARCOS rank
Kendall's tau	SAW_Rank	.817**	.778**	.829**	.830**	.510**
Spearman's rho	SAW_Rank	.947**	.917**	.960**	.951**	.704**
Aggregated final rank		0.882	0.848	0.8945	0.8905	0.607

** Correlation is significant at the 0.01 level (2-tailed).

Table 5.18 to Table 5.20 show the criteria weights for Case 2-4 as derived from the performance values of the alternatives subject to influences of the criteria involved. In Case 2, we used the full set of criteria but a reduced number of alternatives, while in Case 3, we used the full set of alternatives subject to a reduced set of criteria. In Case 4, we considered a reduced set for both alternatives and criteria. It may be noted from Table 5.18 that C_1, C_2, and C_{13} obtain higher weights (more than 10%) while C4 and C_8 are holding the least weight. It suggests that when we reduce the number of alternatives, there is a change in the derived criteria weights (see Table 5.10 and Table 5.18). The same phenomenon is observed when we compared the derived criteria weights for the reduced set of criteria (for Cases 3 and 4, see Table 5.19 and Table 5.20).

Table 5.18. Criteria weights (Case 2)

Criteria	(+) C_1	(+) C_2	(+) C_3	(+) C_4	(+) C_5	(+) C_6	(+) C_7	(+) C_8	(–) C_9	(–) C_{10}	(–) C_{11}	(–) C_{12}	(–) C_{13}
H_j	0.6296	0.8716	0.7732	0.9319	0.8127	0.8225	0.8202	0.9197	0.8744	0.9120	0.9181	0.9015	0.7753
w_j	0.1818	0.0630	0.1113	0.0334	0.0919	0.0871	0.0882	0.0394	0.0617	0.0432	0.0402	0.0484	0.1103

Table 5.19. Criteria weights (Case 3)

Criteria	(+) C_1	(+) C_2	(+) C_4	(+) C_6	(+) C_7	(–) C_9
H_j	0.8436	0.8556	0.9456	0.8998	0.9178	0.9498
w_j	0.2660	0.2457	0.0925	0.1705	0.1398	0.0854

Table 5.20. Criteria weights (Case 4)

Criteria	(+) C_1	(+) C_2	(+) C_4	(+) C_6	(+) C_7	(–) C_9
H_j	0.6296	0.8716	0.8127	0.8225	0.8202	0.8744
w_j	0.3169	0.1098	0.1602	0.1519	0.1538	0.1075

Table 5.21 to Table 5.23 show the alternatives' comparative ranking under Case 2-4, respectively. After obtaining the ranking of the alternatives by various algorithms, we found the aggregate rank by using the SAW method based on the appraisal scores.

Now, for comparative analysis of various MCDM methods, it is important to see the consistency of their results with the final preferential order. Hence, we

performed a non-parametric rank correlation test. Table 5.17 for Case 1 and Table 5.24 to Table 5.26 for Case 2–4 exhibit the results of correlation tests. From Table 5.24, we find that COPRAS > EDAS > ARAS > MABAC (MARCOS shows non-consistency with the final ranking).

Table 5.21. Comparative analysis of the ranking by different MCDM methods (Case 2)

SMD	Comparative ranking					Final rank (SAW)
	EDAS	ARAS	MABAC	COPRAS	MARCOS	
M_1	3	5	6	2	6	4
M_{10}	9	8	9	9	5	9
M_{15}	8	9	3	8	7	7
M_{20}	7	4	4	6	4	5
M_{25}	5	2	5	5	10	6
M_{30}	6	7	8	7	8	8
M_{35}	1	1	1	1	3	1
M_{40}	4	6	7	4	1	2
M_{45}	2	3	2	3	9	3
M_{50}	10	10	10	10	2	10

Table 5.22. Comparative analysis of the ranking by different MCDM methods (Case 3)

SMD	Ranking Results					Final rank (SAW)
	EDAS	ARAS	MABAC	COPRAS	MARCOS	
M_1	50	48	46	48	42	50
M_2	16	23	21	23	4	13
M_3	48	50	49	50	22	46
M_4	41	34	29	34	50	44
M_5	20	27	32	26	34	29
M_6	10	9	18	11	25	15
M_7	32	33	20	33	3	18
M_8	26	20	9	20	48	31
M_9	40	42	44	42	32	43
M_{10}	38	36	19	36	35	33
M_{11}	11	12	4	14	13	7
M_{12}	36	40	37	40	8	23
M_{13}	4	6	3	6	30	4
M_{14}	5	7	15	7	10	6
M_{15}	13	16	11	16	26	16
M_{16}	43	44	48	43	46	49
M_{17}	34	39	33	37	14	28
M_{18}	1	2	2	2	2	2
M_{19}	24	4	41	3	44	30
M_{20}	44	45	35	45	18	35
M_{21}	46	43	42	44	12	39
M_{22}	12	14	8	15	7	5
M_{23}	37	30	40	30	37	36
M_{24}	22	24	24	24	1	14
M_{25}	42	38	38	39	17	34
M_{26}	30	31	34	31	11	22
M_{27}	17	19	12	18	29	21
M_{28}	19	18	10	19	40	24
M_{29}	49	47	45	47	24	45
M_{30}	21	15	31	12	9	17
M_{31}	28	28	27	28	43	37

SMD	Ranking Results					Final rank (SAW)
	EDAS	ARAS	MABAC	COPRAS	MARCOS	
M_{32}	15	13	26	9	6	11
M_{33}	27	29	36	29	45	40
M_{34}	29	32	16	32	27	26
M_{35}	2	1	14	1	19	1
M_{36}	47	49	50	49	23	47
M_{37}	8	5	6	4	28	9
M_{38}	45	46	43	46	47	48
M_{39}	31	26	25	27	33	32
M_{40}	6	8	7	8	20	8
M_{41}	9	10	13	10	41	19
M_{42}	14	17	17	17	5	10
M_{43}	23	21	22	21	16	20
M_{44}	18	25	30	25	39	27
M_{45}	3	3	1	5	15	3
M_{46}	35	37	28	38	49	42
M_{47}	39	41	47	41	21	41
M_{48}	7	11	5	13	31	12
M_{49}	33	35	39	35	38	38
M_{50}	25	22	23	22	36	25

Table 5.23. Comparative analysis of the ranking by different MCDM methods (Case 4)

SMD	Comparative ranking					Final rank (SAW)
	EDAS	ARAS	MABAC	COPRAS	MARCOS	
M_{1}	9	10	10	10	8	10
M_{10}	6	5	2	5	2	3
M_{15}	5	6	5	6	3	6
M_{20}	7	8	7	8	10	8
M_{25}	8	7	8	7	7	7
M_{30}	4	4	6	3	4	4
M_{35}	1	1	4	1	1	1
M_{40}	3	3	3	4	9	5
M_{45}	2	2	1	2	5	2
M_{50}	10	9	9	9	6	9

Table 5.25 indicates that EDAS > ARAS > MABAC > COPRAS > MARCOS, while from Table 5.26, we trace that ARAS > EDAS > COPRAS > MABAC > MARCOS in terms of consistency of their individual results with final ranking order as obtained by using SAW.

Table 5.24. Correlation test II (Case 2)

Coefficient	Final rank	EDAS rank	ARAS rank	MABAC rank	COPRAS rank	MARCOS rank
Kendall's tau	SAW_Rank	0.778**	0.556*	0.556*	0.778**	0.067
Spearman's rho	SAW_Rank	0.903**	0.758*	0.709*	0.927**	0.139
Aggregated final rank		0.8405	0.657	0.6325	0.8525	0.103

** Correlation is significant at the 0.01 level (2-tailed).
* Correlation is significant at the 0.05 level (2-tailed).

5.4.3 Sensitivity Analysis

Some of the essential requirements for MCDM-based analysis are the rationality,

stability, and reliability of the rankings [688]. There are several variations in the given conditions, for instance, change in the weights of the criteria, MCDM algorithms and normalization methods, and deletion/inclusion of the alternatives that often lead to instability of the results [668] [689] [690]. Sensitivity analysis is conducted to experimentally check the robustness of the results obtained using MCDM based analysis [691] [692]. A particular MCDM method shows stability in the result if it can withstand variations in the given conditions, such as fluctuations in the criteria weights.

Table 5.25. Correlation test III (Case 3)

Coefficient	Final rank	EDAS rank	ARAS rank	MABAC rank	COPRAS rank	MARCOS rank
Kendall's tau	SAW_Rank	.763**	.701**	.659**	.700**	.407**
Spearman's rho	SAW_Rank	.917**	.870**	.840**	.866**	.585**
Aggregated final rank		0.84	0.7855	0.7495	0.783	0.496

** Correlation is significant at the 0.01 level (2-tailed).

Table 5.26. Correlation test IV (Case 4)

Coefficient	Final rank	EDAS rank	ARAS rank	MABAC rank	COPRAS rank	MARCOS rank
Kendall's tau	SAW_Rank	0.733**	0.867**	0.733**	0.911**	0.511*
Spearman's rho	SAW_Rank	0.891**	0.952**	0.867**	0.964**	0.685*
Aggregated final rank		0.812	0.9095	0.8	0.9375	0.598

** Correlation is significant at the 0.01 level (2-tailed).
* Correlation is significant at the 0.05 level (2-tailed).

For the sensitivity analysis, we used the scheme followed in [693], which simulates different experimental scenarios by interchanging criteria weights. Table 5.27 to Table 5.30 present the experimentations vis-à-vis the four cases used in this study. Here, the green-highlighted numbers denote that the cell values of that particular column interchange their weights, in each experiment. In this scheme, we attempt to interchange weights of optimum and sub-optimum criteria, beneficial and cost type of criteria to simulate various possible scenarios for examining the stability of the ranking results obtained by various MCDM methods.

Fig. 5.4 depicts the comparative variations in the rankings of the alternatives as derived by using five MCDM algorithms under different experimental set up for Case 1. We observe that all five considered MCDM methods provide reasonable stability in the solution while COPRAS and ARAS perform comparatively better.

Table 5.31 highlights the correlation of the actual ranking with those obtained by changing the criteria weights (see Table 5.27). In the same way, we carried out the

sensitivity analysis for all MCDM methods for Cases 2 to 4. Table 5.32 to Table 5.34 show the results of the correlation test as we do for Case 1.

Table 5.27. Interchange of criteria weights for sensitivity analysis (Case 1)

Criteria	Criteria weights under different experimental cases				
	Original	Exp1	Exp2	Exp3	Exp4
C_1	0.1441964	0.0169798	0.0501301	0.1441964	0.1441964
C_2	0.1331763	0.1331763	0.1331763	0.1331763	0.1331763
C_3	0.0936409	0.0936409	0.0936409	0.0936409	0.0936409
C_4	0.1049768	0.1049768	0.1049768	0.1049768	0.1049768
C_5	0.0501301	0.0501301	0.1441964	0.0501301	0.0925919
C_6	0.0924398	0.0924398	0.0924398	0.0924398	0.0924398
C_7	0.0757997	0.0757997	0.0757997	0.0757997	0.0757997
C_8	0.0803856	0.0803856	0.0803856	0.0803856	0.0803856
C_9	0.0462696	0.0462696	0.0462696	0.0462696	0.0462696
C_{10}	0.0413577	0.0413577	0.0413577	0.0413577	0.0413577
C_{11}	0.0169798	0.1441964	0.0169798	0.0925919	0.0169798
C_{12}	0.0280555	0.0280555	0.0280555	0.0280555	0.0280555
C_{13}	0.0925919	0.0925919	0.0925919	0.0169798	0.0501301

Table 5.28. Interchange of criteria weights for sensitivity analysis (Case 2)

Criteria	Criteria weights under different experimental cases				
	Original	Exp1	Exp2	Exp3	Exp4
C_1	0.1818299	0.1112996	0.0334131	0.1102984	0.1818299
C_2	0.063014	0.063014	0.063014	0.063014	0.063014
C_3	0.1112996	0.1818299	0.1112996	0.1112996	0.1112996
C_4	0.0334131	0.0334131	0.1818299	0.0334131	0.0334131
C_5	0.0919374	0.0919374	0.0919374	0.0919374	0.0919374
C_6	0.0871434	0.0871434	0.0871434	0.0871434	0.0871434
C_7	0.0882454	0.0882454	0.0882454	0.0882454	0.0882454
C_8	0.0394249	0.0394249	0.0394249	0.0394249	0.0394249
C_9	0.061668	0.061668	0.061668	0.061668	0.061668
C_{10}	0.0431881	0.0431881	0.0431881	0.0431881	0.0431881
C_{11}	0.0401855	0.0401855	0.0401855	0.0401855	0.1102984
C_{12}	0.0483521	0.0483521	0.0483521	0.0483521	0.0483521
C_{13}	0.1102984	0.1102984	0.1102984	0.1818299	0.0401855

Table 5.29. Interchange of criteria weights for sensitivity analysis (Case 3)

Criteria	Criteria weights under different experimental cases				
	Original	Exp1	Exp2	Exp3	Exp4
C_1	0.2660	0.0854	0.0925	0.2660	0.2660
C_2	0.2457	0.2457	0.2457	0.2457	0.1705
C_4	0.0925	0.0925	0.2660	0.0854	0.0925
C_6	0.1705	0.1705	0.1705	0.1705	0.2457
C_7	0.1398	0.1398	0.1398	0.1398	0.1398
C_9	0.0854	0.2660	0.0854	0.0925	0.0854

Fig. 5.4 depicts the comparative variations in the rankings of the alternatives as derived by using five MCDM algorithms under different experimental set up for Case 1. We observe that all five considered MCDM methods provide reasonable stability in the solution while COPRAS and ARAS perform comparatively better. Table 5.31 highlights the correlation of the actual ranking with those obtained by

changing the criteria weights (see Table 5.27). In the same way, we carried out the sensitivity analysis for all MCDM methods for Cases 2 to 4. Table 5.32 to Table 5.34 show the results of the correlation test as we do for Case 1.

Table 5.30. Interchange of criteria weights for sensitivity analysis (Case 4)

Criteria	Criteria weights under different experimental cases				
	Original	Exp1	Exp2	Exp3	Exp4
C_1	0.3168661	0.1074659	0.1098115	0.3168661	0.3168661
C_2	0.1098115	0.1098115	0.3168661	0.1074659	0.1098115
C_4	0.1602149	0.1602149	0.1602149	0.1602149	0.1518606
C_6	0.1518606	0.1518606	0.1518606	0.1518606	0.1602149
C_7	0.153781	0.153781	0.153781	0.153781	0.153781
C_9	0.1074659	0.3168661	0.1074659	0.1098115	0.1074659

5.4.4 Time Complexity Analysis

This section reports the time complexity analysis and the runtimes of the five MCDM methods considered in this study, as summarized in Table 5.35. All the methods have a worst-case time complexity of O(mn), where m is the number of alternatives and n is the number of considered criteria. However, EDAS, MABAC, and COPRAS exhibit Ω(m + n) as the best-case time complexity if the decision matrix is already prepared. But if the matrix is constructed in runtime, the best-case time complexity for these methods also would be Ω(mn).

Depending on the MCC application and architecture, the MCC coordinator where the SMD selection program would run might be a computer or an SMD. That is why, to check the performance of the MCDM methods, we checked the runtime of each of them by running on a laptop and a smartphone.

To run the MCDM algorithms on the laptop, we used Java (version 16) as the programming language and MS Excel (version 2019) as the database. The programs were executed on a laptop with AMD Ryzen 3 dual-core CPU (2.6 GHz, 64 bit) and 4 GB of RAM, operating on Windows 10 (64-bit). To run the programs on a smartphone, we designed an app that could accommodate and run Java program scripts; and in this case, we used a text file to store the decision matrix. The programs were executed on an SoC with 1.95 GHz Snapdragon 439 (12 nm), octa-core (4 × 1.95 GHz Cortex-A53 and 4 × 1.45 GHz Cortex A53) CPU, and Adreno 505 GPU, with 3 GB of RAM, operating on Android 11.

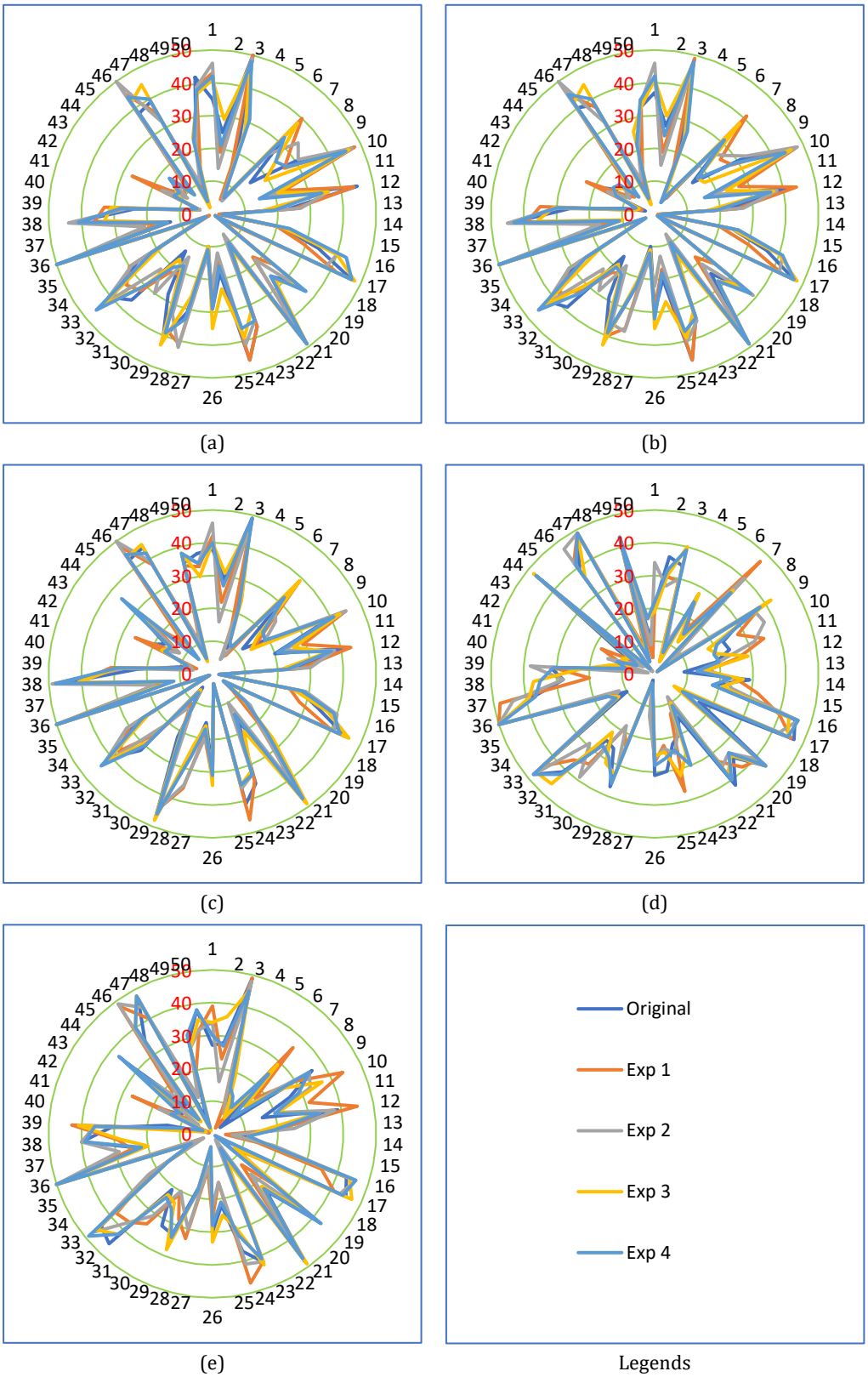


Fig. 5.4. Pictorial representation of sensitivity analysis (Case 1) (a) EDAS, (b) COPRAS, (c) ARAS, (d) MARCOS, (e) MABAC

The MCDM module may get the decision matrix either from the secondary storage or primary memory. We generally might store the database on the secondary storage when we need to maintain the log for future analysis and prediction. But,

updating the SMD resource values in the decision matrix on the secondary storage and retrieving them frequently for decision making involves considerable overhead. Alternatively, the decision matrix could be updated dynamically where the SMD resource values come directly to the coordinator's memory. Compared to secondary storage, accessing memory takes negligible time.

Table 5.31. Correlation test V (sensitivity analysis—Case 1)

Coefficient	Method	Scenario	Exp1	Exp2	Exp3	Exp4
Kendall's tau	EDAS	Original	0.789 **	0.729 **	0.799 **	0.824 **
	ARAS		0.812 **	0.781 **	0.868 **	0.896 **
	MABAC		0.616 **	0.749 **	0.780 **	0.882 **
	COPRAS		0.799 **	0.755 **	0.827 **	0.874 **
	MARCOS		0.734 **	0.752 **	0.796 **	0.881 **
Spearman's rho	EDAS	Original	0.932 **	0.892 **	0.938 **	0.952 **
	ARAS		0.948 **	0.936 **	0.971 **	0.981 **
	MABAC		0.816 **	0.914 **	0.935 **	0.979 **
	COPRAS		0.939 **	0.910 **	0.950 **	0.973 **
	MARCOS		0.905 **	0.914 **	0.945 **	0.974 **

** Correlation is significant at the 0.01 level (2-tailed).

Table 5.32. Correlation test VI (sensitivity analysis—Case 2)

Coefficient	Method	Scenario	Exp1	Exp2	Exp3	Exp4
Kendall's tau	EDAS	Original	0.911 **	0.733 **	0.689 **	0.867 **
	ARAS		0.778 **	0.689 **	0.956 **	0.733 **
	MABAC		0.556 *	0.200	0.556 *	0.600 *
	COPRAS		0.911 **	0.689 **	0.867 **	0.778 **
	MARCOS		0.511 *	0.111	0.556 *	0.867 **
Spearman's rho	EDAS	Original	0.976 **	0.806 **	0.806 **	0.939 **
	ARAS		0.903 **	0.806 **	0.988 **	0.879 **
	MABAC		0.709 *	0.370	0.758 *	0.745 *
	COPRAS		0.964 **	0.830 **	0.939 **	0.915 **
	MARCOS		0.673 *	0.212	0.661 *	0.964 **

** Correlation is significant at the 0.01 level (2-tailed).
* Correlation is significant at the 0.05 level (2-tailed).

Table 5.33. Correlation test VII (sensitivity analysis—Case 3)

Coefficient	Method	Scenario	Exp1	Exp2	Exp3	Exp4
Kendall's tau	EDAS	Original	0.665 **	0.685 **	0.980 **	0.863 **
	ARAS		0.767 **	0.706 **	0.985 **	0.878 **
	MABAC		0.615 **	0.628 **	0.976 **	0.830 **
	COPRAS		0.778 **	0.719 **	0.982 **	0.879 **
	MARCOS		0.946 **	0.956 **	1.000 **	0.979 **
Spearman's rho	EDAS	Original	0.844 **	0.863 **	0.998 **	0.964 **
	ARAS		0.923 **	0.870 **	0.999 **	0.974 **
	MABAC		0.799 **	0.811 **	0.998 **	0.956 **
	COPRAS		0.926 **	0.880 **	0.998 **	0.974 **
	MARCOS		0.992 **	0.994 **	1.000 **	0.998 **

** Correlation is significant at the 0.01 level (2-tailed).

Since in MCC, the SMDs are mobile, the available SMDs (alternatives) continuously change. Existing SMDs may leave, and new SMDs may join the network

randomly. Also, the status of the variable resources (e.g., C_5, C_7, C_8, C_9, C_{10}, C_{11}) of each SMD varies time-to-time depending on its usage. In fact, in a typical centralized MCC, a data logging program always runs in the background to track the values of these recourses. This leads to change the decision matrix continuously. And based on the changed decision matrix, the SMD ranking also changes. It is desirable to store the decision matrix in the memory in such a dynamic scenario as long as resource selection is required.

Table 5.34. Correlation test VIII (sensitivity analysis—Case 4)

Coefficient	Method	Scenario	Exp1	Exp2	Exp3	Exp4
Kendall's tau	EDAS	Original	0.600 *	0.600 *	1.000 **	1.000 **
	ARAS		0.600 *	0.556 *	1.000 **	1.000 **
	MABAC		0.556 *	0.289	1.000 **	1.000 **
	COPRAS		0.556 *	0.511 *	1.000 **	1.000 **
	MARCOS		1.000 **	0.867 **	1.000 **	1.000 **
Spearman's rho	EDAS	Original	0.709 *	0.770 **	1.000 **	1.000 **
	ARAS		0.745 *	0.685 *	1.000 **	1.000 **
	MABAC		0.709 *	0.345	1.000 **	1.000 **
	COPRAS		0.721 *	0.673 *	1.000 **	1.000 **
	MARCOS		1.000 **	0.952 **	1.000 **	1.000 **

** Correlation is significant at the 0.01 level (2-tailed).
* Correlation is significant at the 0.05 level (2-tailed).

Therefore, to have a comparative analysis in this aspect, we calculated the runtime considering both the scenarios: (a) when the dataset was fetched from the secondary storage and (b) when it was preloaded on RAM. The execution time was calculated using a timer (a Java function) in the program. The timer counted the time from data fetching (either from RAM or storage) to completion of the program execution. We executed each algorithm twenty times and took the average runtime. To eliminate the outliers, we discarded the particular execution instances that were abnormally protracted.

From Table 5.35, it can be observed that the average runtimes of the MCDM programs, when they are executed on a laptop, are significantly higher when the decision matrix is in the secondary storage as compared to when it is in memory. However, when these programs are executed on the smartphone, this difference is not that high. This is because the typical storage used in smartphones is much faster than the hard disks of laptops. Another point is worth mentioning that we used text files as a database to execute the programs on the smartphone in our study. If

it were other traditional database applications, the time taken to fetch the dataset from the phone storage would probably be much higher. In that case, the difference between the dataset in memory and storage would be significantly larger.

Table 5.35. Time complexity and runtimes for each MCDM method under various considerations

Method	Time complexity			Case	Average runtime on laptop (milliseconds)		Average runtime on smartphone (milliseconds)	
	Best case	Average case	Worst case		Data in memory	Data in secondary storage	Data in memory	Data in phone storage
Entropy (criteria weight calculation)	Ω(m + n)	θ(mn)	O(mn)	Case 1	0.28391	135.1061	0.69546	1.16032
				Case 2	0.08841	125.0397	0.17581	0.36809
				Case 3	0.12917	124.2696	0.34542	0.73407
				Case 4	0.06234	83.45512	0.09523	0.28998
EDAS	Ω(m + n)	θ(mn)	O(mn)	Case 1	0.36754	124.50158	2.02136	2.46483
				Case 2	0.08993	65.93222	0.42106	0.63313
				Case 3	0.16748	67.90012	0.97938	1.36073
				Case 4	0.06874	54.86296	0.22848	0.39752
ARAS	Ω(mn)	θ(mn)	O(mn)	Case 1	0.30266	139.12975	0.87001	1.32013
				Case 2	0.06918	65.64650	0.22711	0.41631
				Case 3	0.08789	62.64661	0.44734	0.80465
				Case 4	0.04303	49.42035	0.12672	0.30301
MABAC	Ω(m + n)	θ(mn)	O(mn)	Case 1	0.27496	118.52908	1.03990	1.50524
				Case 2	0.0904	64.17373	0.26752	0.45166
				Case 3	0.11870	66.00892	0.53094	0.90594
				Case 4	0.07156	52.62466	0.14914	0.34052
COPRAS	Ω(m + n)	θ(mn)	O(mn)	Case 1	0.12264	122.95953	0.61347	1.05754
				Case 2	0.04076	64.35327	0.13521	0.34481
				Case 3	0.05597	64.29061	0.32844	0.69645
				Case 4	0.03058	50.04589	0.08334	0.25656
MARCOS	Ω(mn)	θ(mn)	O(mn)	Case 1	0.30410	127.74245	0.85634	1.29126
				Case 2	0.06955	64.84879	0.21106	0.40832
				Case 3	0.09898	64.22248	0.44186	0.81885
				Case 4	0.04487	53.29281	0.12259	0.29045

In our comparative analysis, we executed each algorithm ten times for each case. The average runtimes of ten executions were noted. The runtime of any program varies depending on several internal and external factors. That is why we took the average of ten execution instances. However, it is observed that the runtime variations are much higher on a laptop than on a smartphone. This is because the number of background processes typically run on laptops is significantly higher than on smartphones. Also, the resource scheduling in a laptop is more complex than in a smartphone. Nevertheless, the variations in each execution could be more neutralized if the number of considered execution instances is increased.

5.5 Discussion

In this section, we discuss the experimental findings and our observations. We also present a critical discussion on the judiciousness and practicability of this work and the findings.

5.5.1 Findings and Observations

In this section, we discuss the observations on the findings obtained through data analysis. As mentioned throughout the chapter, we had the following four conditions:

- Condition 1: Full set (Case 1: complete set of 13 criteria and 50 alternatives)
- Condition 2: Reduction in the number of alternatives keeping the criteria set unaltered (Case 2: reduced set of 10 alternatives and complete set of 13 criteria)
- Condition 3: Variation in the criteria set (Case 3: reduced set of 6 criteria) keeping the alternative set the same (i.e., 50)
- Condition 4: Variations in both alternative and criteria sets (Case 4: reduced set of 10 alternatives and 6 criteria).

For all conditions, we noticed some variations in the relative ranking orders. By further introspecting the results obtained from different methods and their association with the final ranking (obtained by using SAW), we found that for Case 1, MABAC and COPRAS are more consistent. For Case 2, COPRAS and EDAS outperformed others in terms of consistency with the final ranking. For Case 3, we observed that EDAS and ARAS showed better consistency while COPRAS performed reasonably well. For Case 4, we found that COPRAS and ARAS showed relatively better consistency with the final ranking. Therefore, the first level inference advocates in favour of COPRAS for all conditions under consideration.

Moving further, we checked for stability in the results. We performed a sensitivity analysis for all methods under all conditions, as demonstrated in Section 5.4.3. Here also, we noticed mixed performance. However, COPRAS shows reasonably stable results under all conditions given the variations in the criteria weights except Case 4.

Therefore, it may be concluded that given our problem statement and

experimental setup, COPRAS has performed comparatively well under all case scenarios, while ARAS being its nearest competitor in this aspect. For both methods, the procedural steps are less in number, simple ratio-based or proportional approach is followed, i.e., no need to identify anti-ideal and ideal solutions or calculate distance. Therefore, the result does not show any aberrations. It may, however, be interesting to examining the performance of the algorithms when criteria weights are predefined, i.e., not depending on the decision matrix.

We further investigated the time complexities of the MCDM algorithms used in this work to find out the most time-efficient one. All the considered MCDM methods perform equally in this aspect, though the best-case time complexity for EDAS, MABAC, and COPRAS is better than others. Fig. 5.5 to Fig. 5.8 graphically present the case-wise comparisons of the runtimes of each MCDM method for all the scenarios. Our experiment observed that the COPRAS method exhibits the most petite runtime for each dataset (cases) for all the considered scenarios, i.e., whether the dataset is in the secondary storage or memory or the program is run on a laptop or smartphone. Specifically, considering the average runtime for all the cases and scenarios, the ranking of the MCDM methods as per their runtime (RT) is: RT_COPRAS < RT_MARCOS < RT_ARAS < RT_MABAC < RT_EDAS.

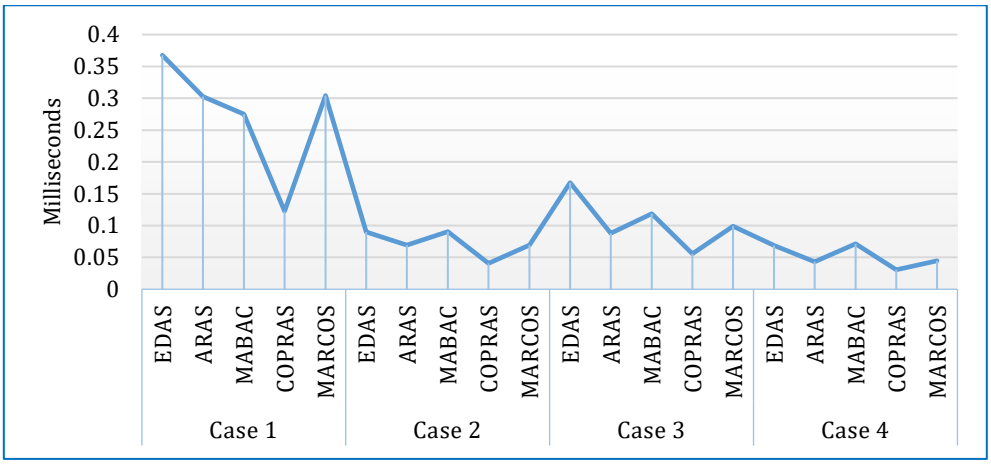


Fig. 5.5. Runtime comparison of MCDM methods on the laptop for each case when the dataset is in the memory

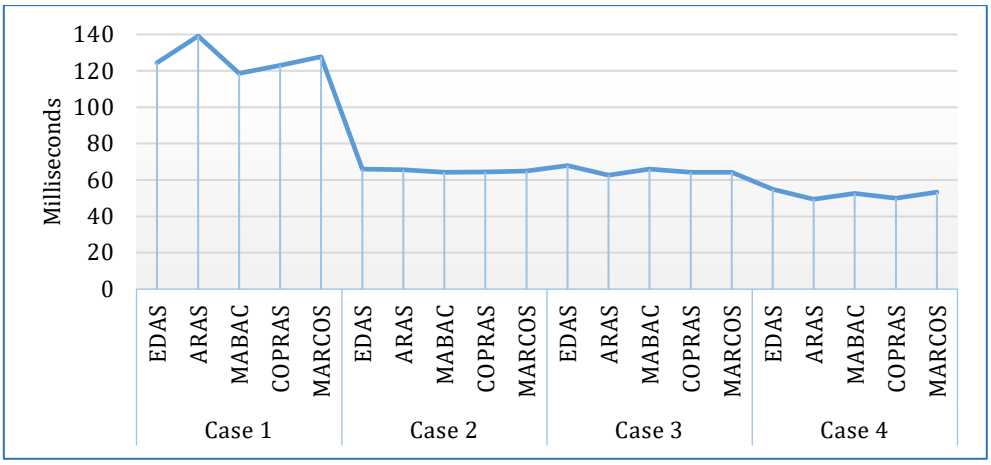


Fig. 5.6. Runtime comparison of MCDM methods on the laptop for each case when the dataset is in the secondary storage

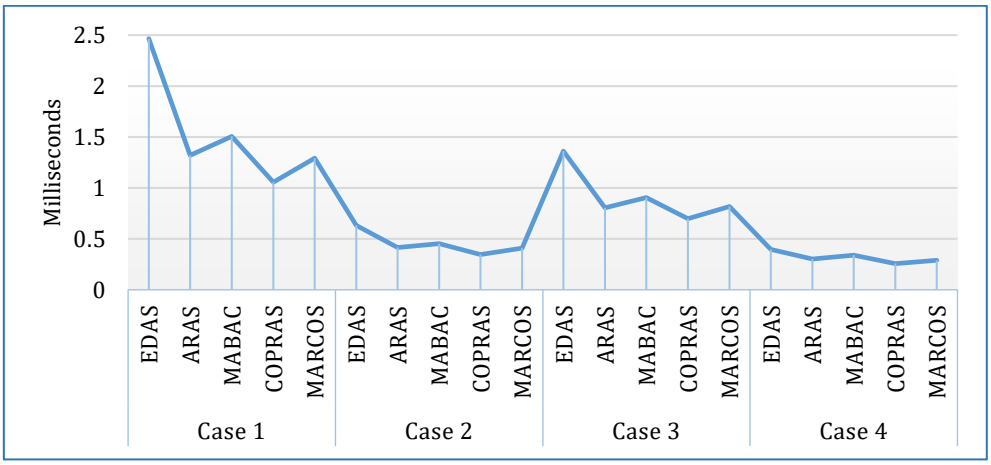


Fig. 5.7. Runtime comparison of MCDM methods on the smartphone for each case when the dataset is in the phone storage

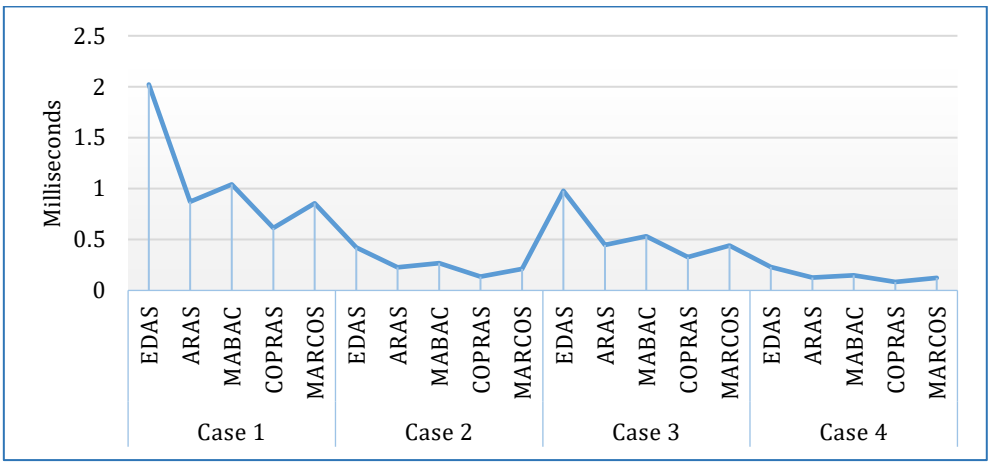


Fig. 5.8. Runtime comparison of MCDM methods on the smartphone for each case when the dataset is in the memory

However, this rank does not hold true for all the executions in each case. For example, from Fig. 5.6, it can be noted that ARAS and MABAC took less time to execute in Case 1. In practice, Case 3 probably would be more common than other cases for a typical MCC application, i.e., there would be few numbers of SMDs

available as computing resources and the application demanding a certain number of selection criteria. For this case, COPRAS took 0.05597 milliseconds on average if it runs on a laptop while the dataset resides in the memory and 0.32844 milliseconds for a smartphone. For a dynamic resource selection in MCC, this time requirement is tolerable. However, when the dataset is on the secondary storage, the runtime increases exponentially in the case of the laptop but not a smartphone.

The runtime for both the MCDM method and Entropy calculation should be considered to get the effective runtime for the ranking process. Like the MCDM methods, for Entropy calculation also, when the dataset is on the secondary storage, the runtime increases exponentially in the case of the laptop but not a smartphone, as shown in Fig. 5.9. Therefore, we can postulate that if the MCC coordinator is a laptop or desktop computer, the dataset needs to be stored in the memory before resource selection.

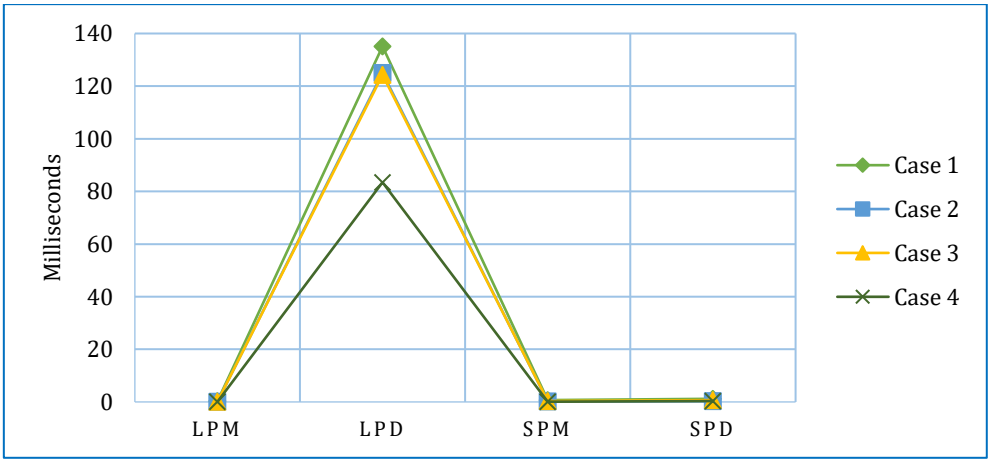


Fig. 5.9. Runtime comparison for Entropy method

Considering the above discussions, it can be deduced that the COPRAS method is the most suitable for resource selection in MCC in terms of correctness, robustness, and computational (time) complexity.

5.5.2 Rationality and Practicability

In this section, we present a critical discussion on the rationality and practicability of this study.

5.5.2.1 Assertion

In the previous section, we conclusively observed that for resource selection in MCC, the COPRAS method is the most favourable in all respect. However, it should

not be misinterpreted that the COPRAS method is the ideal solution for resource selection in MCC. In fact, optimized resource selection in a dynamic environment like MCC is an NP-hard problem. Hence, practically no solution can be claimed as optimal. We only assert that we found that COPRAS scales favourably in all aspects compared to other methods. But this does not mean that COPRAS is the ideal solution. There is always scope to explore further for a more suitable multi-criteria resource selection algorithm that would be more computing and time-efficient.

It is to be noted that the effectiveness of an MCDM solution depends on the particular problem and the data. In real implementations of MCC, the actual SMD data would certainly change, be it for different instances of the same MCC system or in different MCC systems. Due to the dynamic nature of a typical MCC, the SMDs are not fixed. Even if the SMDs are fixed in an MCC for a certain period, their resource values will vary depending on the applications running on them and their users' device usage behaviour. Moreover, since the need for computing resources varies according to application requirements, the selection criteria and weights also differ accordingly. In these cases, the datasets would vary from those we used in our experiment. But the problem behaviour and data types would be the same for all MCC applications and throughout their different execution instances. Hence, a solution found suitable for the given dataset would be applicable to any similar dataset for MCC. Even if the size of the datasets varies in different MCC, the finding of this study will hold true because we found that COPRAS performed comparatively better in all four considered datasets of different sizes.

5.5.2.2 Application

The resource selection module is generally incorporated in the resource manager module of a typical distributed system. And the resource manager module generally is part of the middleware of a 3-tier system. Therefore, in the actual designing and implementation of an MCC system, the MCDM-based resource selection algorithm would be integrated into the middleware of MCC. This resource selection algorithm should generate a ranked list of the available SMDs based on their resources. The MCC job scheduler would dispatch the MCC jobs to the top-ranked SMDs from the list. This would ensure a better turnaround time and throughput

and, in turn, better QoS of the MCC.

5.5.2.3 Implications

The findings of this work would allow the MCC system designers and developers to adopt the right resource selection method for their MCC based on its scale and also on the preference and priority of the resource types. This would also contribute to managerial decision-making for implementing organizational MCC. As the study simulates different scenarios and compares the available options, it would be a likely reference for the decision-makers to choose the right MCDM method for resource selection and consider the appropriate size of the employed MCC and decide on the right number of selection criteria.

Furthermore, the pronouncements of this work shall allow the researchers to choose a suitable MCDM method with reasonably higher accuracy and lesser run time complexity to solve real-life problems similar to the one discussed in this chapter. Not only the researchers in the area of MCC and other allied fields (e.g., mobile grid computing, mobile cloud computing, and other related forms of distributed computing), this study would be of interest also to the people from the MCDM field who might find it motivating to nurture this problem domain and come up with some novel or improved methods that would be more suitable to address the associated resource dynamicity.

5.6 Limitations and Further Scopes

The MCC environment is really dynamic in nature, i.e., not only the SMDs but also the status of the resource parameters of each existing SMDs change frequently. Therefore, the resource selection not only needs to be optimal but also to be adaptive in an unpredictable MCC environment. That is to say, the MCDM method should be capable of acclimating to the continual alteration in the data matrix due to the frequent variation of the available SMDs and their resource values. Ideally, whenever there is a change in the alternative list or in the performance score, the MCDM method should be able to reflect this change in the overall ranking without reranking the whole list in the next iteration of resource selection. This should not only minimize the SMD selection and decision-making time but also truly reflect

the dynamic and scalable nature of MCC, which is not in the case of the traditional MCDM methods.

We used the entropy method to calculate the criteria weights. It is an objective approach in which the criteria weights depend on the decision matrix values. In a dynamic environment like MCC, the SMDs may join and leave the network frequently, and the status of their variable resources also changes as per device usage. This results in frequent alteration in the decision matrix. This implies that the entropy calculation should be done every time for criteria weight determination, which is a real overhead.

Here, the criteria weights were calculated dynamically based on the present resource status of the SMDs, expressed in metric terms. We did not take into account the criteria preferences in line with the resource specification preference of the MCC applications. As the dataset gets changed based on varying criteria and alternative sets, the criteria weights also get changed according to the performance values of the alternatives. Hence, this approach might not provide the optimal resource ranking as per the real applicational requirements. So, our future study can explore the possibility of defining the criteria weights based on the required resource specifications of a typical MCC user or application.

We opted for the most straightforward normalization technique, i.e., linear normalization. But there are various normalization techniques in practice that could be used. Therefore, there is a scope to study the effect of different normalization techniques in the ranking and execution performance of the MCDM methods.

5.7 Summary

For better QoS of MCC, selecting the most capable SMDs is essential. Since the selection is made based on several diverse SMD resources, the SMD selection problem can be described as multicriteria decision-making (MCDM) problem.

In this chapter, we performed a comparative assessment of different MCDM methods (EDAS, ARAS, MABAC, MARCOS, and COPRAS) to rank the SMDs based on their resource parameters, among a number of available SMDs, for being considered as computing resources in MCC. The assessment was done in terms of ranking

robustness and the execution time of the MCDM methods. Considering the dynamic nature of MCC, where the resource selection is supposed to be on-the-fly, the selection process needs to be as less time-consuming as possible. For selection criteria, we considered the fixed (e.g., CPU and GPU power, RAM and battery capacity, etc.) and the variable (e.g., current CPU and GPU load, available RAM, battery remaining, etc.) resource parameters.

We used the final score values of the alternatives as obtained by using different algorithms and applied the SAW method for arriving at the aggregate ranking of the alternatives. We also carried out a comparison of the ranking performance of the MCDM methods used in this study. We investigated their consistency with respect to the aggregate ranking and their stability through sensitivity analysis.

We calculated the time complexities of all the methods. We also assessed the actual runtime of all the methods by executing them on a Windows-based laptop and an Android-based smartphone. To assess the effect of the size of the dataset, we executed the MCDM methods with four datasets of different sizes. To have datasets of varied sizes, we changed the number of selection criteria and alternatives (SMDs) separately. For each dataset, we executed the programs considering two scenarios, when the dataset resides in the primary memory and when it is fetched from secondary memory.

It is observed that in terms of correctness, consistency, and robustness, the COPRAS method exhibits better performance under all case scenarios. As per time complexity, all the five MCDM methods are equal, i.e., O(mn), where $m \times n$ is the decision matrix (m is the number of SMDs and n is the number of selection criteria). However, EDAS, MABAC, and COPRAS have a better best-case ($\Omega(m + n)$) complexity. Overall, COPRAS has been shown to have the least runtime for each execution case, i.e., for all four matrix sizes, on the laptop and on the smartphone.

The COPRAS method is found to be better than other MCDM methods (EDAS, ARAS, MABAC, and MARCOS) for all test parameters and in all test scenarios. Hence, it can be concluded that among the existing MCDM methods, COPRAS would be the most suitable choice for resource ranking to select best resources in MCC and other similar systems.

6

Task Scheduling in MCC

"We can't change the wind, but we can set the sails differently." --- Aristotle

6.1 Introduction

To attain the best performance of an MCC system, it is crucial to schedule the tasks to the SMDs optimally. The hardware organisations of the SMDs are vastly heterogeneous. The overall computing capacity depends on various SMD resources such as CPU and GPU power, available RAM, etc. For example, SMDs might have different CPU and GPU models with different clock frequencies. Accordingly, the execution time for a certain task on different SMDs would not be the same. It is understood that an SMD would be more efficient in executing the assigned task if it is equipped with more computing power. Besides, factors such as the remaining battery, device temperature, etc., also play crucial roles for an SMD being considered a suitable computing entity. An efficient scheduling policy should consider all these parameters to maximise the overall performance of MCC. Additionally, considering the limited battery power of the SMDs an ideal schedular should be energy-efficient, i.e., to achieve the successful and sustainable attainment of MCC as HPC, it is vital to manage the SMD load efficiently so that the MCC tasks are executed with minimum energy consumption.

However, there is an issue in fulfilling the above goals unilaterally. It might happen that to maximise the system's overall throughput and/or energy efficiency, the same SMDs are assigned tasks repetitively. While, the low-profile and/or and high energy consuming SMDs might receive the tasks dispersedly. This leads to improper utilisation of resources and a huge load imbalance among the SMDs. This is not appreciable for any distributed systems. It becomes more vital in MCC since it is a crowdsourced system. Obliging the typical human nature, people would be apprehensive about being part of MCC if their SMDs are continuously overloaded.

Hence, it is necessary to have an efficient scheduling algorithm that not only

maximises the MCC performance with minimum energy consumption but also negotiates the issue of underutilisation of resources with even load balancing among all the available SMDs. The task scheduling problem in distributed systems is typically an NP-complete, that is why its solution approaches are generally heuristic or metaheuristic, by which we attempt to reach an approximately optimal solution instead of an absolute optimal one. To attain the above objectives, we also use heuristic and metaheuristic approaches to propose two scheduling solutions in this chapter, divided into the following two sections:

a) In Section 6.2, we present an efficient resource-aware task scheduling algorithm for MCC, conforming to multiple optimisation criteria such as maximised makespan, and resource utilisation and minimised dispersity in load balance. For this we, followed a heuristic approach.
b) In Section 6.3, we present a load balance aware energy-efficient task scheduling algorithm for MCC, aiming to schedule the MCC tasks to the designated SMDs so that the overall energy consumption of the SMDs remains minimum as well the load distribution among the SMDs remain fairly even. For this solution, we use a PSO-based metaheuristic approach.

Overall, in this chapter, we aim to achieve the followings:

- Design a heuristic-based resource-aware task scheduling algorithm for MCC by considering multiple real-time dynamic resource parameters of SMDs.
- The algorithm should attain the objectives like minimised makespan and load dispersity, and maximised resource utilisation that are important aspect of the performance of MCC.
- Design a metaheuristic-based energy-efficient task scheduling algorithm for MCC by considering the real-time CPU information of SMDs.
- This algorithm should attain the objectives like minimum energy consumption and also fair load balancing, which is a crucial aspect for energy-constraint resources.
- Conduct extensive simulation for both the algorithms on synthetic and real datasets to analyse and validate their efficacy.
- Frame multiple simulation scenarios with different sets of task-SMD mappings

to check the algorithms' reliability and consistency.

- Compare the performance of the proposed algorithms with similar heuristic and metaheuristic algorithms.

6.2 Resource-Aware Scheduling

In this section, we present a heuristic scheduling algorithm that considers different resource parameters of the SMDs to schedule the MCC tasks considering makespan, resource utilization and load balance.

6.2.1 System Model and Problem Formulation

The system and execution models of the proposed scheduler for MCC are discussed below. Here we also formally establish the addressed problem.

6.2.1.1 System Model

Here, we have considered a local MCC [465]. The coordinator divides a large computationally intensive job into a few smaller noninterfering and parallelly executable tasks, as shown in Fig. 6.1. Here, we considered the MCC model for the applications with loosely-coupled parallel tasks, such as the bag-of-tasks (BoT) applications whose tasks are completely independent. These tasks are assumed of homogeneous characteristics and have a uniform format but they might have nonuniform computation length (execution time) and input and output data size. The execution time also varies due to the diverse computing capacities of SMD processors. The middleware selects and schedules the tasks to the SMDs based on different optimising criteria.

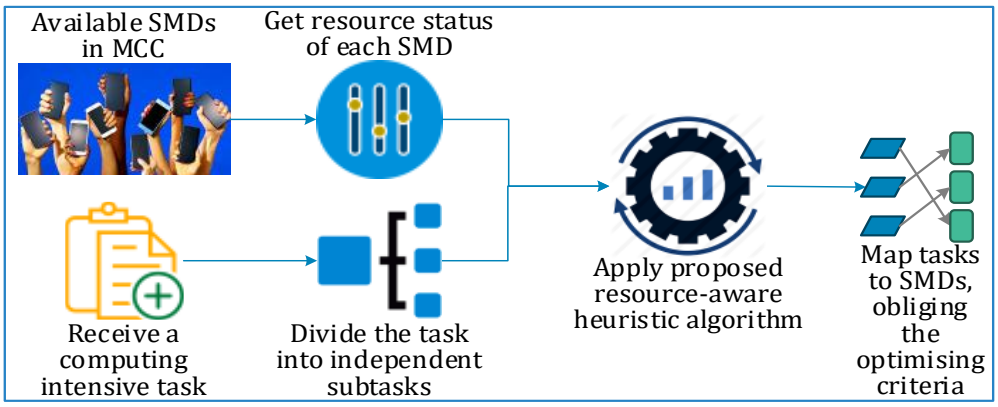


Fig. 6.1. Task scheduling in MCC

We considered the following resource parameters of the SMDs for task scheduling:

a. **CPU:** CPU power denotes the computation capability of the CPU of an SMD. It depends on two factors: the clock frequency of the CPU and the number of cores within it. The effective CPU power of an SMD (m_k) is calculated using Eq. 6.1, where $CPU\_frequency_k$ denotes the highest clock frequency of the CPU cores of m_k, CPU_core_k indicates the number of CPU cores in m_k and CPU_load_k represents the current CPU load of m_k.

$$\boldsymbol{P(m_k) = \frac{CPU\_frequency_k \times CPU\_core_k}{CPU\_load_k}} \tag{6.1}$$

b. **Available RAM:** Since the MCC tasks need to be loaded into the RAM for execution, there should be enough free space in the RAM for efficient execution. We denote available RAM of an SMD as $R(m_k)$.

c. **Battery:** It indicates how much battery charge is left. Higher is better. The battery is generally represented by %, which is typically relative to the total capacity of the SMDs' battery. For example, for a 60% availability, the SMD with 12000 mA of the battery will have a much higher charge remaining than the one having a battery of 6000 mA. The effective available battery of SMD (m_k) is calculated using Eq. 6.2, where $total\_battery\_capacity_k$ indicates the total capacity of m_k's battery and $present_{battery_{avl_k}}(\%)$ suggests the current charge % of m_k's battery.

$$\boldsymbol{B(m_k) = \frac{total\_battery\_capacity_k \times present_{battery\_avl_k}(\%)}{100}} \tag{6.2}$$

d. **Device temperature:** SMD's temperature depends on the heat generated by individual components such as the processing unit, signal module, battery, etc. However, for simplicity, we considered only the overall device temperature (°C). We denote the temperature of an SMD as $T(m_k)$.

6.2.1.2 Execution model

Let us assume, at a time instant (τ), the coordinator divides a job T into a set of independent tasks such that $T = \{t_1, t_2, t_3, \dots, t_n\}$. The tasks vary in terms of instruction lengths in million instructions per second (MIPS), resource requirements, processing time, etc. At τ, a set of SMDs M ($M = \{m_1, m_2, m_3, \dots, m_m\}$) is available in the MCC system. These n tasks need to be executed on m *SMDs*. The processors of M are heterogeneous, i.e., they have different computing capacities,

and hence, they take dissimilar durations to execute a certain task. In the following, we define the terms used in this work.

Execution time: The estimated time to execute t_j on m_k is calculated by Eq. 6.3.

$$X_T(t_j, m_k) = \frac{z_{t_j}}{P(m_k)} \tag{6.3}$$

where, $\boldsymbol{z_{t_j}}$ denotes the instruction length of t_j in MIPS and $\boldsymbol{P(m_k)}$ is the effective CPU of m_k.

Start time: The timestamp at which m_k starts executing t_j is formulated by Eq. 6.4.

$$S_T(t_j, m_k) = \min\{R_T(m_k)\} \tag{6.4}$$

Release time: $R_T(m_k)$ denotes the timestamp when m_k completes executing all the assigned tasks to it. Initially, $R_T(m_k) = 0 | \forall m_k \in M$. After execution of t_j on m_k, $R_T(m_k)$ is updated using Eq. 6.5.

$$R_T(m_k) = R_T(m_k) + EF_T(t_j) \tag{6.5}$$

Earliest finish time: The earliest finish time denotes the shortest time to complete the execution of t_j on m_k, and is calculated using Eq. 6.6.

$$EF_T(t_j) = \min\{F_T(t_j, m_k) | \forall m_k \in M\} \tag{6.6}$$

Finish time: The finish time denotes the timestamp when t_j starts its execution on m_k plus its own execution time on m_k. It is calculated using Eq. 6.7.

$$F_T(t_j, m_k) = S_T(t_j, m_k) + X_T(t_j, m_k) \tag{6.7}$$

Makespan: It denotes the total scheduled length by timestamping when all the tasks are completed by all the allocated SMDs. The makespan is calculated using Eq. 6.8.

$$MS = \max\{R_T(m_k)\} \tag{6.8}$$

Load balancing: The scheduling algorithm should assign the tasks to the SMDs so that they are evenly loaded. The load balancing is defined by Eq. 6.9. The value of *LB* lies between (0,1], and a lower value of *LB* indicates better balancing of task loads where R_T^{avg} is the average release time and is defined by Eq. 6.10.

$$LB = \sqrt{\frac{\sum_{k=1}^{r}\{R_T^{avg} - R_T(m_k)\}^2}{M}} \tag{6.9}$$

$$R_T^{avg} = \frac{1}{M}\sum_{k=1}^{r} R_T(m_k) \tag{6.10}$$

Resource utilisation: The ratio between the average release time and the scheduled length is calculated using Eq. 6.11.

$$RU = \frac{R_T^{avg}}{MS} \tag{6.11}$$

6.2.1.3 Problem Formulation

We present the proposed multicriteria-based resource-aware task scheduling problem as a linear programming problem (LPP). The goal is to schedule a set of n number of independent tasks, i.e., $T = \{t_1, t_2, t_3, \ldots, t_n\}$ to a set of m number of SMDs, i.e., $M = \{m_1, m_2, m_3, \ldots, m_m\}$ based on combined resources. The objective is to minimise makespan (MS) and load balance (LB) and maximise resource utilisation (RU) simultaneously.

Here, we adopted a weighted sum approach. A weight value (ω_i), defined by Eq. 6.12, is multiplied with each objective parameter and subsequently are summed up to form a combined final scheduling objective (S_{obj}).

$$\sum_{i=1}^{3} \omega_i = 1, 0 < \omega_i \leq 1 | \forall i, 1 \leq i \leq 3 \tag{6.12}$$

The goal is to minimise S_{obj}, satisfying the set objectives, and the problem is formally expressed by Eq. 6.13, subject to Eq. 6.14 and Eq. 6.12.

$$S_{obj} = \omega_1 \times MS + \omega_2 \times LB + \omega_3 \times (1 - RU) \tag{6.13}$$

$$\sum_{k=1}^{m} \alpha_k^j = 1 | \forall j, 1 \leq j \leq n \tag{6.14}$$

where, α_k^j is a be a Boolean variable and is defined by Eq. 6.15.

$$\alpha_k^j = \begin{cases} 1, \text{if } t_j \text{ is assigned to } m_k \\ 0, \qquad\qquad \text{otherwise} \end{cases} \tag{6.15}$$

Considering Eq. 6.15, Eq. 6.14 represents a non-preemptive scheduling property.

6.2.2 Proposed Heuristic-based Resource-aware Scheduling for MCC

In this section, to discuss the proposed scheduling method, we first calculate the

resource strength of each SMD and then accordingly prioritise the tasks for scheduling.

6.2.2.1 Resource Strength Assessment

Each SMD or m_k consists of p number of resource parameters as $R = \{r_1, r_2, ..., r_p\}$. The resource strength, $RS(m_k)$, of an SMD suggests the overall status of its resources. It signifies an SMD's competence as a computing node at any instant. Considering the real-time resources, as described in Section 6.2.1.1, $RS(m_k)$ is calculated by equation Eq. 6.16.

$$RS(m_k) = \sum_{i=1}^{r} \theta_i \times \frac{r_{ik}}{\max(r_{ik})} \tag{6.16}$$

where θ_i is the weight value of i^{th} resource, subject to $0 < \theta_i \leq 1$ and $\sum_{i=1}^{r} \theta_i = 1$, r_{ik} is the current measure of the i^{th} resource of SMD m_k, r is the number of resource parameters considered (in our case, it is 4), and max (r_{ik}) indicates the maximum current measure of the i^{th} resource among all the SMD available in MCC at the current time.

6.2.2.2 Scheduling Cost Estimation

For mapping the tasks with respect to suitable SMDs, we calculate the scheduling score (SC) for each combination of (t_j, m_k) using Eq. 6.17. Subsequently, the average SC of a particular task is calculated for all the SMDs using Eq. 6.18.

$$SC(t_j, m_k) = \{\frac{z_{t_j}}{RS(m_k)} \times \frac{1}{P(m_k)} | \forall m_k \in M, \forall t_j \in T\} \tag{6.17}$$

$$SC_{avg}(t_j, M) = \sum_{k=1}^{m} \frac{SC(t_j, m_k)}{m} \tag{6.18}$$

The overall stepwise mapping procedure of t_j to m_k based on calculated $SC_{avg}(t_j, M)$ is demonstrated in Fig. 6.2. The pseudocode of the proposed scheduling algorithm is presented in Algorithm 6.1.

6.2.2.3 Illustration

In the following, we illustrate the working of the proposed methodology. Subsequently, we compare it with three other methods.

Let us consider, at a time instance (τ), an MCC task arrives at the MCC coordinator

and is divided into ten independent subtasks $(t_1, t_2, t_3, \ldots, t_{10})$. Each task has a different instruction length. Let us assume at τ four SMDs (m_1, m_2, m_3, m_4) are available, and the tasks are to be scheduled to these SMDs. A task would take different times for execution on different SMDs.

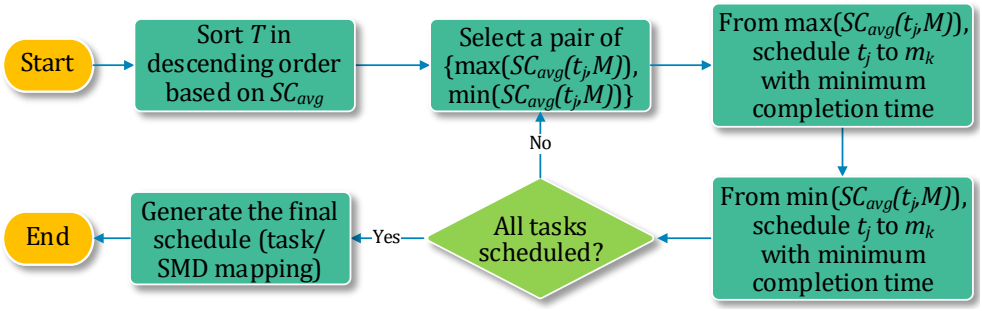


Fig. 6.2. Procedure of mapping tasks to SMDs

Algorithm 6.1: Multicriteria-based Resource-Aware Scheduling for MCC

Input: (1) Set of tasks, $T = \{t_1, t_2, t_3, \ldots, t_n\}$ with z_{t_j} instruction length of t_j
(2) Set of SMDs, $M = \{m_1, m_2, m_3, \ldots, m_m\}$
(3) Set of resources, $R = \{r_1, r_2, \ldots, r_p\}$ for each m_k

Output: $T \rightarrow M$ mapping

```
for (∀m_k ∈ M)
{
        Calculate P(m_k) using Eq. 6.1
        Calculate B(m_k) using Eq. 6.2
}
for (∀m_k ∈ M)
{
        Calculate RS(m_k) using Eq. 6.16
}
for (∀t_j ∈ T)
{
        Calculate SC(t_j, m_k) using Eq. 6.17
        Calculate SC_avg(t_j, M) using Eq. 6.18
}
Sort T in descending order based on SC_avg(t_j, M)
Initialise
        max = 1
        min = j
for (i = 1 to j)  // run for all the tasks in T
{
        if (i%2 == 1) then
        {
                Select the max task from T and assign it to the SMD with min(E_FT(t_j)) as per Eq. 6.6
                max ++
        }
        else
        {
                Select the min task from T and assign it to the SMD with min(E_FT(t_j)) as per Eq. 6.6
                min --
        }
}
Calculate S_obj using Eq. 6.13
```

As mentioned in Section 6.2.2.1, each SMD consists of several resources. The

normalised values of the resources of each SMDs are shown in Table 6.1. The resource strength (*RS*) of each SMD is calculated by Eq. 6.16, as shown in the table. With respect to each task, the scheduling score (*SC*) is calculated for each SMD, as shown in Table 6.2.

Now, as per Algorithm 6.1, we first select the task with maximum SC_{avg} and schedule it to the SMD with minimum execution time. From Table 6.2, we can see that t_2 has a maximum SC_{avg} value and is mapped to m_2. Next, the task with minimum SC_{avg} should ideally be mapped to the SMD with minimum execution time. Here, t_6 has the minimum SC_{avg}, which should be mapped to m_2 because m_2 provides the minimum execution time for t_6. However, m_2 is still busy executing t_2. Hence, t_6 is mapped to m_3 with the next minimum completion (as per Eq. 6.6) time after m_2.

The complete scheduling sequence for all the tasks is shown in Table 6.3. It can be observed (highlighted in bold) that the final schedule length is 331.754, accounting for $t_9 \rightarrow m_3$, i.e., task t_9 is scheduled to SMD m_3.

To check the stand of our proposed algorithm, we compared it with PSO, GA, and MCT. The makespan and execution sequences for PSO, GA, and MTC are shown in Table 6.4, Table 6.5, and Table 6.6, respectively. The release times for each SMD after executing all the scheduled tasks are given in Table 6.7. The maximum release time is the makespan for the respective method, as per Eq. 6.5 and Eq. 6.8. The overall performance comparison of all the evaluated methods with respect to the considered objectives is given in Table 6.8. It can be seen that the proposed algorithm outperforms all other three methods in all respects.

Remark 6.1. It can be observed in Table 6.3 that t_1 is scheduled to m_4; however, from Table 6.2, it can be seen that the minimum execution for t_1 is achieved if it was mapped to m_2. The same can be stated for other tasks also. However, our algorithm does not blindly schedule a task to the SMD that has the minimum execution time; rather, it calculates the earliest completion time (calculated using Eq. 6.6) of the given task on any SMDs.

Remark 6.2. The proposed algorithm schedules the tasks not only based on the earliest finish time but also considers the other two optimising objectives, i.e., load

balance and resource utilisation. Thus, our algorithm provides balanced scheduling to achieve the best possible combinations of all three objectives.

Table 6.1. Resource strength calculation

SMD	$P(m_k)$	$R(m_k)$	$B(m_k)$	$T(m_k)$	$RS(m_k)$
m_1	0.195	0.074	0.022	0.091	0.653019
m_2	0.226	0.078	0.025	0.093	0.724554
m_3	0.115	0.144	0.096	0.097	0.967786
m_4	0.211	0.112	0.206	0.096	0.698103

Table 6.2. Computation of execution time and scheduling score

Task	Task size	$X_T(t_j, m_k)$				$SC(t_j, m_k)$				
		m_1	m_2	m_3	m_4	m_1	m_2	m_3	m_4	SC_{avg}
t_1	20	102.564	88.889	94.787	173.913	157.061	122.681	97.942	249.122	156.702
t_2	44	225.641	195.556	208.531	382.609	345.535	269.898	215.472	548.069	344.744
t_3	31	158.974	137.778	146.919	269.565	243.445	190.155	151.810	386.140	242.888
t_4	11	56.410	48.889	52.133	95.652	86.384	67.474	53.868	137.017	86.186
t_5	20	102.564	88.889	94.787	173.913	157.061	122.681	97.942	249.122	156.702
t_6	8	41.026	35.556	37.915	69.565	62.825	49.072	39.177	99.649	62.681
t_7	38	194.872	168.889	180.095	330.435	298.417	233.094	186.089	473.332	297.733
t_8	23	117.949	102.222	109.005	200.000	180.621	141.083	112.633	286.491	180.207
t_9	20	102.564	88.889	94.787	173.913	157.061	122.681	97.942	249.122	156.702
t_{10}	14	71.795	62.222	66.351	121.739	109.943	85.877	68.559	174.386	109.691

Table 6.3. Makespan using the proposed algorithm

Task	m_1	m_2	m_3	m_4	Schedule sequence
t_1	-	-	-	121.739-295.652	1. $t_2 \rightarrow m_2$
t_2	-	0-195.556	-	-	2. $t_6 \rightarrow m_3$
t_3	-	-	90.048-236.967	-	3. $t_7 \rightarrow m_1$
t_4	-	-	37.915-90.048	-	4. $t_4 \rightarrow m_3$
t_5	194.872-297.436	-	-	-	5. $t_3 \rightarrow m_3$
t_6	-	-	0-37.915	-	6. $t_{10} \rightarrow m_4$
t_7	0-194.872	-	-	-	7. $t_8 \rightarrow m_2$
t_8	-	195.556-297.778		-	8. $t_1 \rightarrow m_4$
t_9	-	-	**236.967-331.754**	-	9. $t_5 \rightarrow m_1$
t_{10}	-	-	-	0-121.739	10. $t_9 \rightarrow m_3$

Table 6.4. Makespan using PSO

Task	m_1	m_2	m_3	m_4	Schedule sequence
t_1	-	0-88.889	-	-	1. $t_3 \rightarrow m_4$
t_2	-	88.889-284.445	-	-	2. $t_1 \rightarrow m_2$
t_3	-	-	-	0-269.565	3. $t_2 \rightarrow m_2$
t_4	-	284.445-333.334	-	-	4. $t_5 \rightarrow m_3$
t_5	-	-	0-94.787	-	5. $t_4 \rightarrow m_2$
t_6	0-41.026	-	-	-	6. $t_6 \rightarrow m_1$
t_7	-	-	94.787-274.882	-	7. $t_7 \rightarrow m_3$
t_8	-	-	274.882-383.887	-	8. $t_{10} \rightarrow m_2$
t_9	41.026-143.59	-	-	-	9. $t_8 \rightarrow m_3$
t_{10}	-	**333.334-395.556**	-	-	10. $t_9 \rightarrow m_1$

Table 6.5. Makespan using GA

Task	m_1	m_2	m_3	m_4	Schedule sequence
t_1	-	-	0-94.787	-	1. $t_2 \rightarrow m_2$
t_2	-	0-195.556	-	-	2. $t_1 \rightarrow m_3$
t_3	-	195.556-333.333	-	-	3. $t_4 \rightarrow m_1$
t_4	0-56.410	-	-	-	4. $t_3 \rightarrow m_2$
t_5	-	-	-	0-173.913	5. $t_5 \rightarrow m_4$
t_6	-	-	94.787-132.701	-	6. $t_6 \rightarrow m_3$
t_7	56.41-251.282	-	-	-	7. $t_9 \rightarrow m_4$
t_8	-	-	132.701-241.706	-	8. $t_7 \rightarrow m_1$
t_9	-	-	-	**173.913-347.826**	9. $t_8 \rightarrow m_3$
t_{10}	251.282-323.077	-	-	-	10. $t_9 \rightarrow m_1$

Table 6.6. Makespan using MCT

Task	m_1	m_2	m_3	m_4	Schedule sequence
t_1	-		0-94.787		1. $t_7 \rightarrow m_2$
t_2	-		94.787-303.318	-	2. $t_1 \rightarrow m_3$
t_3	0-158.974	-		-	3. $t_3 \rightarrow m_1$
t_4	-	**303.318-352.207**		-	4. $t_9 \rightarrow m_4$
t_5		168.889-257.778	-	-	5. $t_2 \rightarrow m_3$
t_6	-	-		173.913-243.478	6. $t_5 \rightarrow m_2$
t_7		0-168.889	-	-	7. $t_{10} \rightarrow m_1$
t_8	230.769-348.718			-	8. $t_6 \rightarrow m_4$
t_9	-	-		0-173.913	9. $t_8 \rightarrow m_1$
t_{10}	158.974-230.769	-	-		10. $t_4 \rightarrow m_2$

Table 6.7. Comparing the release time of each SMD for each method

SMD	Proposed	PSO	GA	MCT
m_1	297.436	143.59	323.077	**348.718**
m_2	297.778	**395.556**	333.333	306.667
m_3	**331.754**	383.886	241.706	303.318
m_4	295.652	269.565	**347.826**	243.478

Table 6.8. Objective comparison

Objective	Criteria	Proposed	PSO	GA	MCT
Makespan	Minimise	331.754	395.556	347.826	348.718
RU	Maximise	0.921	0.754	0.896	0.862
LB	Minimise	15.09	101.912	41.236	37.491

6.2.2.4 Computation Complexity Analysis

The complexity analysis of the proposed algorithm consists of two phases i) complexity during prioritisation of the tasks based on available resource measures, and ii) efficient assignment of tasks to its appropriate SMDs by incorporating the considered objectives. There exist *n* number of tasks, *m* number of SMDs and *r* number of resources in an MCC. According to Algorithm 6.1, the complexity calculations of individual segments are as follows:

a) The effective CPU and usable battery calculations using Eq. 6.1 and Eq. 6.2 require a total $O(m)$.

b) Resource strength calculation using Eq. 6.16 needs $O(r \times m)$.

c) Resource cost estimation requires $O(n \times m)$.

d) Prioritizing the task execution order needs $O(n)$.

e) Finally, mapping a task to SMDs requires $O(n^2 \times m)$.

Hence, the overall time complexity is $O(m) + O(r \times m) + O(n \times m) + O(n^2 \times m) \approx O(n^2 \times m)$.

6.2.3 Experiment, Results and Analysis

This section presents an extensive simulation of the proposed algorithm on a real dataset. The performance is compared with three other algorithms simulated on the same data set.

6.2.3.1 Data Curation

We used the collected data as described in Chapter 4. In this experiment, we considered CPU, RAM, battery, and temperature of the SMDs. We considered a total of ten SMDs' ($M_2 = \{m_1, m_2, m_3, \dots, m_{10}\}$) data, chosen randomly from the original dataset. The complete details of M_2 are shown in Table 6.9. We also wanted to check the performance of the proposed scheduling algorithm on a smaller and larger set of tasks on a smaller and larger number of SMDs. Therefore, from M_2, we randomly took out another smaller set ($M_1 = \{m_1, \dots, m_k, \dots, m_5\}$) consisting of five SMDs.

The raw collected data needed to be further prepared for the experiment. The first thing we did to calculate $P(m_k)$ and $B(m_k)$ using Eq. 6.1 and Eq. 6.2, respectively. The values of $R(m_k)$ and $T(m_k)$ are considered as they are.

Since each parameter has different units, it is difficult to introduce equal importance to all the parameters. Therefore, we normalised the considered parameters. The normalised value of the j^{th} resource of the i^{th} SMD is calculated by Eq. 19.

$$N_{ji} = \frac{v_{ji}}{\sum_{i=1}^{m} v_{ji}} \tag{6.19}$$

where, v_{ji} is the considered value (shown in Table 6.9) of the j^{th} resource of i^{th} SMD, and m is the total number of available SMDs.

To eliminate any undefined or zero normalised value, we replaced the zero values

of v_{ji} in the datasets with 1, as shown in Eq. 6.20.

$$v_{ji} = 1, \text{ if } v_{ji} == 0 \tag{6.20}$$

Table 6.9. Details of the dataset used in the experiment

SMD	CPU frequency (GHz)	Number of CPU cores	CPU load (%)	Effective CPU	Available RAM (MB)	Battery capacity (mAh)	Available battery (%)	Actual battery	Device temp (C°)
m_1	1.3	8	13	0.80	1807	3000	10	300	69
m_2	2.5	2	53	0.09	1767	4000	24	960	89
m_3	2.2	8	19	0.93	1916	3000	11	330	71
m_4	1.7	2	62	0.05	2855	4000	22	880	72
m_5	1.5	4	71	0.08	2851	3500	31	1085	81
m_6	1.3	4	11	0.47	3537	3500	37	1295	74
m_7	1.3	8	12	0.87	2755	3000	92	2760	73
m_8	1.3	8	22	0.47	2690	4000	56	2240	67
m_9	2.5	8	94	0.21	2628	3500	69	2415	82
m_{10}	1.3	8	91	0.11	1753	4000	29	1160	84

Each parameter has associated weight values depending on its importance, as shown in Table 6.10. However, depending on the MCC application type and requirement, these weights might need to be adjusted.

Table 6.10. Details of the effective parameters used in the algorithm

Considered parameters	Considered value calculation	Parameter weight	Ideal value
Effective CPU	Calculated using Eq. 6.1	0.30	Maximized
RAM	Present available RAM (MB)	0.30	
Effective battery	Calculated using Eq. 6.2	0.30	
Device temp	On a scale of 0-100°C	0.10	Minimised

6.2.3.2 Simulation Provisioning

In this section, we discuss the requirements and considerations to set up the simulation environment for the experiment.

6.2.3.2.1 Experimental Setup

We performed the simulations to demonstrate our proposed work on a system running on Intel(R) Core™ i7-5500U CPU with 2.40 GHz and 4 GB of RAM. It was implemented on Ubuntu 16.04 using 'C' programming language.

6.2.3.2.2 Task Initiation

For the experiment, we created two task groups – T^r and T^f. T^r was further divided into two subgroups – $T_1^r = \{t_{1_1}^r, t_{1_2}^r, t_{1_3}^r, \dots, t_{1_{100}}^r\}$ and $T_2^r = \{t_{2_1}^r, t_{2_2}^r, t_{2_3}^r, \dots, t_{2_{200}}^r\}$,

where $T_1^r \neq T_2^r$. For each of T_1^r and T_2^r, we generated four different sets of tasks (i.e., a total of eight) such that $T_1^{r_i} \neq T_1^{r_j}$ and $T_2^{r_i} \neq T_2^{r_j}$. The tasks were randomly generated with the instruction length ranging from 5 to 250.

T^f was divided into two task sets – $T_1^f = \{t_{1_1}^f, t_{1_2}^f, t_{1_3}^f, \ldots, t_{1_{100}}^f\}$ and $T_2^f = \{t_{2_1}^f, t_{2_2}^f, t_{2_3}^f, \ldots, t_{2_{200}}^f\}$, where $T_1^f \neq T_2^f$. The instruction lengths of the tasks of both T_1^f and T_2^f were between 1 to 150.

The details of task generation and bifurcation are shown in Fig. 6.3. The rationale behind considering these varieties of task sets is to establish the efficiency of the proposed algorithm in diverse scenarios, as discussed in Section 6.2.3.3.

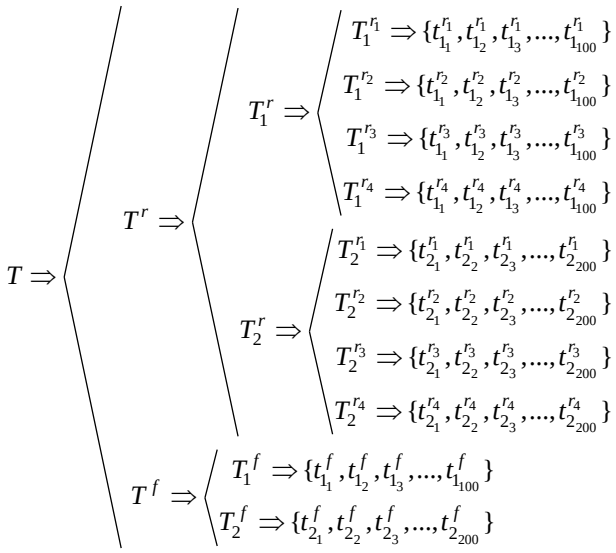


Fig. 6.3. Different task sets used in the experiment

6.2.3.2.3 Control Parameters

The details of the control parameters of PSO and GA are listed in Table 6.11. Since MCT is a heuristic algorithm, no such specific control parameters are there, and hence not included in the list. We set the weight values for each considered objectives $\omega_1 = 0.4$ and $\omega_2 = \omega_3 = 0.3$.

6.2.3.3 Performance Analysis

To assess and analyse the performance of the proposed scheduling algorithm, we performed an extensive simulation on the real dataset as described above. For a comparative assessment, we performed the same experiment with the same assumed scenarios using three other popular algorithms. Since our algorithm is a

heuristic one, we deliberately chose one heuristic (MTC) and two metaheuristic (PSO and GA) algorithms that are popularly used in similar problem scenarios.

Table 6.11. Control parameters for PSO and GA

Algorithm	Parameter		Values
PSO	Population size		10
	Iterations		100
	Initial parameters	C_1, C_2	1.0962
		r_1, r_2	Rand(0,1)
		w	0.0968
	Initial social influence		Rand(0,1)
	Initial personal influence		Rand(0,1)
	v_{max}		0.5
	v_{min}		0.5
	w_{max}		1.0
	w_{min}		0.0
GA	Population size		10
	Iterations		100
	Crossover point		40
	Mutation rate		0.1

As discussed in Section 6.2.3.2.2, we initiated task sets of different sizes – one consisting of 100 (T_1) and another consisting of 200 (T_2) subtasks. These tasks were scheduled to two SMD sets ($M_1$and M_2, $M_1 \subseteq M_2$). Considering this setup, to evaluate the efficiency of the proposed algorithm in diverse MCC scenarios, we divided our experiment into two approaches. In the first, the sets of the same task size were scheduled to particular sets of SMDs, i.e., T_1 and T_2 were scheduled to M_1 and M_2, respectively. In the second case, T_1 and T_2 were scheduled to both M_1 and M_2, alternatively, as shown in Fig. 6.4.

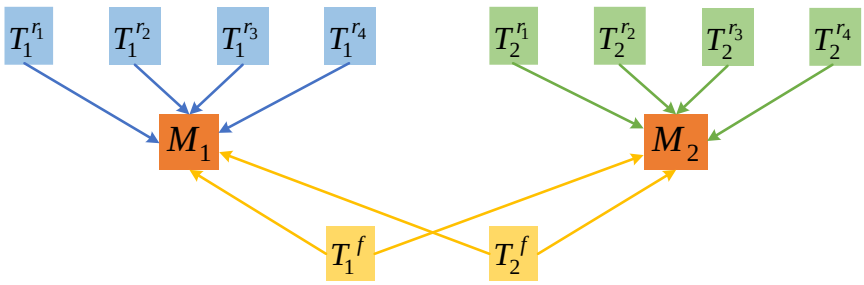


Fig. 6.4. Two experimental scenarios of task-SMD mapping

6.2.3.3.1 Experiment Case I

In the first provision of the experiment, we aimed to assess the overall effectivity of the algorithm for task heterogeneity. For this, we used eight task sets from the task group T^r, as shown in Fig. 6.3, in which each task had different instruction lengths. The tasks were scheduled in the pair of ($T_1^r \rightarrow M_1$) and ($T_2^r \rightarrow M_2$).

Fig. 6.5, Fig. 6.6, and Fig. 6.7 show the performance of the proposed algorithm along with PSO, GA, and MCT in terms of makespan, resource utilisation, and load balancing, respectively. It can be observed that for all the task sets, our proposed algorithm performs better. To make this inference more intuitive, in Fig. 6.8, Fig. 6.9, and Fig. 6.10, we show the average performance of the algorithms for two different task sizes for all three objective criteria.

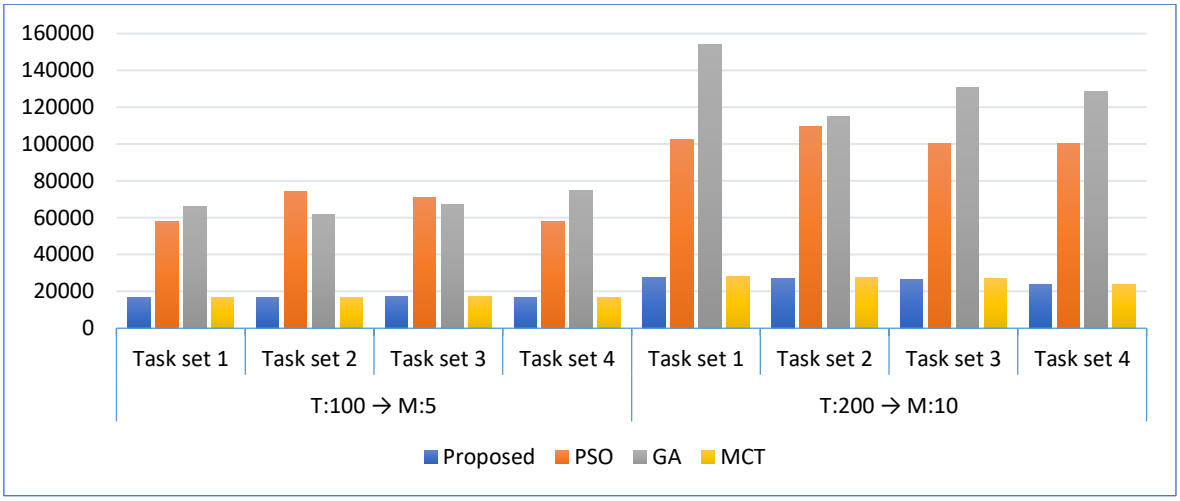


Fig. 6.5. Makespan comparison with different task sizes with eight task sets

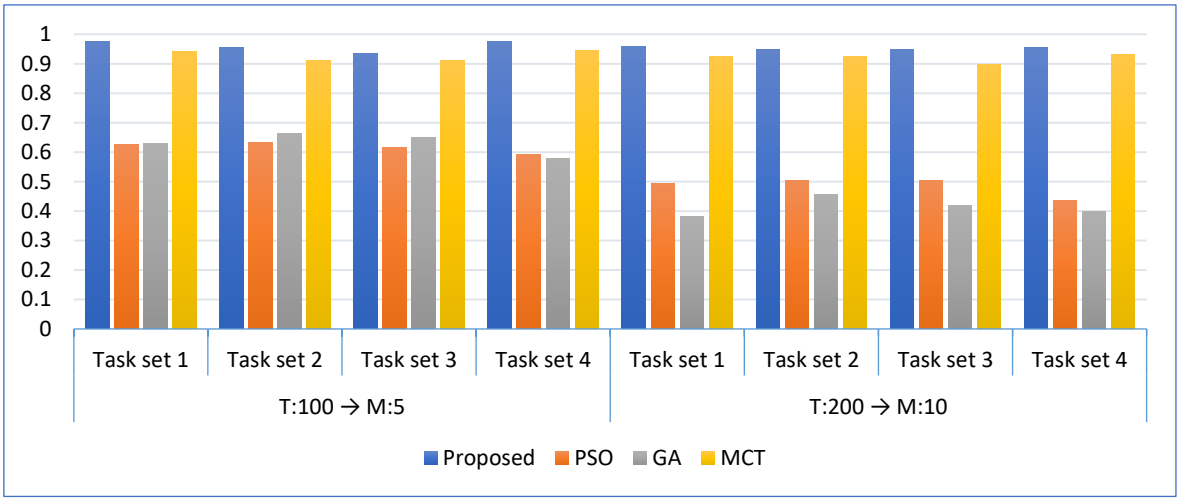


Fig. 6.6. Resource utilisation comparison with different task sizes with eight task sets

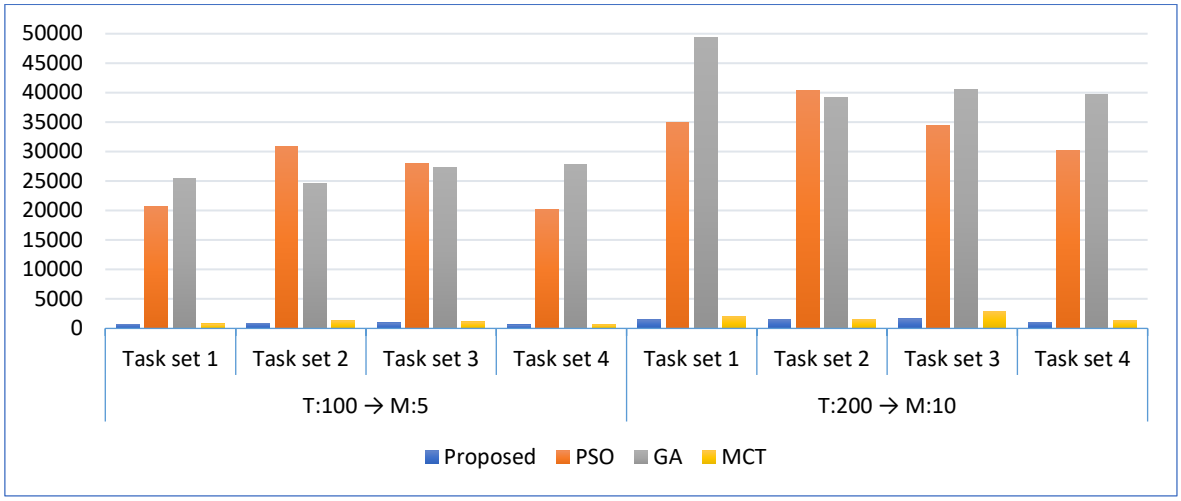


Fig. 6.7. Load balance comparison with different task sizes with eight task sets

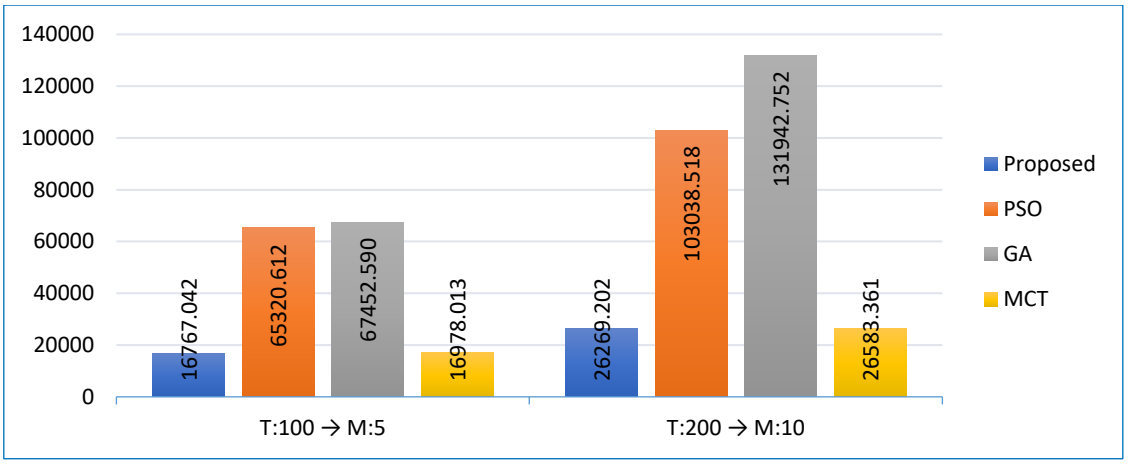


Fig. 6.8. Average makespan comparison with different task sizes

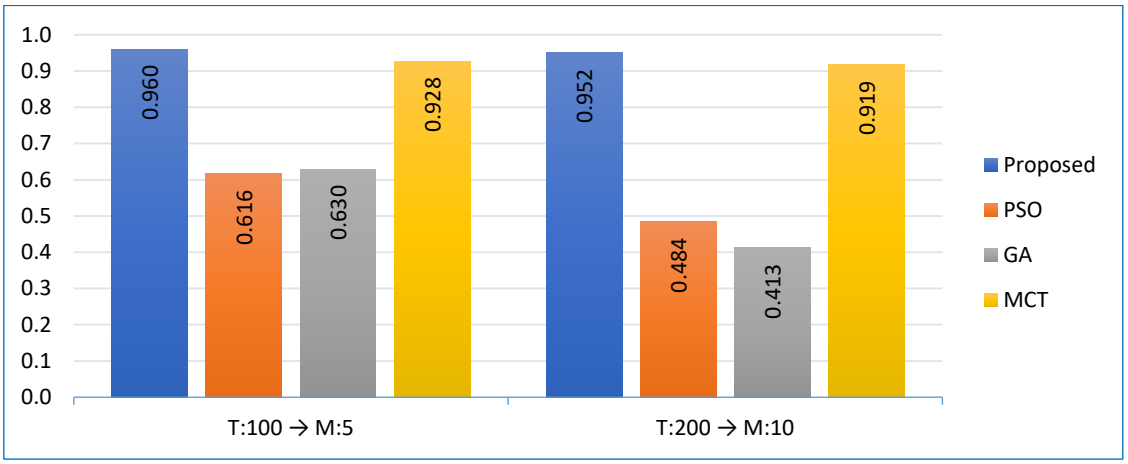


Fig. 6.9. Average resource utilisation comparison with different task sizes

6.2.3.3.2 Experiment Case II

To infer more from the experiment, we wanted to assess the consistency in the performance of the proposed algorithm with SMD variation with the same task sizes. For this, we used the task group T^f. As shown in Fig. 6.4, the algorithm was

tested in four different task-SMD pairs - $(T_1^f \rightarrow M_1)$, $(T_2^f \rightarrow M_1)$, $(T_1^f \rightarrow M_2)$, and $(T_2^f \rightarrow M_2)$. For this case, the makespan, resource utilisation and load balance performances of the proposed algorithm and other compared algorithms are shown in Fig. 6.11, Fig. 6.12, and Fig. 6.13, respectively. In this case, also, we observe that the proposed algorithm performs better than PSO, GA and MCT in all aspects.

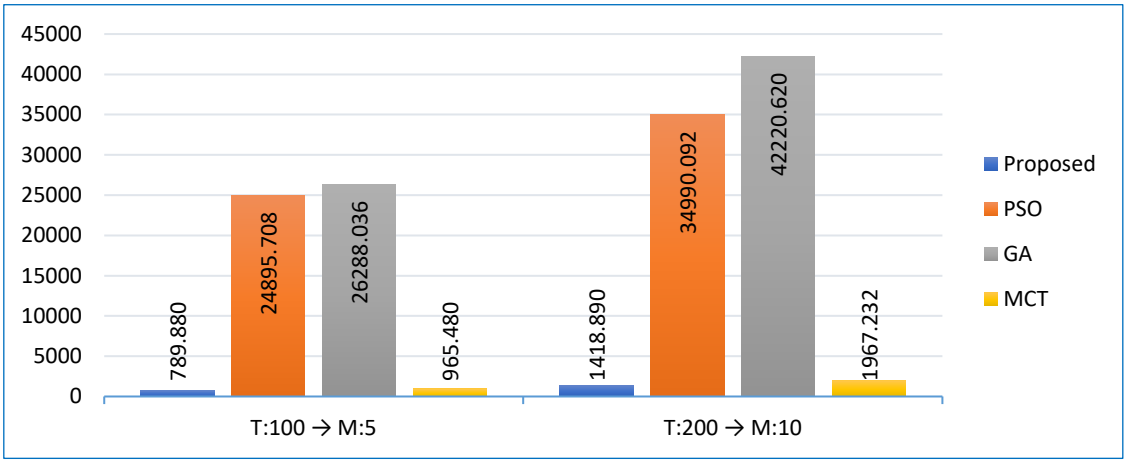


Fig. 6.10. Average load balance comparison with different task sizes

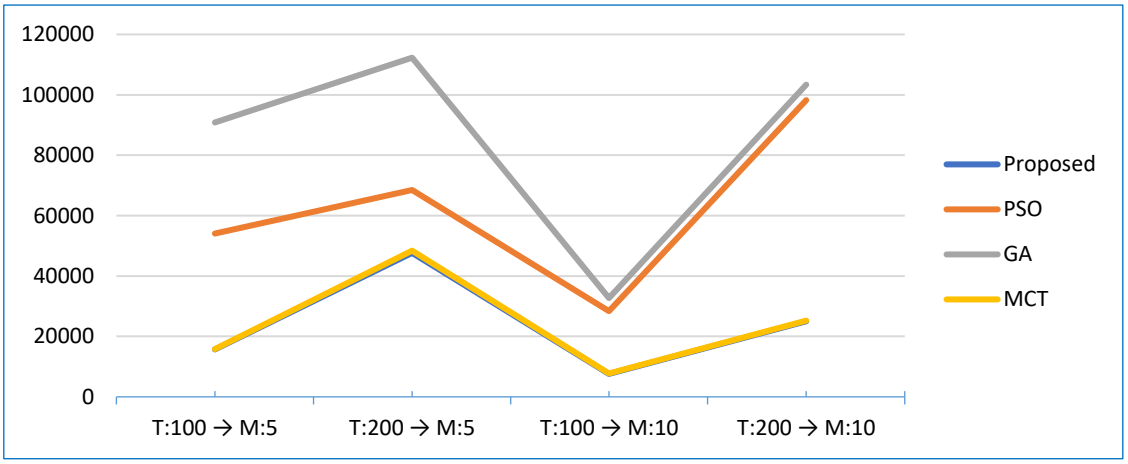


Fig. 6.11. Makespan comparison with same task size

6.2.3.4 Statistical Analysis

In this section, we present two statistical analyses to assess the dominance of the proposed algorithm compared with the other three considered algorithms.

6.2.3.4.1 ANOVA

ANOVA [694] is a well-known statistical method for hypothesis testing. It allows checking the significance of the results by determining the equality of the means of the considered groups. We conducted the one-way ANOVA test to ensure that

our proposed algorithm is significantly varied from other algorithms.

The null hypothesis (H_0) assumes that the mean of all the groups is equal, and based on that, one group can be rejected. Whereas the alternative hypothesis (H_1) assumes means are not equal.

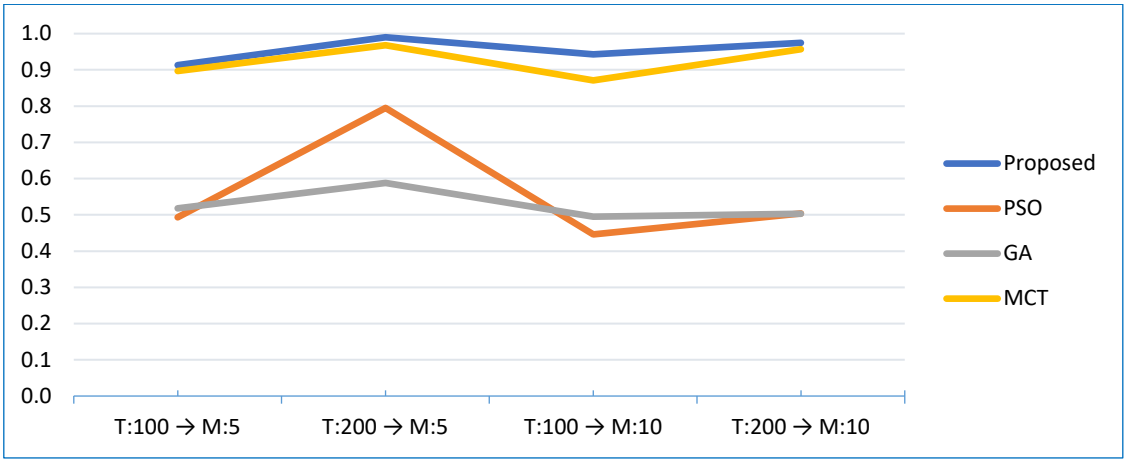


Fig. 6.12. Resource utilisation comparison with same task size

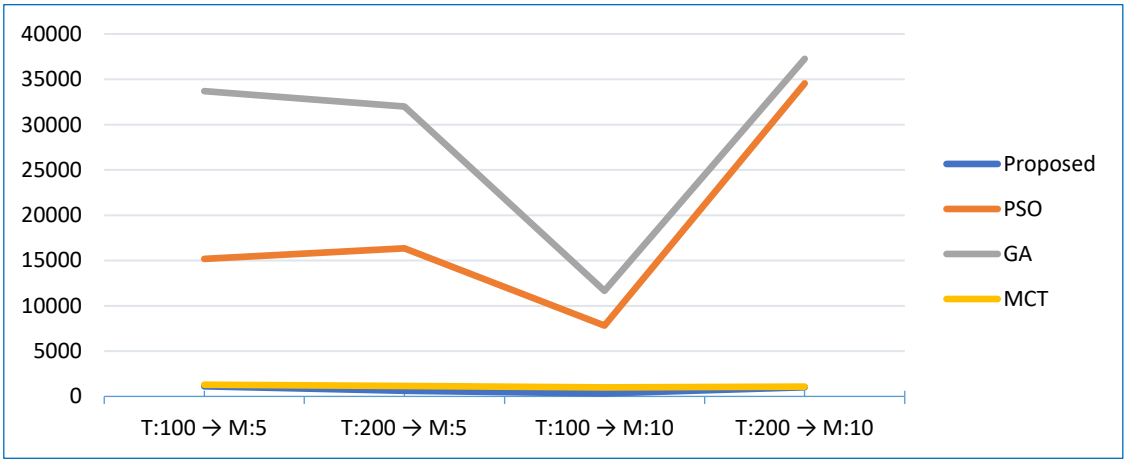


Fig. 6.13. Load balance comparison with same task size

We considered eight sets of sample task groups. The groups consist of 100 or 200 tasks, as shown in Fig. 6.3. The task size of each set is different. The ANOVA test was conducted using makespan and load balance. We did not consider resource utilisation values because it ranges from 0 to 1, which would produce a non-significant result. The H_0 was rejected if the P-value < α, α =0.05, i.e., F-statistic >> F-critical.

The descriptive statistics of the input set for the ANOVA test using makespan and load balance are given in Table 6.12. It can be observed that the proposed algorithm

has the lowest mean, whereas GA has the lowest variance. The results of the ANOVA test using makespan and load balance are shown in Table 6.13 and Table 6.14, respectively. It can be observed from both the tables that the proposed algorithm exhibits dissimilar mean and variance compared to other algorithms. Therefore, we rejected the null hypothesis at the one percent level.

Table 6.12. Input sets of ANOVA test using makespan and load balance

Groups	Count	Makespan			Load balance		
		Sum	Mean	Variance	Sum	Mean	Variance
Proposed	8	172144.975	21518.12	27096272.62	8835.08	1104.385	149600.198
PSO	8	673436.521	84179.57	446158010.5	239543.202	29942.900	48622828.09
GA	8	797581.367	99697.67	1314262960	273183.521	34147.940	85744322.78
MCT	8	174245.495	21780.69	27808493.14	12314.902	1539.363	453088.919

Table 6.13. ANOVA test results using makespan

Source of variation	SS	DF	MS	F-statistic	P-value	F-critical
Between groups	40488110602	3	13496036867	29.73799489	7.53E-09	2.946685266
Within groups	12707280153	28	453831434	-	-	-
Total	-	31	-	-	-	-

Table 6.14. ANOVA test results using load balance

Source of variation	SS	DF	MS	F-statistic	P-value	F-critical
Between groups	7622976648	3	2540992216	75.30548206	1.61E-13	2.946685266
Within groups	944788879.9	28	33742460	-	-	-
Total	-	31	-	-	-	-

6.2.3.4.2 Post Hoc

After confirming the superiority of the proposed algorithm in terms of mean difference, we performed a post hoc analysis [695] to further assess the algorithm's success. The ANOVA test merely indicates a difference between the groups, but it does not specify which groups. The post hoc test exposes the specific differences between the group means when the ANOVA test is significant.

It can be observed from Table 6.14 that the means are different. A 2-sample T-test was conducted for post hoc analysis. The alpha level was adjusted via the Bonferroni method (adjusted alpha=0.016666). Table 6.15 shows that the pairs of (proposed and PSO) and (proposed and GA) significantly differ. The mean difference between proposed and PSO is very low compared to the other two groups. Hence, it can be concluded that this group (proposed and PSO) is significantly different from others.

Table 6.15. Post hoc test results

Groups	P-value (T-test)	Significant?
Proposed - PSO	1.10717E-06	Yes
Proposed - GA	3.05123E-05	Yes
Proposed - MCT	0.921586675	No

6.2.4 Discussion

We represented the experiment scenario and result analysis in two phases. First, the algorithm aims to prove its efficacy with task heterogeneity (different task sets) over fixed SMDs. Second, its efficacy is further analysed and validated with SMD variability (tasks of fixed size operated on different SMD sets). The results of both the cases suggest that irrespective of the variations in the size of the task sets, instruction lengths of the tasks, the number of SMDs, and their resource parameters, the performance of the proposed algorithm consistently remains better.

In Fig. 6.5 to Fig. 6.13, we witness that the proposed algorithm outperforms PSO and GA while performing moderately better than MCT over all the considered objectives inflicted simultaneously. Further from Fig. 6.5 to Fig. 6.13 and Table 6.12 to Table 6.15, we realise that among the other three compared algorithms, MCT is nearest to our algorithm for all the objectives and in all the scenarios, while PSO digresses most.

The observed performance of our algorithm was achieved due to the appropriate sequence of task executions based on the overall resource status of the SMDs. Before the actual scheduling, the algorithm always ensures the earliest start time of a task along with the load distribution and the proper utilisation of all the SMDs. Thus, the algorithm returns a better makespan along with load balance and resource utilisation simultaneously.

The consistency and scalability of the proposed algorithm would allow the MCC system developers to adopt the right task scheduling method without worrying about the size of the MCC both in terms of problem size (that is to be executed in MCC) and the availability of the resources (number of SMDs in the MCC). Based on the MCC application type, the weight values of resources (θ_i) and objectives (ω_i) can be tuned based on the preference and priority of the resource requirements and system goals.

6.3 Energy-efficient Scheduling

In this section, we present a metaheuristic scheduling algorithm based on PSO that considers the overall energy consumption to carry out certain MCC task along with load balance among the SMDs.

6.3.1 Overview of PSO

PSO is a nature-inspired population-based optimisation method developed by Kennedy and Eberhart [696] and Eberhart and Shi [697]. It is inspired by the social behaviour of animals like bird flocking and fish schooling. This swarm intelligence-based metaheuristic technique has been popularly used to solve different optimisation problems in various domains. Initially, a swarm of particles is randomly generated within the multidimensional problem search space. Each particle attempts to move towards the optimum solution. After each movement, the particle assesses its personal best position (P_{best}) and the global best position (G_{best}) within the swarm. The next movement of an individual particle depends on these two pieces of knowledge acquired in the previous movements. A particle should always provide a complete solution to the problem. The ith particle of gth generation can be represented in Eq. 6.21.

$$PR_i^g = \{pos_{(i,1)}^g, pos_{(i,2)}^g, \ldots, pos_{(i,D)}^g\} \tag{6.21}$$

where, D be the dimension of the solution space. All the particles have positions value, $pos_{(i,j)}^g, 1 \leq j \leq D$ and corresponding velocity, $vel_{(i,j)}^g$. The personal best of particle PR_i^g is defined by Eq. 6.22.

$$P_{best\,i} = \{Pb_{(i,1)}, Pb_{(i,2)}, \ldots, Pb_{(i,D)}\} \tag{6.22}$$

At every iteration, the velocities of the particles are updated by Eq. 6.23.

$$vel_{(i,1)}^g = \propto \times vel_{(i,1)}^{(g-1)} + c_1 \times r_1 \times \left[Pb_{(i,j)} - pos_{(i,j)}^{(g-1)}\right] + c_2 \times r_2 \times [G_{best\,j} - pos_{(i,j)}^{(g-1)}] \tag{6.23}$$

where, $\propto$ denotes inertia weight which restricts the uncontrolled velocity of the particles. c_1 and c_2 are the acceleration coefficients. r_1 and r_2 are two different random generated numbers in the range (0, 1]. After updating velocity, the latest positions

($pos_{(i,j)}^{g}$) are updated using Eq. 6.24.

$$pos_{(i,j)}^{g} = pos_{(i,j)}^{(g-1)} + vel_{(i,j)}^{g} \tag{6.24}$$

After updating the positions of a particle, the new fitness value is measured. Based on the updated fitness value, $Pbest_i$ and G_{best} are also updated. The position and velocity of all the particles are updated iteratively to achieve a rational and better solution. The better solution of the particles is measured using the designed fitness function. The iterative updation process is terminated when an iteration constraint is obtained.

6.3.2 System Model and Problem Formulation

The system and execution models of the proposed energy efficient scheduler for MCC are discussed below. We also formally establish the addressed problem. The system model described in Section 6.2.1.1 is considered here also.

6.3.2.1 Execution Model

Let us assume, at a time instant (τ), the coordinator divides a task ($T = \{t_1, t_2, t_3, \dots, t_n\}$) into a set of independent microtasks. These microtasks need to be scheduled to the available set of SMDs ($M = \{m_1, m_2, m_3, \dots, m_m\}$). We consider a real MCC scenario where the SMDs have a different number of CPU cores with different clock frequencies depending on which the computing capability of each SMD varies. The available computing power of an SMD also depends on its present load. A highly loaded SMD would be able to execute a smaller number of microtasks compared to a lightly loaded SMD, given the same turnaround time. Therefore, the effective CPU of an SMD at τ is calculated using Eq. 6.25.

$$P(m_k) = \frac{CPU\_frequency \times CPU\_cores}{CPU\_load^{\tau}} \tag{6.25}$$

The execution time of a task t_j on an SMD m_k is calculated by Eq. 6.26.

$$X(t_j, m_k) = \frac{z_{t_j}}{P(m_k)} \tag{6.26}$$

where, $\boldsymbol{z_{t_j}}$ denotes the instruction size (MIPS) of t_j.

An ideal MCC should utilise all the available SMDs judiciously so that the

computing loads of all the SMDs are evenly balanced as far as possible. The overall load balance of an MCC with respect to a set of MCC tasks is calculated by Eq. 6.27.

$$LB = \sqrt{\frac{\sum_{k=1}^{r}\{R_T^{avg} - R_T(m_k)\}^2}{M}} \tag{6.27}$$

where $R_T(m_k)$ denotes the k^{th} SMD's release time and R_T^{avg} denotes average release time and is defined by Eq. 6.28.

$$R_T^{avg} = \frac{1}{M}\sum_{k=1}^{m} R_T(m_k) \tag{6.28}$$

6.3.2.2 Computational Energy Calculation

Different SMD processors operate at different speeds, with different voltage and frequency with respect to t_j. The cumulative voltage (V_M) and frequency (F_M) are given by Eq. 6.29 and Eq. 6.30.

$$F_M = \{f_{m_1}, f_{m_2}, f_{m_2}, \dots, f_{m_m}\} \tag{6.29}$$

$$V_M = \{v_{m_1}, v_{m_2}, v_{m_2}, \dots, v_{m_m}\} \tag{6.30}$$

Here, $f_{(m_k)}$ and $v_{(m_k)}$ denote the operating frequency and supply voltage of k^{th} SMD's CPU, assuming each core has the same operating frequency and voltage.

In heterogeneous multiprocessing, the multi-core SMD CPUs are comprised of two different sets of cores paired together into a single unit. It is done so to make a balance between performance and energy efficiency. One set of cores is less powerful and more energy-efficient than the other. The decision to submit jobs to the appropriate core is taken dynamically by mapping to the varying computational demand of the application. Normally, the first set of cores deals with the regular background tasks that require nominal energy, whereas the user-interactive and performance-demanding tasks are run on the second set of cores. In the second case, energy consumption is a big concern, so minimising it is highly desired.

The total energy consumption of an SMD CPU is the combined energy consumption of both the high- (E_H) and low-performance (E_L) core sets, as shown in Eq. 6.31. We assume the energy consumption due to network communication for all SMDs is constant.

$$E_{comp}(m_k) = E_H(m_k) + E_L(m_k) \tag{6.31}$$

The energy consumption is directly related to the total power dissipation caused by high (P_H) and low (P_L) power dissipation. P_H is calculated using Eq. 6.32, whereas P_L arises due to running, bias and leakage of currents and is calculated using Eq. 6.33.

$$P_H(m_k) = K \times (v_{(m_k)})^2 \times f_{(m_k)} \tag{6.32}$$

$$P_L(m_k) = K \times (v_{lowest(m_k)})^2 + f_{lowest(m_k)} \tag{6.33}$$

where K is the capacitance load.

As already mentioned, the E_H consumption is done when the SMD's CPU performs extensive computation, operating in high voltage and frequency. Therefore, E_H can be calculated using Eq. 34.

$$E_H(m_k) = \sum_{j=1}^{n} P_H(m_k) \times X(t_j, m_k) \tag{6.34}$$

On the contrary, E_L consumption is done when the CPU is either idle or performing very low-profile background jobs. In this case, we can consider that the CPU (m_k) operates on v_{lowest} and f_{lowest}. Therefore, E_L can be calculated using Eq. 35.

$$E_L(m_k) = \sum_{i=1}^{n} P_L \times t_{L\,(m_k)} \tag{6.35}$$

where $t_{L\,(m_k)}$ denotes idleness or lower activity of m_k.

Remark 6.3. Typically, $E_L \ll E_H$, and the effect of E_L on the scheduling decision would be insignificant. Therefore, in this work, we overlooked E_L and considered only E_H.

6.3.2.3 Data Transfer Energy Calculation

Receiving the tasks from the MCC coordinator, i.e., downloading them through Wi-Fi and uploading or sending the results back to the coordinator, also involves some energy consumption. The energy consumption (E_C) of m_k for data transfer can be calculated by Eq. 36.

$$E_C(m_k) = \frac{1}{B_{m_k}^{\tau}} \left(\sum_{j=1}^{n} z_{t_j}, m_k \times E_D(m_k) + \sum_{j=1}^{n} z_{r_j}, m_k \times E_U(m_k) \right) \tag{6.36}$$

where, $B_{m_k}^{\tau}$ is the bandwidth of m_k, which is dependent on its signal strength at τ,

$\sum_{j=1}^{n} z_{t_j}$, m_k denotes the total instruction length of all the tasks executed by a particular SMD m_k, z_{r_j} is the size of the result of t_j and $E_D(m_k)$ and $E_U(m_k)$ denote the energy consumption rate for downloading and uploading data blocks (tasks and results), respectively.

Remark 6.4. In Eq. 36, we considered the power consumption accounted for only downloading the MCC tasks and uploading the corresponding results. We assumed that Wi-Fi is by default on and hence ignored the actual power consumption due to Wi-Fi operation.

Remark 6.5. Here, we considered our MCC comprised of a single WLAN with a consistent bandwidth and signal strength for all the SMDs. So, it can be assumed that the power consumed to download the same set of tasks (sent by the MCC coordinator) and upload corresponding results (to be sent to the MCC coordinator) are uniform for all SMDs. Though the power consumption for data transmission slightly varies depending on signal strength (which diverges as per the distance of the SMD from the AP), it can be considered negligible [698]. Also, the data transfer rate variations for different SMDs are minimal.

Remark 6.6. We further assumed that our MCC application has mostly computation-bound tasks. Computation-bound tasks typically have high CPU utilisation with a smaller extent of data transfer [253]. This suggests higher energy consumption for processing compared to data transferring.

From the above remarks, it can be concluded that the energy consumption due to data transfer became almost inconsequential in the scheduling decision. Hence, we ignored the communication power consumption for the further experiment in this work. We set our focus on the energy consumed by the SMDs accounting only to execute the MCC tasks.

6.3.2.4 Final Objective

Therefore, the final problem can be stated as designing a scheduling algorithm conforming to two objectives, as follows:

i) **Objective I:** Minimise the overall energy consumption considering

computational energy along with communication energy, i.e., $\min(E_H(m_k) + E_L(m_k) + E_C(m_k)) \Rightarrow \min((E_H(m_k))$, neglecting $E_L(m_k)$ and $E_C(m_k)$.

ii) **Objective II:** Minimise the overall energy consumption with load balance, i.e., $\min(E_H(m_k) + LB)$.

6.3.3 Proposed PSO-based Energy-aware Scheduling for MCC

In this section, we present the proposed PSO-based scheduling algorithm along with demonstrative illustrations for both objectives.

6.3.3.1 Particle Representation

A particle (PR_i^g) must be represented in such a way that it should always provide a complete solution to the problem. A particle is always associated with its position ($pos_{(i,j)}^{(g)}$) and velocity ($vel_{(i,j)}^{(g)}$), where $1 \leq j \leq D$. Here dimension of a particle is equal to the number of tasks, i.e., $D = n$. A particle representation is shown in Table 6.16. The first row denotes the task scheduling sequence, while the second row denotes the position values as SMD index (m_k).

From Table 6.16, we understand that five tasks are initiated by the MCC coordinator, which need to be executed by three SMDs. It can also be observed that the first task (i.e., t_1) is assigned to the second SMD (i.e., m_2). Likewise, t_2, t_3, t_4, and t_5 are assigned to m_3, m_1, m_2, and m_3, respectively. This is how our particle representation produces a complete solution.

Table 6.16. Particle representation

Tasks	t_1	t_2	t_3	t_4	t_5
Position	2	3	1	2	3

Algorithm 6.2: Generate_Population()

Input: Size of the population N and dimension of the particle D

Output: Generation of population $P_{pop} = \{G_1, G_2, \ldots, G_N\}$ //G_i denotes the i^{th} iteration

```
1. P_pop = {} //initial population is null
2. for (i = 1 to N)
3.    for (d = 1 to D)
4.       pos_(i,d)^(g) = Rand(1, M) //M is total number of available SMDs
5.    end for
6.    G_i = {pos_(i,d)^(g) ∨∀i, 1 ≤ i ≤ N, ∀d, 1 ≤ d ≤ D}
7.    P_pop = P_pop ∪ G_i
8. end for
```

The initial population (P_{pop}) is generated as per Algorithm 6.2, which is depicted in Table 6.17. It can be observed from the table that the initial population consists

of three particles (N = $\{\boldsymbol{PR_1^0}, \boldsymbol{PR_2^0}, \boldsymbol{PR_3^0}\}$) with dimension five where each $\boldsymbol{PR_i^0}$ provides a complete solution.

Table 6.17. Initial population of three particles with dimension five (same as number of tasks)

$\boldsymbol{PR_i^g}$	$\boldsymbol{pos_{(i,1)}^g}$	$\boldsymbol{pos_{(i,2)}^g}$	$\boldsymbol{pos_{(i,3)}^g}$	$\boldsymbol{pos_{(i,4)}^g}$	$\boldsymbol{pos_{(i,5)}^g}$
PR_1^0	1	3	2	1	2
PR_2^0	2	1	3	2	1
PR_3^0	2	3	1	2	3

6.3.3.2 Fitness Calculation

The fitness function evaluates the excellence of particles. Here, the fitness function is calculated in two phases, corresponding to two objectives mentioned in Section 6.3.2.4. Our final objective is to minimise fitness, as given in Eq. 6.37.

$$\min(\text{fitness}) = \sum_{i=1}^{k} w_i \times Objective_i \tag{6.37}$$

where, k and w_i denote the number of objectives and their weight values, respectively.

Here, we followed the weight sum approach for fitness calculation. For Objective I, since there is only a single parameter (energy efficiency), the value of w is 1. For Objective II, the value of w for both parameters (energy efficiency and load balance) are equally assigned, i.e., 0.5 for each.

6.3.3.3 Velocity and Position Updation

Along with position and velocity, after each iteration, the particle PR_i^g has a personal best value (P_{best_i}), as shown in Eq. 6.22. and a global best (G_{best}) among all the particles within the swarm. At every iteration, the velocities and the positions of the particles are updated by Eq. 6.23 and Eq. 6.24.

During the position's updation, new positions may violate the restricted position range, i.e., [1,m]. If the updated position becomes negative, less than one, or greater than m, it maps to a random position [1,m]. This is how n number of particles are updated iteratively.

After the updation of the positions of a particle, the new fitness value is measured. Based on the new fitness value, P_{best_i} and G_{best} are also updated as discussed in Section 6.3.1. The position and velocity of all the particles are updated iteratively till the termination condition is satisfied (MAX_{itr}). After reaching MAX_{itr}, the

particle with minimum fitness, i.e., $G_{best}(MAX_{itr})$ is chosen as the final solution. The pseudocode of the proposed PSO-based scheduling algorithm is shown in Algorithm 6.3. The overall pictorial representation of the algorithm is depicted in Fig. 6.14.

Algorithm 6.3: PSO-based Energy-Efficient Scheduling for MCC

Input: (1) N number of particles, i.e., $N = \{PR^{g}_{1,D}, PR^{g}_{2,D}, \dots, PR^{g}_{n,D}\}$
(2) Termination criterion is MAX_{itr}

Output: Best solution as the minimised *fitness* value

```
1.  Generate initial_population P_pop = {} //as per Algorithm 6.2
2.  Compute fitness() using Eq. 6.37 //considering energy as well as load balance
3.  Initialise P_best ∀ i ∈ N and G_best from the swarm
4.  while (i ≤ MAX_itr) do  //termination condition (MAX_itr) as number of iterations
5.    for (i = 1 to N)
6.      Update velocity () and position () using Eq. 6.23 and Eq. 6.24
7.      if (fitness(pos_(i,j)^(g)) > fitness(P_best)) then
8.        P_best = pos_(i,j)^(g)
9.      end if
10.     if (fitness(pos_(i,j)^(g)) > fitness(G_best)) then
11.       G_best = P_best
12.     end if
13.     i++;
14.   end for
15. end while
16. Select G_best(MAX_itr) as the final solution
```

Remark 6.7. It can be observed from the particle representation that the particle always guarantees to produce a complete solution to the problem. Moreover, while updating the velocity and position of a particle, it always preserves its predefined range of position values, i.e., [1,m]. Therefore, our particle representation always provides a valid solution even after the updating phase.

Remark 6.8. The computation of a particle's fitness function (as per Eq. 6.37) is evaluated locally in an iterative manner without having any advance or global knowledge of the particles or swarm along with their respective fitness values. Therefore, the fitness calculation is done independently, irrespective of other solutions.

6.3.3.4 Illustration

In this section, we demonstrate the working of the proposed algorithm with a set of synthetic data. Let us consider, at a time instance (τ), four SMDs (m_1, m_2, m_3, m_4) are available with an MCC system, and the MCC coordinator has eleven independent subtasks ($t_1, t_2, t_3, \dots, t_{11}$) to execute on these SMDs. Each task

has a different instruction length (task size). The MCC needs to schedule the tasks for the SMDs conforming to the minimum energy efficiency criteria with load balance. We present the illustration in two segments: scheduling considering only energy efficiency and scheduling considering energy efficiency with load balance.

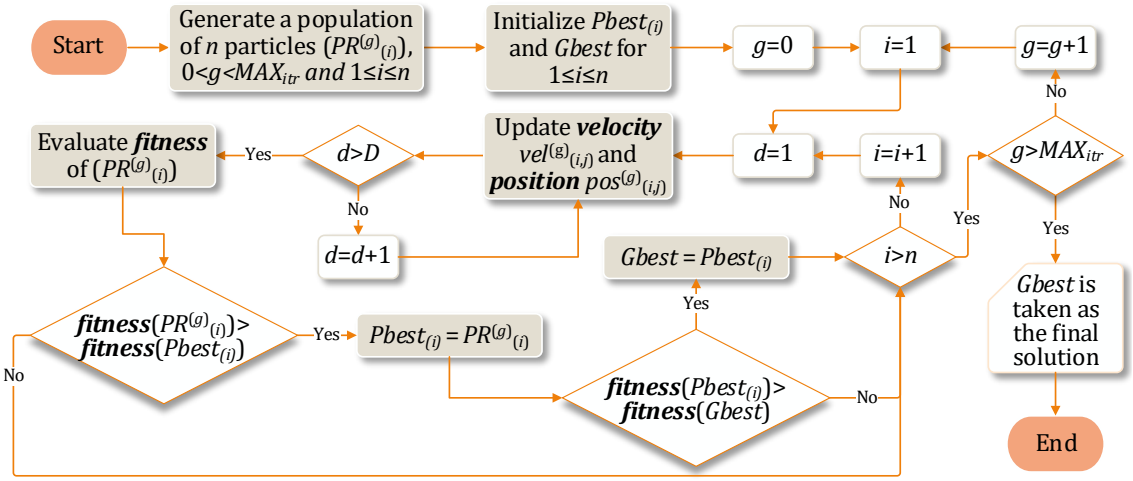


Fig. 6.14. Flowchart of the proposed algorithm

6.3.3.4.1 Energy-efficient Scheduling

Table 6.18 shows the details of scheduling using the proposed algorithm. The 2nd column shows the instruction length of each task. Columns 3rd to 6th show the start and finish times of the tasks assigned to the corresponding SMDs based on the execution time of that task on that SMD, given in the 8th column. For instance, the task t_2 has an instruction length of 61, and to execute it on m_3, its required time is 630.165. It can further be seen (in the 5th column) that t_2 starts its execution on m_3 at 165.289 and completes at 795.455. Column seven shows the complete scheduling sequence of the eleven tasks on four SMDs.

Energy consumed by each SMD to execute the assigned tasks and the total energy consumption to complete the eleven tasks, are given in Table 6.19. It can be observed that by the scheduling policy, no task was scheduled to m_2 and m_4; hence, their energy consumption is zero. Whereas most tasks (74.048% of total tasks) were assigned to m_3, and to execute the assigned tasks, m_1 was mostly busy (90.166%).

6.3.3.4.2 Energy-efficient Scheduling with Load balance

In the above, we saw that by using the proposed scheduling algorithm though we

achieved a very low overall energy consumption, the scheduling is highly biased to one or two SMDs to minimise the overall energy consumption. However, as mentioned earlier, in MCC, it is vital to ensure that no SMDs are get overloaded, i.e., the tasks should be equally distributed as far as possible so that the load of all SMDs are evenly balanced. We incorporated the load balance factor in the proposed scheduling algorithm to address this. The scheduling details for this case are shown in Table 6.20.

Table 6.18. Scheduling sequences without load balancing

Task	Task size	Execution duration				Schedule sequence	Execution time
		m_1 (v: 0.259, f: 0.004)	m_2 (v: 0.527, f: 0.021)	m_3 (v: 0.107, f: 0.097)	m_4 (v: 0.882, f: 0.017)		
t_1	16	-	-	0 - 165.289	-	1. $t_1 \rightarrow m_3$	165.289
t_2	61	-	-	165.289 - 795.455	-	2. $t_2 \rightarrow m_3$	630.165
t_3	15	0 - 4054.054	-	-	-	3. $t_3 \rightarrow m_1$	4054.054
t_4	46	-	-	795.455 - 1270.661	-	4. $t_4 \rightarrow m_3$	475.207
t_5	9	-	-	1270.661 - 1363.636	-	5. $t_5 \rightarrow m_3$	92.975
t_6	31	-	-	1363.636 - 1683.884	-	6. $t_6 \rightarrow m_3$	320.248
t_7	30	4054.054 - 12162.162	-	-	-	7. $t_7 \rightarrow m_1$	8108.108
t_8	23	-	-	1683.884 - 1921.488	-	8. $t_8 \rightarrow m_3$	237.603
t_9	11	12162.162 - 15135.135	-	-	-	9. $t_9 \rightarrow m_1$	2972.973
t_{10}	28	-	-	1921.488 - 2210.744	-	10. $t_{10} \rightarrow m_3$	289.256
t_{11}	19	15135.135 - 20270.270	-	-	-	11. $t_{11} \rightarrow m_1$	5135.135

Table 6.19. Task lengths for each SMD and energy consumption without load balancing

SMD	Task(s) scheduled	Total task size	% of total task size	Total execution time	% of total engaged time	Energy consumption	Load balance
m_1	t_3, t_7, t_9, t_{11}	75	25.952	20270.27	90.166	5.031	-
m_2	-	0	0	0	0	0.000	
m_3	t_1, t_2, t_4, t_5, t_6, t_8, t_{10}	214	74.048	2210.744	9.832	2.450	
m_4	-	0	0	0	0	0.000	
Total	11	289	100	22481.014	100	7.481	

Now let us check how the issue of biased loading has been mitigated. Table 6.21 shows that in this case, there is no such SMD that has not been assigned any task. From the 4th column, we see that the task distribution (according to task size) has been significantly improved than earlier (Table 6.19). A comparison of the variance

and standard deviation (STD) of the load distribution for both the cases (with and without load balance) in terms of the total size of tasks executed and total execution time is shown in Fig. 6.15. It can be expectedly observed that the variance and STD are much higher when the focus is on only energy efficiency. But introducing the load factor in the scheduling has reduced both the parameters' values. However, to achieve this, we had to sacrifice energy efficiency to some extent. It can be seen from Table 6.21 that the energy consumption is increased a little bit while the total load balance factor of 402.614 is achieved.

Table 6.20. Scheduling sequences with load balancing

Task	Task size	Execution duration				Schedule sequence	Execution time
		m_1 (v: 0.259, f: 0.004)	m_2 (v: 0.527, f: 0.021)	m_3 (v: 0.107, f: 0.097)	m_4 (v: 0.882, f: 0.017)		
t_1	11	-	-	0 - 165.289	-	1. $t_1 \rightarrow m_3$	165.289
t_2	12	-	-	165.289 - 795.455	-	2. $t_2 \rightarrow m_3$	630.165
t_3	5	-	0 - 704.225	-	-	3. $t_3 \rightarrow m_2$	704.225
t_4	6	-	704.225 - 2863.850	-	-	4. $t_4 \rightarrow m_2$	2159.625
t_5	39	0 - 2432.432	-	-	-	5. $t_5 \rightarrow m_1$	2432.432
t_6	41	-	-	-	0 - 1867.470	6. $t_6 \rightarrow m_4$	1867.470
t_7	36	-	-	795.455 - 1105.372	-	7. $t_7 \rightarrow m_3$	309.917
t_8	23	-	-	1105.372 - 1342.975	-	8. $t_8 \rightarrow m_3$	237.603
t_9	25	-	-	1342.975 - 1456.612	-	9. $t_9 \rightarrow m_3$	113.636
t_{10}	28	-	-	1456.612 - 1745.868	-	10. $t_{10} \rightarrow m_3$	289.256
t_{11}	19	-	-	1745.868 - 1942.149	-	11. $t_{11} \rightarrow m_3$	196.281

Table 6.21. Task lengths for each SMD and energy consumption with load balancing

SMD	Task(s) scheduled	Total task size	% of total task size	Total execution time	% of total engaged time	Energy consumption	Load balance
m_1	t_5	39	13.495	2432.432	26.713	0.604	402.614
m_2	t_3, t_4	11	3.806	2863.85	31.450	16.941	
m_3	t_1, t_2, t_7, t_8, t_9, t_{10}, t_{11}	154	53.287	1942.149	21.328	2.152	
m_4	t_6	41	14.187	1867.47	20.508	24.116	
Total	11	289	100	9105.901	100	43.813	

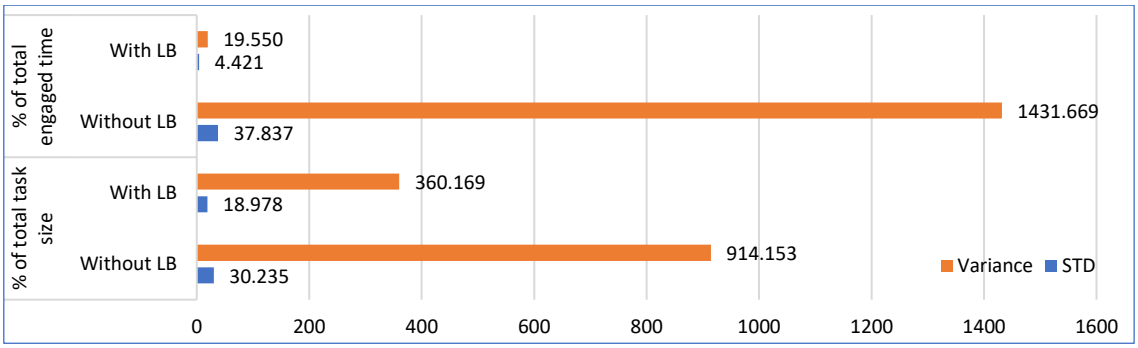


Fig. 6.15. Variance and standard deviation of the load distribution for scheduling with and without load balance in terms of the total size of tasks executed and total execution time

6.3.3.5 Time Complexity Analysis

The complexity analysis of the proposed algorithm is discussed as follows.

- Particle representation and generation of the swarm: The dimension of a particle is the same as the number of tasks (i.e., n). So, the initial population P_{pop} of N particles is generated in $O(N \times n)$.
- Calculation of fitness value: The particle with length n is evaluated in $O(n)$ time along with $O(N)$ for initialising G_{best}.
- Velocity and position updating: Iteratively, the velocity and positions of a single particle are updated in $O(n)$ times. Here, the overall iteration to update the swarm needs $O(MAX_{itr} \times N \times n)$.

Hence, the time complexity of the overall process is $O(N \times n) + O(N) + O(MAX_{itr} \times N \times n)$ time or $O(N \times n)$ time as an upper bound. Therefore, the overall complexity of our PSO-based scheduling algorithm for MCC is $O(N \times n)$.

6.3.4 Experiment, Results and Analysis

In this section, we present the details of the simulated experiment for the energy-efficient scheduling. We also discuss the observed results and analyse them in the context of the set objectives.

6.3.4.1 Dataset Curation

We used the same dataset as mentioned in Section 6.2.3.1. However, we considered only CPU clock frequency, no. of cores, and current CPU load for this experiment. We considered the CPU information of a total of fifteen SMDs' ($M = \{m_1, m_2, m_3, \dots, m_{15}\}$), chosen randomly. We wanted to check the performance of the proposed scheduling algorithm on a smaller and larger set of tasks

on a smaller and larger number of SMDs. Therefore, we further divided M into two sets of SMDs ($M_1 = \{m_1, m_2, m_3, \ldots, m_{10}\}$ and $M_2 = \{m_{11}, m_{12}, m_{13}, \ldots, m_{15}\}$). The complete details of M_1 and M_2 are shown in Table 6.22 and Table 6.23, respectively.

Table 6.22. Dataset used in the experiment: resource parameters details of SMDs of set 1 ($\boldsymbol{M_1}$)

Parameter	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
CPU frequency (GHz)	2.2	1.5	1.5	1.3	1.3	1.7	2.5	2.5	1.7	2.5
CPU cores (in numbers)	2	4	2	8	8	8	2	4	2	2
CPU load (%)	92	16	44	89	13	64	60	99	26	53
Effective CPU	0.0478	0.3750	0.0682	0.1169	0.8000	0.2125	0.0833	0.1010	0.1308	0.0943

Table 6.23. Dataset used in the experiment: resource parameters details of SMDs of set 2 (M_2)

Parameter	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}
CPU frequency (GHz)	2.5	2.2	2.2	1.5	2.2
CPU cores (in numbers)	2	2	8	8	8
CPU load (%)	53	26	19	15	19
Effective CPU	0.0943	0.1692	0.9263	0.8000	0.9263

6.3.4.2 Simulation Provisioning

The details of the simulation environment and settings are given in the following. The experimental setup for this experiment also was same as mentioned in Section 6.2.3.2.1.

6.3.4.2.1 Task Initiation

For experimental purposes, we generated tasks of two different sizes. One set ($T_1 = \{t_{1_1}, t_{1_2}, t_{1_3}, \ldots, t_{1_{100}}\}$) comprised of 100 subtasks while the other ($T_2 = \{t_{2_1}, t_{2_2}, t_{2_3}, \ldots, t_{2_{200}}\}$) having 200 subtasks, as shown in Fig. 6.16. Here, $T_1 \neq T_2$, i.e., the instruction lengths of the tasks in the two task sets are dissimilar. The tasks were randomly generated with the instruction length ranging from 5 to 250. The purpose and rationale behind considering these varieties of task sets are discussed in Section 6.3.4.3.

$$T \Rightarrow \begin{cases} T_1 \Rightarrow \{t_{1_1}, t_{1_2}, t_{1_3}, \ldots, t_{1_{100}}\} \\ T_2 \Rightarrow \{t_{2_1}, t_{2_2}, t_{2_3}, \ldots, t_{2_{200}}\} \end{cases}, T_1 \neq T_2$$

Fig. 6.16. Different task sets used in the experiment

6.3.4.2.2 Control Parameters

The details of the control parameters used in the proposed PSO-based algorithm and the GA are listed in Table 6.24. Since MCT, MinMin, MaxMin, and PPIA are

heuristic algorithms, no such specific control parameters are required.

Table 6.24. Control parameters for PSO and GA

Algorithm	Parameter		Values
Proposed-PSO	Population size		10
	Iterations		100
	Initial parameters	C_1, C_2	2.0962
		r_1, r_2	Rand(0,1)
		w	0.0967
	Initial social influence		Rand(0,1)
	Initial personal influence		Rand(0,1)
	v_{max}		0.5
	v_{min}		0.5
	w_{max}		1.0
	w_{min}		0.0
GA	Population size		10
	Iterations		100
	Crossover point		40
	Mutation rate		0.1

6.3.4.3 Performance Analysis

We performed an extensive simulation of the proposed PSO-based energy efficiency scheduling algorithm for performance assessment and analysis on the real dataset described in Section 6.3.4.1. We performed the same experiment with the same assumed scenarios for a comparative assessment using three other popular algorithms. Since our proposed algorithm is a metaheuristic, to have a justified comparative assessment, we deliberately selected a combination of heuristics (MTC, MinMin, MaxMin, and PPIA) and metaheuristic (GA) algorithms that are popularly used in similar problem scenarios.

As we did in Section 6.3.3.4 to demonstrate the illustration, we also frame the performance analysis into two segments. We wanted to compare the performance of the proposed PSO-based algorithm with others in terms of i) scheduling considering only energy efficiency and ii) scheduling considering energy efficiency with load balance.

Further, to assess the performance of the proposed algorithm, we designed diverse task-SMD mapping scenarios. As discussed in Section 6.3.4.2.1, we generated task sets with two different sizes of 100 (T_1) and 200 (T_2) subtasks, $T_1 \neq T_2$. These tasks were scheduled to two SMD sets of 5 (M_1) and 10 (M_2), $M_1 \neq M_2$. To achieve task heterogeneity, we assigned two different task sets to the same SMDs alternatively, i.e., T_1 and T_2 both were mapped first to M_1 and then to M_2. Similarly, for SMD

variability, we assigned a single set of tasks to both sets of SMD, i.e., T_1 was first mapped to both M_1 and M_2 and then T_2 was mapped to both M_1 and M_2. All the four mapping scenarios are shown in Fig. 6.17.

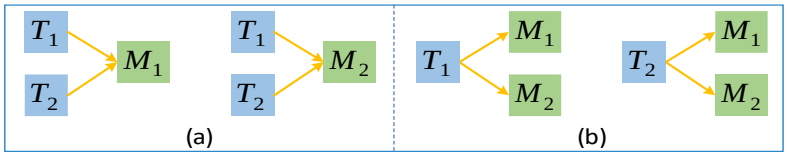
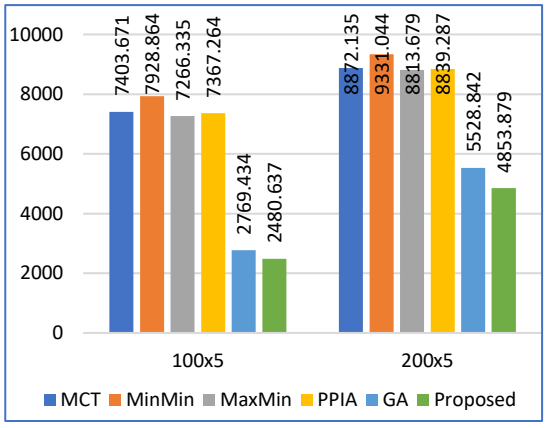


Fig. 6.17. Task sets to SMD mapping scenarios: (a) task heterogeneity and (b) SMD variability

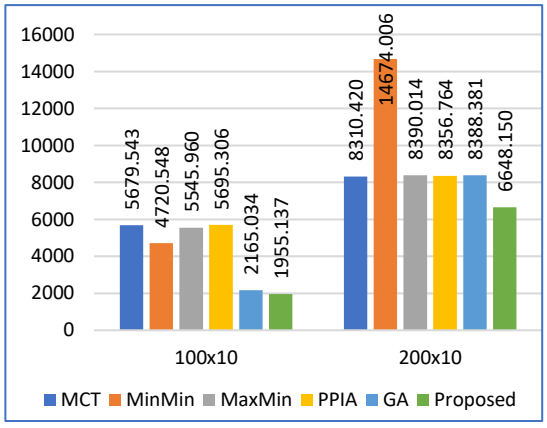
6.3.4.3.1 Energy Efficiency

First, we ran all the algorithms without considering load balance but with four different sets of task-SMD combinations.

Task heterogeneity: Fig. 6.18 shows the energy efficiency achieved by all six algorithms when a) T_1 and T_2 are scheduled to M_1 and b) T_1 and T_2 are scheduled to M_2. In all the combinations, our algorithm performs significantly well compared to the other five algorithms. It is no-brainer to assume that the larger task set would consume more energy. From the figure, it can be seen that our algorithm reflects this variability better than others. It exhibits almost linear variations for both sets, very close to the exact variation.



(a) Task heterogeneity with a smaller set of SMDs

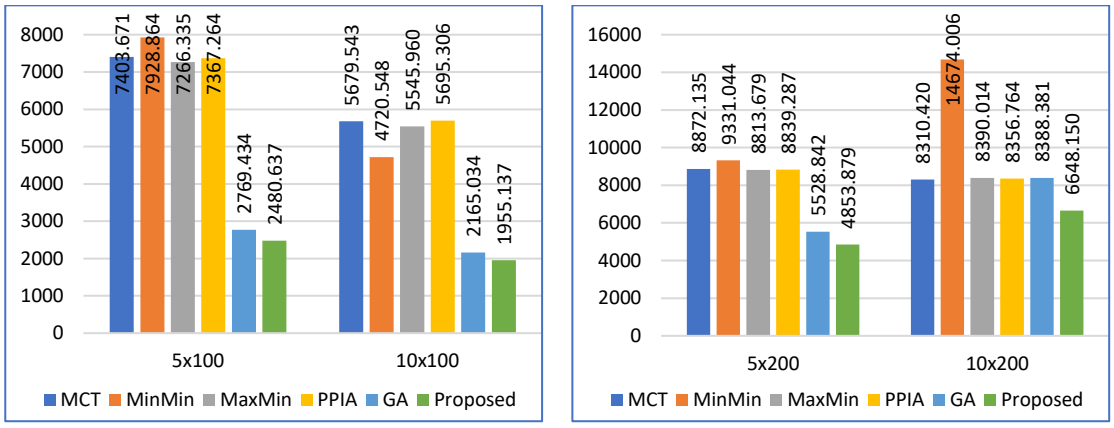


(b) Task heterogeneity with a bigger set of SMDs

Fig. 6.18. Task heterogeneity for energy efficiency without load balance

SMD variability: Fig. 6.19 shows the energy efficiency achieved by all six algorithms when a) T_1 is scheduled to M_1 and M_2 and b) T_2 is scheduled to M_1 and M_2. In both the cases, our algorithm performs far better than others. However, here we witness a contrasting pattern for SMD-task variability. When the number of SMDs

is increased, the energy consumption decreases for T_1 but it increases for T_2. This is true for all other algorithms except MCT, MaxMin and PPIA. For these, the energy consumption decreases marginally when the number of SMDs increases.



(a) SMD variability with a smaller set of tasks (b) SMD variability with a bigger set of tasks

Fig. 6.19. SMD variability for energy efficiency without load balance

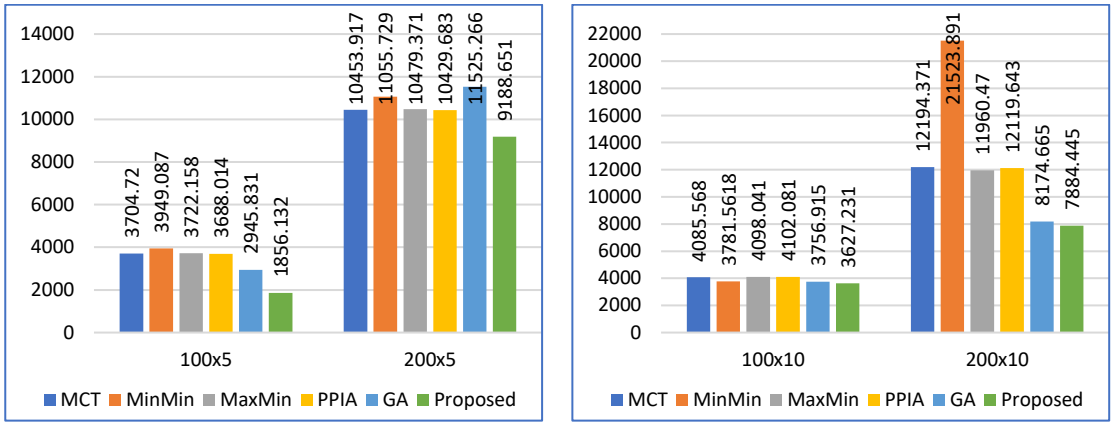
6.3.4.3.2 Energy Efficiency with Load Balance

In the second case, we ran all the algorithms considering load balance for scheduling along with energy efficiency with four different sets of task-SMD combinations.

Task heterogeneity: Fig. 6.18 shows the energy efficiency achieved by all six algorithms while considering load balance when a) T_1 and T_2 are scheduled to M_1 and b) T_1 and T_2 are scheduled to M_2. In this case also for all the combinations, our algorithm performs appreciably better than other algorithms. As usual, the energy consumptions upsurge with a larger task size for the same set of SMDs. From the figure, it can be seen that the energy consumption increased almost more than threefold when the task size gets doubled for five SMDs. For ten SMDs also, this trend continues.

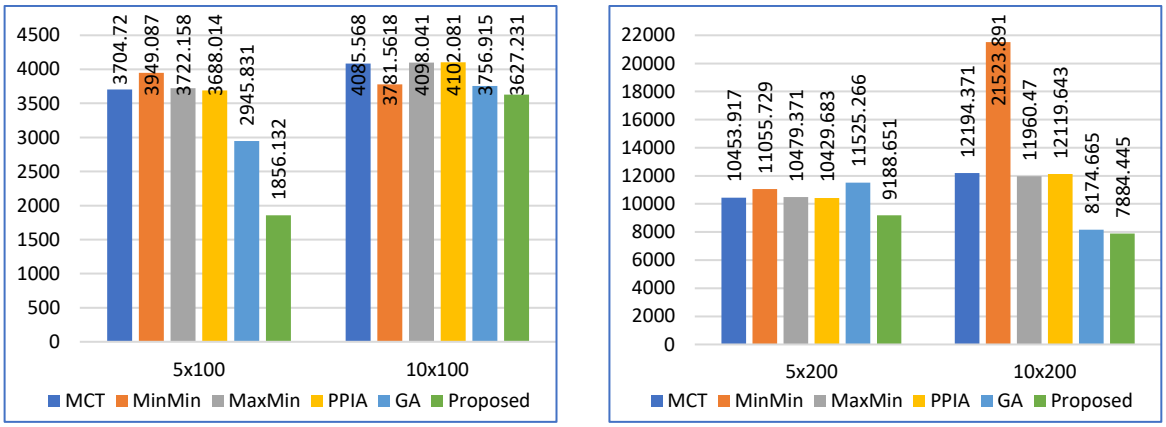
SMD variability: Fig. 6.21 shows the energy efficiency achieved by all six algorithms while considering load balance when a) T_1 is scheduled to M_1 and M_2 and b) T_2 is scheduled to M_1 and M_2. As expected, in this case also our algorithm beats others handsomely. Similar to the case of SMD variability (Fig. 6.19) in Section 6.3.4.3.1, here also we can see a mix of contrasting patterns for SMD-task variability. For GA and our algorithm, when the number of SMDs is increased, the energy consumption increases for T_1 but it decreases for T_2. For MCT, MaxMin and PPIA it

increases for both T_1 and T_2. For MinMin, it is just opposite, i.e., the energy consumption decreases for T_1 and increases for T_2.



(a) Task heterogeneity with a smaller set of SMDs (b) Task heterogeneity with a bigger set of SMDs

Fig. 6.20. Task heterogeneity for energy efficiency with load balance



(a) SMD variability with a smaller set of tasks (b) SMD variability with a bigger set of tasks

Fig. 6.21. SMD variability for energy efficiency with load balance

6.3.4.4 Statistical Analysis

To check the significance of our algorithm in comparison to other considered algorithms, we conducted a one-way ANOVA [694] test on energy efficiency with load balance. As usual, the null hypothesis (H_o) assumes that the mean of all the groups is equal, and based on that, one group can be rejected. Whereas the alternative hypothesis (H_i) assumes means are not equal.

We used ten random task sets where each set consisted of 100 tasks executed on five SMDs. The summary of the input sets of the ANOVA test is presented in Table 6.25. It is observed from the table that the means and variances of the considered algorithms are different. It can be observed that the proposed algorithm has the lowest mean, whereas MinMin has the highest.

The overall statistics of the input set using energy efficiency with load balance are given in Table 6.26. Similar to Section 6.2.3.4.1, here also, the H_o was rejected if the P-value < α, α =0.05, i.e., F-statistic >> F-critical. Therefore, we rejected the null hypothesis at the one percent level.

Table 6.25. Input sets of ANOVA test

Groups	Count	Sum	Mean	Variance
MCT	10	76807.342	7680.734	1476793.675
MinMin	10	90568.831	9056.883	11639390.706
MaxMin	10	76111.989	7611.199	1604243.861
PPIA	10	76723.792	7672.379	1457634.273
GA	10	46001.658	4600.166	5911437.533
PSO	10	39210.122	3921.012	3493183.924

Table 6.26. ANOVA test results

Source of variation	SS	DF	MS	F-statistic	P-value	F-critical
Between groups	204050825.470	5	40810165.094	9.571	0.000	2.386
Within groups	230244155.748	54	4263780.662	-	-	-
Total	434294981.218	59	-	-	-	-

6.3.5 Discussion

In Section 6.3.4.3, we witnessed that our algorithm is well capable of coping with the task heterogeneity. It performs almost linearly compared to other algorithms, reflecting the dynamics of task heterogeneity. However, for SMD variability, it exhibits inconsistent behaviour. In fact, it is true for all the algorithms. None shows a definitive pattern of energy consumption variations with varying numbers of SMDs. One possible explanation for this could be the heterogeneity in the CPU properties of the SMDs.

Nevertheless, our primary goal was to minimise the overall energy consumption for scheduling a set of MCC tasks to a set of SMDs. In this regard, we can claim that our algorithm is successful in both the cases – only energy efficiency and energy efficiency with load balance. Fig. 6.22 shows the average energy consumption of all the algorithms for four experimental alternatives. Our algorithm outperforms all other five algorithms. Our algorithm performs significantly better than others in the first case (only energy efficiency). In the second case (energy efficiency with load balance), the performance difference is smaller than in the first case.

Based on Table 6.25, the mean differences between the proposed algorithm and the other compared algorithms are shown in Fig. 6.23. It can be observed that the

closest performer to our algorithm was the GA. It is probably due to the similarity in the optimisation approach. Both GA and the proposed PSO-based algorithm are metaheuristics, while others are heuristic. Whereas the performance of the MinMin is the farthest from the proposed one i.e., MinMin has the worst energy efficiency. This is because MinMin always tries to minimise the execution time by repeatedly scheduling the tasks to the most potent SMDs. And the energy consumption increases with the power of the SMD CPU.

The energy efficiency achieved by our proposed algorithm not only allows sustainable computing but also significantly impacts the monetary expenses like energy bills to the organisations which adopted MCC as computing infrastructure. Furthermore, the reduction in energy consumption of individual SMDs would ensure lesser battery drainage due to MCC tasks. This would be a great relief for SMD users, stopping them from worrying about charging their SMDs frequently. This will not only lead to minimised users' inhibition in joining MCC but also increase the retention of the SMD providers in MCC, which would, in turn, steer the success of MCC.

6.4 Limitations and Further Scope

Although we fulfilled the desired objectives with the solutions presented in Section 6.2 and Section 6.3, and achieved satisfactory performances for scheduling tasks in an MCC scenario considering various resource parameters, there are a few inadequacies which might be addressed to augment its effectiveness and usability. Below, we mention a few such arguments.

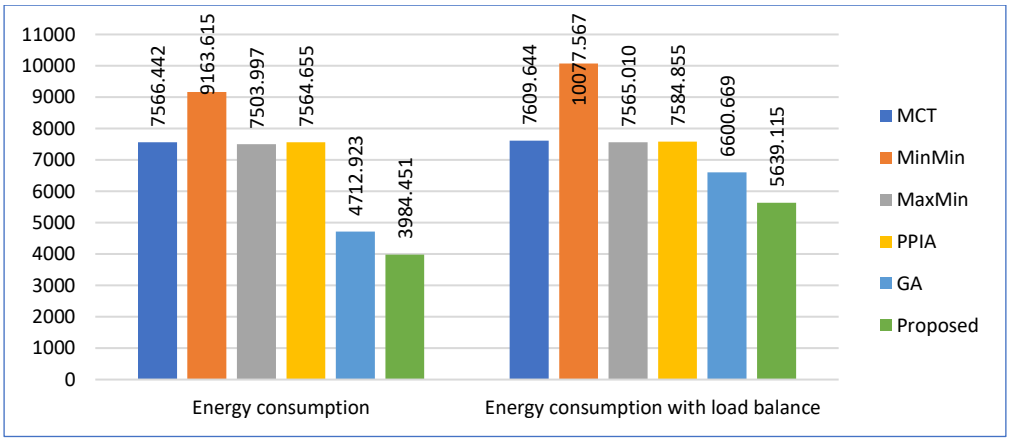


Fig. 6.22. Average energy consumption of all the four case scenarios

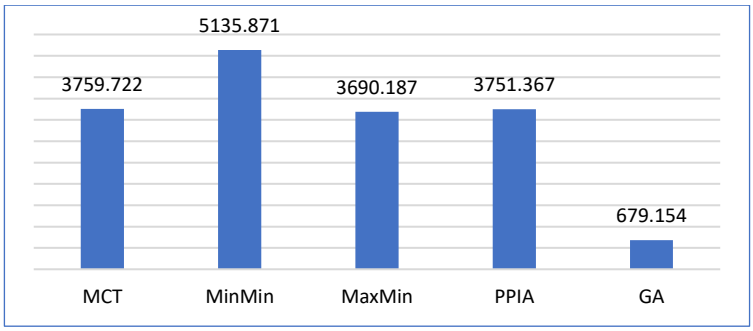


Fig. 6.23. Mean differences between the proposed and other algorithms

Here, we considered a simulated task assignment scenario where the size of each task is exactly known. But for real implementation, pre-calculating the task size accurately is not trivial, and this would affect the estimation of the resource requirement of the assigned tasks. Though this is true for all scheduling problems, it would be interesting to observe the performance of the proposed algorithm in a real MCC implementation.

We assumed that the CPU of the SMDs would be instantaneously available whenever an MCC task was scheduled. But in a practical MCC, the tasks are executed in a cycle stealing fashion, i.e., when the CPU of the allocated SMD is found free, the task is executed. This might affect the overall execution time. Although this should not impact the correctness of the proposed scheduling algorithm, its effectiveness might be affected. However, it would not be generally possible to include CPU cycle availability in the scheduling criteria because it is almost impossible to assess it beforehand. In a local MCC where the SMD users remain more or less the same continuously, and if the usage patterns of the devices can be tracked and analysed, the probable availability of the CPU cycles can be predicted to some extent.

To calculate the resource strength of an SMD, we overlooked the GPU power, though it plays a crucial role in maximising the throughput for highly parallel tasks. However, including the GPU would not change the fundamental behaviour of the algorithm. Hence, it can be said that the proposed algorithm would also work efficiently for GPU-centric tasks.

We assumed that all the tasks were independent and could be executed absolutely in parallel. But generally, when a large task is divided into multiple smaller tasks, dependency constraints are associated among the tasks, meaning not all tasks are

parallelly executable. This makes the scheduling more complex. In future work, we aim to propose a DAG-based task model that would ensure the maximum parallelisation with maintaining the serializability.

In the second case, we ignored the energy consumption accounting for data transfer because we assumed our MCC setup of a single WLAN and the MCC tasks as CPU-intensive. However, for an internetworked MCC and/or an MCC application with mostly I/O bound jobs, the communication energy consumption would be a significant factor. Therefore, for an ideal and universal energy-efficient scheduler for MCC, the data transfer energy should be incorporated.

We did not consider makespan and energy efficiency together as scheduling criteria. It will be challenging to have a balance between these two conflicting objectives. Parity optimality can be used to address this.

6.5 Summary

In this chapter we addressed the scheduling problem in MCC. In the first part of the chapter, we proposed a multicriteria-based resource-aware task scheduling algorithm for MCC. We aimed to develop a scheduling algorithm that would minimise the makespan and maximise the load balance and resource utilisation of an MCC system on the whole. We adopted a heuristic approach to attain the considered optimising criteria to attain the optimisation goal. The SMDs were assessed by their resource strength based on different static (CPU clock frequency and the number of cores, battery capacity) and dynamic (present CPU load, available RAM, available battery, and device temperature) resource parameters. Based on the resource strength and the task length, the tasks were mapped to the suitable SMDs so that the considered objectives were accomplished. We compared our algorithm with three other popular metaheuristics and heuristic algorithms – PSO, GA, and MCT. It was observed that for the considered objectives, our algorithm showed better results in various simulation scenarios. The efficacy of the proposed algorithm was further validated through two statistical measures – ANOVA and post hoc.

In the second part of the chapter, we proposed a PSO-based scheduling algorithm

to minimise the overall energy consumption of the SMDs in executing an MCC task while maintaining a reasonable load balance among the SMDs. We considered two different experimental scenarios – first, targeting only energy efficiency and second, targeting energy efficiency with load balance. Further, we diversified the analysis by bringing variability in the number of tasks and SMDs. For each category, we checked each SMD's energy consumption and the overall energy efficiency achieved to execute all the scheduled tasks. We compared the performance of the proposed algorithm with several well-known heuristic and metaheuristic algorithms on various parameters. With respect to task heterogeneity, our algorithm exhibited almost linear variations of increased energy consumption with increased task size. But the experiment on SMD variability was inconclusive. Nevertheless, our algorithm achieved the most energy efficiency for scheduling in MCC with or without load balance consideration.

7
Resource Availability Prediction in Local MCC

"Your ability will not help if you do not give your availability." --- Saji Ijiyemi

7.1 Introduction

Due to the user's mobility, the SMDs in a particular network may not be available continuously over time. However, they might be available for several discrete periods. Due to this instability, there is always a high probability that a crowdworker leaves the network without finishing the assigned job. There are two possible solutions when a crowdworker departs before completing the assigned task:

i. The job is restarted from the beginning by another crowdworker. This delays the task execution as the whole process is to be started again, including resource (crowdworker) selection and job assignment.
ii. Savepoints are maintained periodically. When a crowdworker departs, the task is rollbacked to the last savepoint and resumed from there by another crowdworker. This approach might be better than starting the job from the beginning, but keeping savepoints is overhead, and also determining the period length after which the savepoints are noted is a decision challenge.

In both cases, the QoS of MCC is compromised, which ultimately affects the successful realization of MCC. That is why assessing the availability of the assignee crowdworker before assigning a task to it is so crucial in MCC.

To address this issue, we suggest, before submitting a task to a crowdworker, it is to be assured that it would not leave until the job is finished. But, for mobile devices, guaranteeing availability is not straightforward. One approach, as assumed in [156], is that every crowdworker should announce their departure time immediately after entering the MCC network. Based on the declared departure time, suitable jobs would be assigned so that they could be finished timely. But it has two issues - i) it is not very practical and ii) there is no guarantee that the crowdworker will keep its word that it would not leave before the declared time. Due to several

reasons, a crowdworker may leave the network unscheduled even if it is priorly agreed to be there for a specified period.

Another possible option, as suggested in [140], is that if a crowdworker wants to leave the network, it will notify the coordinator. This would solve the first issue mentioned above, but not the second completely. The overhead of job handover remains. Also, some dishonest devices may not follow the rule and leave abruptly without notifying.

Owing to these drawbacks, we propose an availability-aware SMD selection scheme for a local MCC. We predict the availability of an SMD for a minimum duration of the task length (execution period), based on which the resource selection decision can be made. For this, we tracked the in-time and out-time of the SMDs on the previous occasions when they were connected to the considered Wi-Fi AP. Based on this historical mobility/availability information, the probable availability till a particular duration of an SMD at any given point of time is assessed. This problem can be represented as time-series analysis.

Considering this, we propose a specialized convolutional feature extraction method to enhance the performance of the LSTM and GRU models. The presented approach is most useful for a local MCC environment where the SMD owners visit and join the MCC network on a regular basis. In such a scenario, the main objective of this chapter is to improve the QoS and reliability of the MCC by minimizing the handoff or job offloading and reassignment. The success of this approach will depend on the accuracy of the availability prediction of the considered SMDs.

In particular, in this chapter, we aim to achieve the followings:

- Design a logging model for recording the in-time and out-time of SMDs in consideration with respect to a Wi-Fi AP.
- Propose a novel dynamic feature extraction process suitable for the datasets where the features are unknown.
- Designed a novel method for representing time-series data into a vector to perform the convolutional feature extraction.
- Combine the proposed convolutional feature extraction with both LSTM and

GRU for prediction.

- Compare the prediction performances with each other as well as with the basic LSTM and GRU, and also with ARIMA.

7.2 Solution Approaches and the Proposed Solution

Time-series analysis is exercised on the set of observations where each record is observed at a specific time. Analysis of these types of data is not like other statistical modelling and inference due to its apparent correlation in adjoining records introduced by the sampling time. These features limit the applicability of many statistical models that assume the observations are independent and identically distributed. The ARIMA model has been a widely used linear model for forecasting time-series data, and it has been a standard for a long time. ARIMA considers lag value determined by the correlation among the continuous values, the dependency between an observation and the residual, and seasonality (if it exists) for building the model and result in close prediction of the future [699]. ARIMA models have the advantages of being more flexible compared to other statistical models and have a better performance for a longer sequence of data with a stable correlation between past observations. But ARIMA model can only capture the linear pattern of the data, but not the hidden patterns that are stochastic and non-linear in nature [700] [701]. Furthermore, an ARIMA model assumes a constant standard deviation in errors, which may not be true in practice. Like most of the real data, the dataset considered in this work is also non-linear in nature.

Another statistical model, the Markov chain process, has been popularly used for time-series prediction, especially where clarifying the interrelationships of the model is important [702] [703]. Though they provide significant accuracy for short-term prediction, they have a probability of presenting miss prediction for long-term sequences, which is necessary for crowdworker selection. For instance, it may happen that a particular user was not available for certain days in the previous months. Due to the lack of memory in the Markov model (and also ARIMA), it will not be able to hold the long-term historical information; hence it cannot incorporate such irregularities in the model. This leads to inaccurate predictions.

In summary, the classical time-series analysis models suffer from the following limitations:

- They are sensitive towards missing values.
- They require special transformations to convert the data into a linear form.
- Most of them support only univariate data; they do not support multiple independent variables to be taken as inputs.
- Though they have satisfactory prediction performance in short-term predictions they miserably fail in long-term predictions.

Furthermore, the traditional prediction models work better when the data follow a statistical distribution. In our problem, we wanted to capture the real and consistent behavior of the user, which required considering a long-term data relationship. In the user mobility data, there is very little chance of finding a perfect fit of a distribution, which is a basic requirement for the traditional prediction tools. To tackle this, a time-series analysis model is needed that can make a prediction on the dataset without fitting a known distribution.

Considering the above-mentioned issues, machine learning techniques are intensely studied for being used in time-series forecasting. Machine learning models are good to exercise the non-linear patterns. K-nearest neighbor, decision tree, support vector machine, etc., can be used to model time-series data when the observation consists of a non-linear pattern.

Capturing the changes of the contexts in time-series data is an important criterion for a prediction model. However, traditional machine learning based prediction models cannot capture these changes appropriately. Hence, they are unable to provide satisfactory prediction accuracy in cases where the contexts of the considered data change frequently. In the user mobility data considered in this work, we needed to identify the users' behavior for a longer period. To provide expected prediction results, the prediction models need to capture these changes appropriately.

Deep learning based model like RNN (recurrent neural network) is capable of retaining long-term contextual information due to the presence of specialized memory. It uses a looping constraint that helps to capture the sequential

information in the data. Deep learning models also have the inherent capability of searching relations in the dataset without prior knowledge of any distribution.

Generally, RNNs are used for time-series data prediction. But traditional RNNs suffer from vanishing gradient problems when the input length is too high [704]. So, to overcome the issue, upgradations of RNNs like LSTM (long-short term memory) [705] [706] and GRU [707] [708] were developed. Both methods have been popularly used for sequence modeling and time-series predictions [709].

However, a prediction model is not sufficient to attain a handsome or the expected prediction accuracy since these models work on the available input features without any feature extraction. For that, it is required to apply a proper feature extraction mechanism to the dataset. The feature extraction is a crucial aspect of designing a prediction model because it captures the most relevant features from the data, which generally improves the model performance.

The existing and known feature extraction methods, in most cases, can extract the exact features required for a particular prediction problem. But they may be incompetent when we do not have a clear idea of the most dominant features. Therefore, we needed to frame a feature extraction methodology that would be able to dynamically extract the features for solving the proposed resource availability prediction problem.

Recently, CNNs (convolutional neural networks), the idea of which was first presented by Fukushima in 1980 [710] and later improved by LeCun *et al.* [711] [712], have become popular for extracting dynamic features for prediction-based models. A CNN is a special kind of neural network for processing 2-D image data [713] [714] [715]. CNNs are very effective in extracting and learning features not only from one-dimensional sequential data, such as univariate time-series data, but also from multivariate time-series data [716] [300].

Owing to their distinct architecture, the LSTM layers in an LSTM model can capture the sequence pattern information quite efficiently. As the LSTM networks are designed to deal with temporal correlations, they utilize only the features provided with the training set [295]. The convolutional layers of CNNs can extract more

valuable features by filtering out the noise prevailing in the raw input data [306]. They are also capable of scooping the hidden features that otherwise could not be pulled out by using only LSTM. This is the core motivation to exercise a convolutional feature extraction layer in addition to the basic LSTM for our presented availability prediction problem, so that we could exploit the benefits of both techniques to achieve a better prediction performance. The combination of CNN and LSTM is useful for learning features of not only short-term time variation but also long-term dependency periodicity [290].

7.3 System Model and Hypothesis

We considered a local environment (campus) where the SMDs get connected to a WLAN through a Wi-Fi AP. The owners of the SMDs may come within the range of the network and get connected more than once a day. Most users (with their SMDs) are often available for a certain duration. For example, in a classroom and a workplace, the students and the workers are regularly available for a specific duration in consistent intervals. If they take public transport for commuting to reach their institute and workplaces, in most probability, they would be available for the duration from boarding point to the destination. Similarly, some people spend a certain amount of time in the library regularly while some go to the same coffee shop or canteen regularly. In all these cases, the availability of the users can be predicted by analysing their presence history.

The accuracy of the availability prediction depends on the campus type. For example, in a classroom or a typical office, the availability is somewhat predetermined. In comparison, the predictability in a coffee shop (where certain customers come regularly) varies according to its location and its services. Likewise, in public transport (regularly used by a group of commuters), the availability is very much fixed (usually one drops at his stop regularly). The crowdworker predictability gradient based on the availability is shown in Fig. 7.1.

To model the working of a local MCC, we have assumed the following:

- We considered a general task execution model where the SMDs receive some compute-intensive tasks either individually or in batches.

- Each task has its own computation requirements, input and output data size, and finite execution time. We assume that these parameters are known.
- Each crowdworker avails a fixed and equal bandwidth.
- Each crowdworker completes the assigned subtask within a finite time and sends back the results before leaving the MCC network.
- A crowdworker would share its resources until it is present in the network.
- All SMDs, which have the MCC client installed, are considered as crowdworker and willing to share their resources, either on a profit or non-profit basis.
- The SMDs in MCC are uniquely identified by the UIDs.

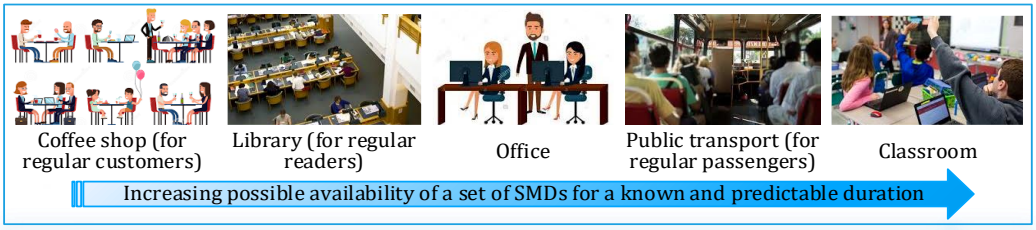


Fig. 7.1. Predictability gradient of crowdworkers' availability in a local MCC

To model our proposed availability prediction, we further assume the following considerations:

- In need of job submission, the coordinator looks for the most appropriate crowdworker(s).
- The MCC coordinator already has a list of suitable crowdworkers (based on some pre-set criteria).
- The coordinator decides to pick the top-ranked crowdworker(s) from the list (as discussed in Chapter 4 and 5).
- Just before submitting the job to the selected crowdworker, the coordinator wants to be sure of the probability of the crowdworker being available until the job is finished.
- If, as per prediction, the SMD is not supposed to be available, the next SMD in the list is considered, and again the availability of this SMD is checked. This continues until the suitable SMD of which the availability period is greater than the task execution time is found.

In this chapter, we address the last point, i.e., before the job is actually be submitted, the stability of the selected crowdworker is to be assessed for the duration of execution of the assigned job. If the crowdworker's presence time is greater than the task size (estimated execution time), then only it is finally considered for the job assignment. The workflow diagram of the whole process is depicted in Fig. 7.2.

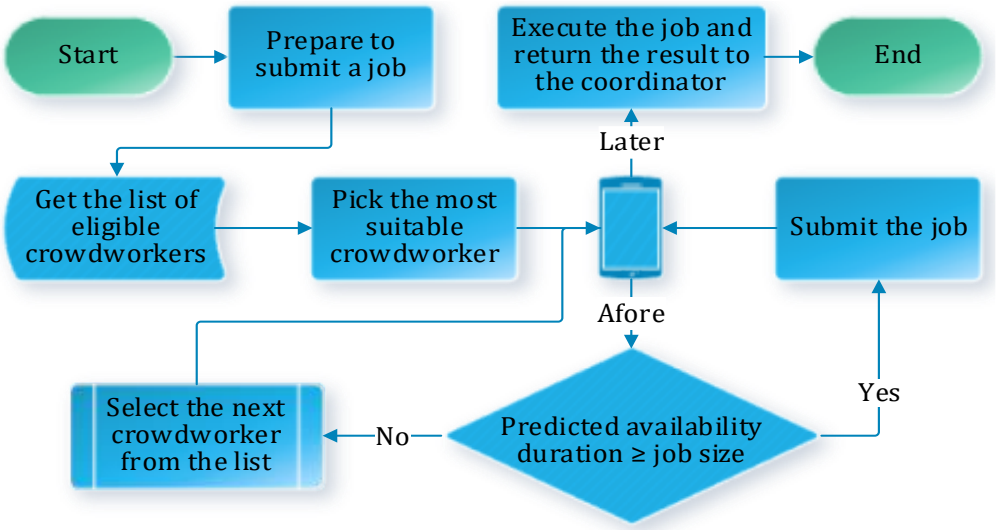


Fig. 7.2. Workflow diagram of crowdworker selection based on availability

7.4 Resource Availability Prediction in MCC

In this section, we present the comprehensive details of the proposed resource availability prediction method along with the experimental results and discussion.

7.4.1 Problem Definition

Let T_j be the job size (execution time) of a job J and M be the preferred crowdworker for J. At the time of job submission (t_j), we need to know how long M might be available after t_j let this be M_a. J should be submitted to M if and only if Eq. 7.1 is satisfied, where k is some constant.

$$M_a \geq T_j + k \tag{7.1}$$

The details of the M_a calculation and crowdworker selection criteria are discussed in the next subsection.

7.4.2 Problem Designing

We considered the resource selection process as the combination of two processes: a) resource availability prediction and b) resource selection. The resource selection

depends on the outcome of resource availability prediction. However, our primary focus is on resource availability prediction. Major components/modules of the crowdworker selection procedure are as follows:

Calculate completion time: The job completion time is an approximate higher bound value of the time required for the particular job to complete. This function needs two parameters, namely, job size (T_j) and time of job submission (t_j). The completion time (T_j^c) of the job J is defined by Eq. 7.2. We assumed that for each J, T_j is the same for all crowdworkers.

$$T_j^c = T_j + t_j \tag{7.2}$$

Get selected device: As depicted in Fig. 7.2, the top-ranked SMD in the crowdworker's list for the reckoned job would be considered.

Get session history: According to the UID from the 'get selected device' module, the history of the SMD is extracted from the log. The details of data collection are discussed in Section 7.7.1.

Predict out-time: This module takes the session history of the device's previous session durations to predict the expected session duration in the current time using CLSTM (convolutional LSTM). The current session in-time (S_i) is added with the forecasted duration (P_M) to get the predicted out-time (S_o) of the device in the current session, as shown in Eq. 7.3. The predicted availability duration (M_a) is calculated by Eq. 7.4.

$$S_o = S_i + P_M \tag{7.3}$$

$$M_a = t_j + S_o \tag{7.4}$$

So, Eq. 7.1 can be rewritten as Eq. 7.5, where $k_1$is the runtime of the prediction algorithm and k_2 is the padding time between decision making and job dispatching.

$$M_a \geq T_j^c + k_1 + k_2 \tag{7.5}$$

Crowdworker selection: This function checks for the availability of the SMD for the specified duration and returns a Boolean for selection. The SMD would be

selected if Eq. 7.5 is satisfied.

Fig 7.3 depicts the combined workflow of the above-mentioned modules, whereas Fig. 7.4 shows the important steps followed towards SMD selection.

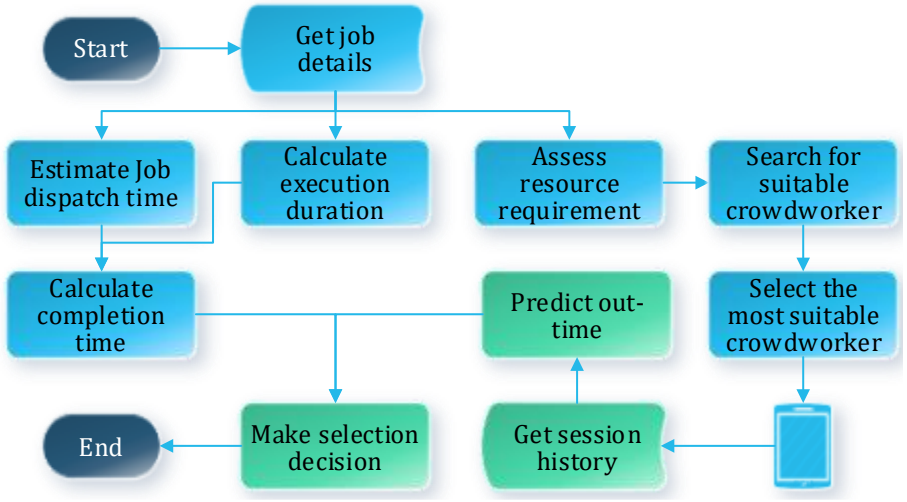


Fig 7.3. Availability prediction process of an SMD in MCC

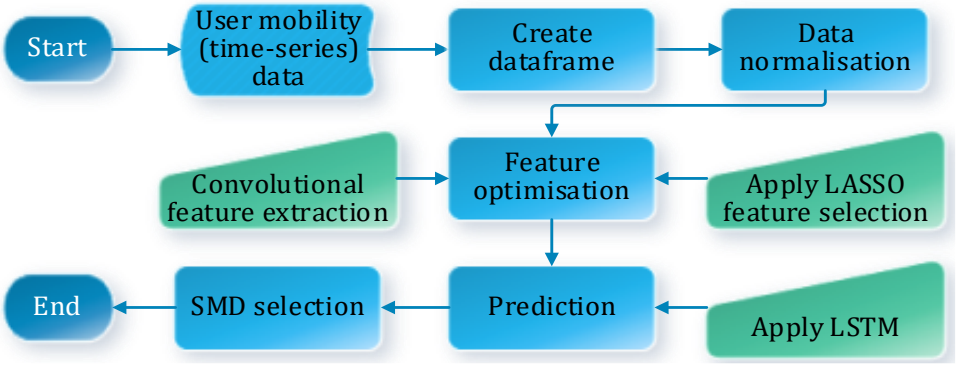


Fig. 7.4. Important steps for SMD selection

As the experiment, we followed the below-mentioned steps:

a. Based on the historical data, a deep learning based prediction model is presented to predict each SMD's stability within a particular network, i.e., Wi-Fi AP, at any given point of time.
b. The session out-time of an SMD is predicted for a given session in-time.
c. A random job submission system is simulated to evaluate the performance of the prediction model using various metrics.

7.5 LSTM and GRU Architectures

In this section, we discuss the basic architectures of LSTM and GRU.

7.5.1 LSTM

An LSTM cell is a special variant of an RNN cell that can handle information of a more extended sequence of information, which can help make the prediction more accurate for longer sequences. Each cell is capable of barring information from flowing through or allowing it to flow through without any change. Allowing the information without change enables LSTM to remember the information from the previous timesteps. Through the LSTM cell sequence chain, there are several inputs and outputs which allow adding or removing information to the cell state. Adding or eliminating information to a cell is done through gates. The gates are the neural networks used to regulate the information flow through the sequence chain of LSTM cells. These gates or the sigmoid layers turn all output values in a value between 0 and 1, where 0 indicates nothing of the component should pass through, and 1 is for the opposite, i.e., everything would be through. The three gates that control the cell states of an LSTM are briefed below, and a typical LSTM block is shown in Fig. 7.5.

Forget gate: This gate gets rid of the information we want to remove from the cell state. The forget gate (f_t) is defined by Eq. 7.6.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{7.6}$$

where, σ is the sigmoid gate activation function, W_f and U_f are the weight matrices for mapping the current input layer and previous output layer into the forget gate, h_{t-1} is the output from the previous cell, x_t is the input layer, and b_f is the bias vector for the forget gate calculation.

Input gate: The input gate (i_t) controls how much information from the current input layer (x_t) pass to the current input cell state ($\check{c}_t$). This gate, defined by Eq. 7.7, gives the outputs between 0 and 1 and decides which values to update. The candidate values which are to be used to update the cell state are calculated by a tanh layer, as shown in Eq. 7.8. The input gate combined with the current cell state updates the current output cell state (c_t), as defined by Eq. 7.9.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{7.7}$$

$$\check{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{7.8}$$

$$c_t = f_t \times c_{t-1} + i_t \times \check{c}_t \tag{7.9}$$

where, W_i and U_i are the weight matrices for mapping the current input layer and previous output layer into the input gate, W_c and U_c are the weight matrices for mapping the current input layer and previous output layer into the current input cell state, b_i and b_c are the bias vectors for the input gate and input cell state calculation, and *tanh*, a hyperbolic tangent function, is the activation function for current input cell state.

Output gate: The output gate (o_t) controls the amount of information passed from the current cell state to the current output cell state. To get the filtered output, the current cell state is passed through a sigmoid layer, as shown in Eq. 7.10, which decides what parts of the cell state would be considered as output. The final output (h_t) is derived, as shown in Eq. 7.11, by multiplying the output of the sigmoid gate, with the output cell state that is passed through a tanh layer (to squeeze the values between -1 and 1).

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{7.10}$$

$$h_t = o_t \times \tanh(c_t) \tag{7.11}$$

where, W_o is the weight matrix for mapping the current input layer into the output gate, b_o is the bias vector for the output gate calculation, and c_{t-1} is the previous output cell state.

7.5.2 GRU

Unlike LSTM, a GRU unit does not have any output gate; rather, it has only two gates – a) update gate and b) forget gate. These two gates are trained to retain information from the past without losing it through time and eradicate the irrelevant information that are not needed for prediction. A typical GRU architecture is shown in Fig. 7.6, while its components are briefed below.

- **Update gate:** The update gate helps a GRU model determine how much of the past information from the previous blocks need to be forwarded to the next block. This allows the model to decide to copy all the past information and eliminate the vanishing gradient problem. The update gate z_t for timestep t is

calculated using Eq. 7.12. Here, x_t is the input at step t, and h_{t-1} is the hidden state that holds the information for the previous t-1 units. W_z and U_z are the respective weights of x_t and h_{t-1}. The sigmoid activation function (σ) helps to keep the value of z_t between 0 and 1.

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{7.12}$$

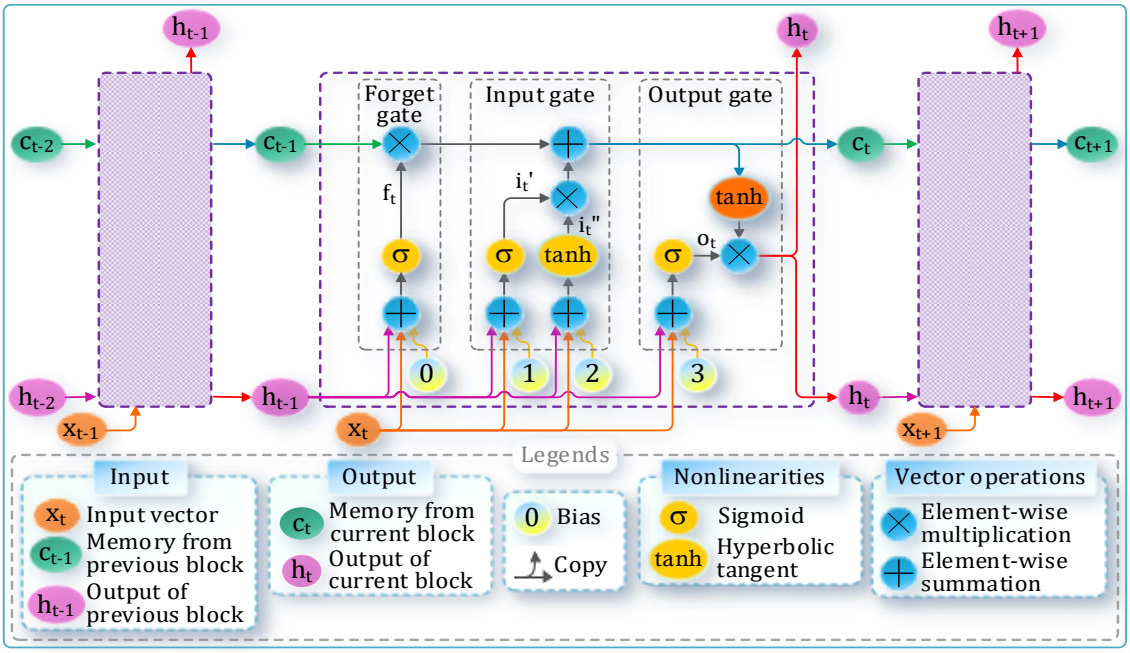


Fig. 7.5. A typical LSTM block

- **Reset gate:** The reset gate r_t is used to decide how much of the past information to forget. r_t is calculated using Eq. 7.13.

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{7.13}$$

- **Current memory content:** The current memory content h'_t uses the reset gate to store the relevant information from the past and is calculated using Eq. 7.14, where ʘ denotes Hadamard (elementwise) product.

$$h'_t = \tanh\,(W x_t + r_t \odot U h_{t-1}) \tag{7.14}$$

- **Final memory:** The final memory h_t at current timestep t is a vector that holds information for the current block and is passed to the next. h_t is calculated using Eq. 7.15. The update gate determines how much information to be retained from current (h'_t) and previous (h_{t-1}) memory contents.

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \tag{7.15}$$

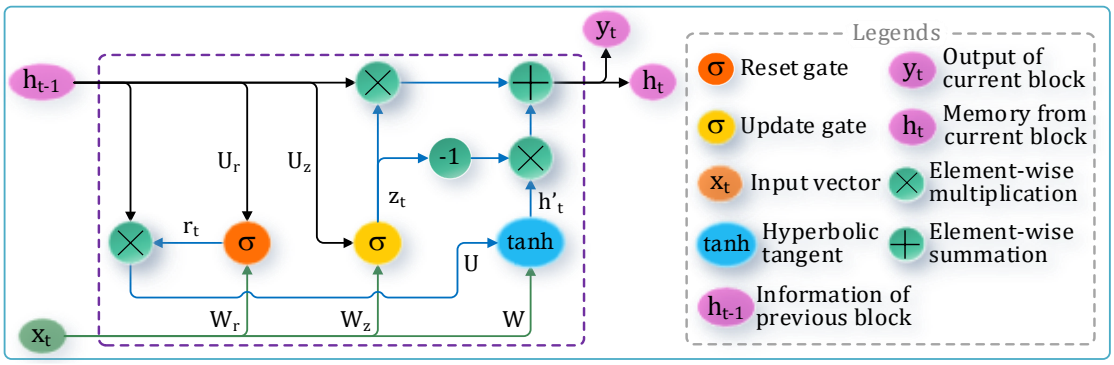


Fig. 7.6. A typical GRU block

7.6 Performance Measurement Metrics

In this section we brief the performance measurement metrics used in the experiment.

Accuracy: It is the measurement of a prediction model's performance based on the total number of correct predictions made and defined by Eq. 7.16. Higher accuracy indicates better efficacy of the prediction model.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{7.16}$$

where, TP: true positive, TN: true negative, FP: true positive, FN: false negative, and N: total no. of predictions.

Precision: It states the proportion of positive predictions that was actually correct and is defined by Eq. 7.17.

$$PRC = \frac{TP}{TP+FP} \tag{7.17}$$

Recall: States the proportion of actual positives that were predicted correctly and is defined by Eq. 7.18.

$$RCL = \frac{TP}{TP+FN} \tag{7.18}$$

F1-Score: It is calculated as the harmonic mean of Precision and Recall, and sometimes provides a better understanding of the model performance than accuracy. F1-score is defined by Eq. 7.19.

$$F1 = \frac{2TP}{2TP+FP+FN} \tag{7.19}$$

Perplexity: It measures how well a probability model predicts an output and is

often used to compare probabilistic prediction models [717]. Although perplexity is popularly used in NLP (natural language processing) [718], here, we used it to represent the prediction loss for an input sample in the current timestep. A lower perplexity signifies a better prediction.

Error: It is the forecast error, which can be defined as the complement of accuracy, as given in Eq. 7.20.

$$1 - \frac{TP+TN}{TP+TN+FP+FN} \tag{7.20}$$

Mean Absolute Error (MAE): An error is defined as the difference between the actual out-time and the predicted out-time of each SMD. MAE is calculated as the average of the prediction errors where all the error values are forced to be positive, as given in Eq. 7.21.

$$MAE = \frac{1}{m}\sum_{i=1}^{m} |(p_i - a_i)| \tag{7.21}$$

where, m is the number of samples, p_i is predicted out-time, and a_i is the actual out-time.

Since an MAE of zero indicates no error, a lower MAE value is always desirable for a good prediction.

Mean Squared Error (MSE): The forecast error is squared to make it squared error and taking the mean gives us MSE. MSE is defined by Eq. 7.22. This score provides worse results to those models that make largely erroneous forecasts.

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(p_i - a_i)^2 \tag{7.22}$$

Root Mean Squared Error (RMSE): It measures the root of the average of the errors' squares, as shown in Eq. 7.23. Here, the deviations are squared to prevent the positive and negative deviations from nullifying each other. It is the most important error measure for a model's performance if the main purpose of the model is prediction. Since RMSE tends to exaggerate large errors, it might be insightful when comparing different prediction methods. Also, it is easier to interpret because the RMSE values are in the same units as the samples. A lower RMSE value suggests a better prediction, while a zero value means no error in prediction.

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(p_i - a_i)^2} \tag{7.23}$$

R-squared (R^2): In comparison to RMSE, which is an absolute measure of correct prediction, R-squared is a relative measure of it. R^2 is defined by Eq. 7.24. The value of R^2 ranges between 0 to 1, while a higher R^2 generally suggests a better prediction performance.

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(a_i - p_i)^2}{\sum_{i=1}^{m}(a_i - \frac{1}{m}\sum_{j=1}^{m} a_j)^2} \tag{7.24}$$

Forecast Bias (FB): It is the mean of the Residual Forecast Error (RFE) metric and is defined by Eq. 7.25. The RFE can be calculated as the difference in the expected value and the predicted value. Minimization of this value should be the objective in order to achieve the best forecasting.

$$FB = \frac{\sum_{i=1}^{m}(a_i - p_i)}{\sum_{i=1}^{m}(p_i)} \tag{7.25}$$

7.7 Data Collection and Selection

In this section, we present the detailed methodologies followed for collecting the real data, selecting the usable data, and making them fit to be fed into the prediction model.

7.7.1 Data Collection

For the experimental purpose, we considered a computer laboratory scenario in an educational institute. To generate the experimental data (i.e., SMD's presence time and duration), the digital simulation could have been opted, but it might not have the uncertainties that are associated with the user presence pattern. Without the uncertainties, the simulation may not behave like the real case scenario. Introducing uncertainties artificially in the simulation may not be feasible as it might break the co-similarities among the data points.

Therefore, we counted on user data traces from a real network with respect to a particular Wi-Fi AP covering a mid-size hall. For every entry and exit time of a user was logged. The data collection details are discussed in Chapter 4. Since we wanted to consider only SMDs, we avoided logging for other connected devices than SMDs

(e.g., PCs and laptops). This enabled us to collect more data efficiently and validate the prediction algorithm using real availability data. The description of the required data particularly for this experiment is shown in Fig. 7.7.

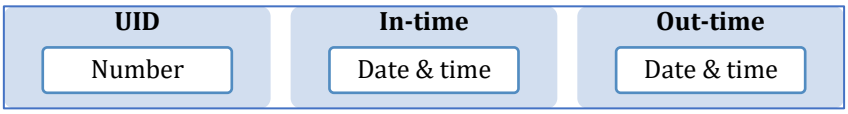


Fig. 7.7. The database schema for SMD availability logging

7.7.2 *Data Selection*

We collected user data for nearly eight months. From the complete dataset, we selected data of 150 days, considering the data quality after applying the normal distribution over the dataset. Out of total days, the maximum concentration of the connected devices was on these 150 days (T_d). However, we also wanted to check the performance of the prediction model when the collected data are less. For this, we had a dataset of 120 days, which is a subset of the 150 days data. Opting for two datasets with a difference of a month of data would allow us to evaluate the applicability of the prediction model in different crowdsourcing applications. Further, out of all recorded users, we considered 50 users for whom there was high presence frequency and less sparsity.

7.8 Prediction Using ConvLSTM2D

For resource availability prediction, we begin with a straightforward and easier approach, i.e., to use a readymade framework. For this, we used ConvLSTM API in Keras [719] that utilizes a convolutional layer for feature extraction and LSTM for prediction. The convolutional layer is directly built into each of the present LSTM units. The stages of SMDs' availability prediction using ConvLSTM2D and thereafter selection are shown in Fig. 7.8. As a prototype experiment, for the particular SMD availability prediction method (i.e., using ConvLSTM2D), we simplified the necessary steps in comparison to the method discussed in Section 7.9.

7.8.1 *Data Preprocessing*

We performed the following operations as data preprocessing:

Data filtering: Data filtering is the process of choosing a smaller segment of the

data set using some threshold parameters such that the remaining data is more suitable for the task. Filtering is mostly used to remove sparse data and redundant data from the dataset. This filtered subset of the data is used for viewing or analysis. Out of all the recorded users, we selected the top 50 users with the highest number of login activities, and the rest of the users were excluded from the test. This filtering ensured that only the active users were considered as a part of the prediction and not some random user who probably logged in only a few times.

Data sampling: Data sampling is the process of selecting the samples of the data from the entire population, which might be useful in introducing variability in the dataset. We used random sampling, a type of sampling technique in which each sample has an equal probability of being chosen. A sample chosen randomly is meant to be an unbiased representation of the total population. Using the random sampling method, the dataset has been divided into ten chunks (D_1 to D_10) of test and train sets with an 80:20 ratio for training and testing, respectively.

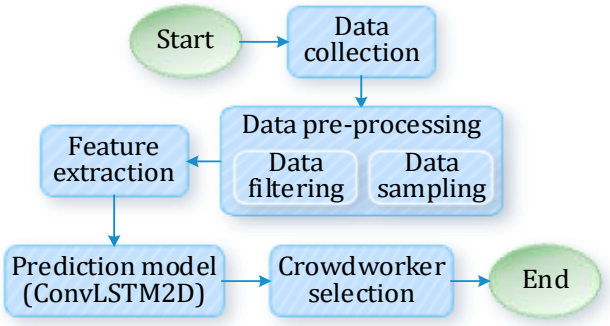


Fig. 7.8. Process flow of the SMD availability prediction model using ConvLSTM2D

7.8.2 Feature Extraction

In the dataset, all the input variables are of numeric type and three important methodologies used are scaling, transforming distribution, and identification and removal of linear dependencies between the input variables. MinMaxScaler has been used to scale the data and normalize them. We used Quantile Transformer for the transformation of distribution, and PCA is performed for removal of linear dependencies.

7.8.3 Prediction Model

The input to our prediction model is the UID of the considered SMD, its past

session duration log, and the expected job completion time. For feeding the data into ConvLSTM model, the data samples are split into subsequences, where the time steps are the count of the subsequence. The number of steps for 50 total users is selected as six for every 2016 data points, and a total of 336 steps for each user. The model expects the input data shape as a 2-D image; hence the input data are reshaped accordingly. Since the session duration is the only feature, the number of features is set to 1. The Keras Python API provides a predefined implementation of the ConvLSTM module. The ConvLSTM2D accepts four parameters (filters, kernel_size, activation, and input_shape) to make the data flatten. The selected parameter values for the experiment are shown in Table 7.1. The steps in designing the ConvLSTM prediction modeling are as follows:

1. The sequential Keras model is selected, and the ConvLSTM2D layer is added using the parameters specified in Table 7.1.
2. The model output is flattened (converted to a single dimension) in order to work with LSTM.
3. A single dense layer is introduced as the last output layer with one node.
4. The model is compiled using the default Adam optimizer [720], and the loss is calculated using the MSE loss function.

7.8.4 Test Result and Analysis

We tested the prediction model by assigning tasks with various task sizes randomly to the simulator. We recorded the number of times when a crowdworker departs before completing the assigned task. The observed performance of the model and the evaluation metrics are discussed below.

Table 7.1. Parameters used for ConvLSTM2D

Parameters	Value	Description
filters	64	64 filters are used for capturing the localized data points.
kernel_size	(1,2)	The row number is always selected as 1 due to the univariate nature of the data.
activation	relu	Rectified Linear Unit (ReLU) is used as an activation function; this helps in introducing non-linearity in the dataset.
input_shape	(2,1,3,1)	n_seq = 2, 1

7.8.4.1 Training and Testing Performance

Accuracy is used to measure the performance of the prediction model based on the

total number of correct predictions made. Higher the accuracy, the better the model. On the other hand, we aimed to minimize the loss, which is the summation of training and testing errors in each epoch. To evaluate our proposed model's performance, we have considered using these metrics over 10 epochs as per the experiment. The accuracy vs. loss graphs for training and testing are shown in Fig. 7.9, respectively. Results show that the training accuracy the model is able to achieve over 10 epochs is 82%, and the testing accuracy is 78%. The differences in accuracy and loss between training and testing are 4% and 7%, respectively. This indicates that the model is neither overfitting nor underfitting.

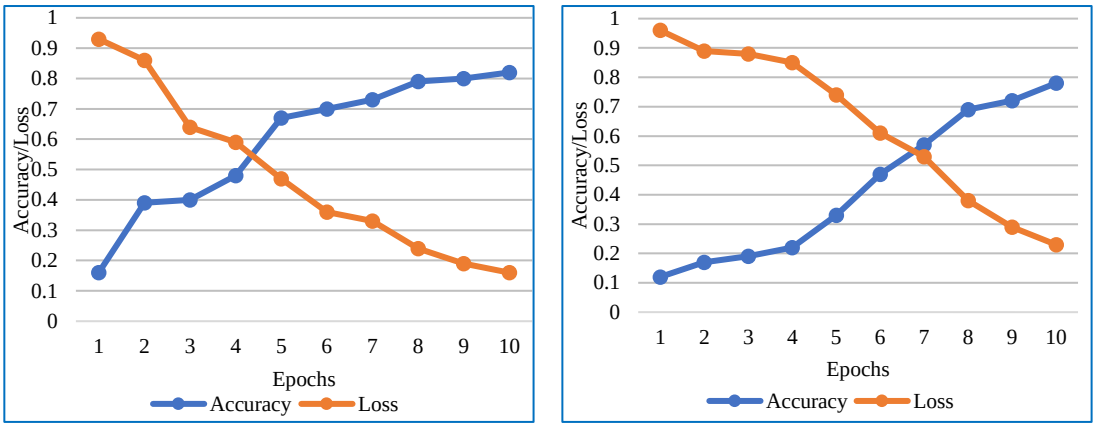


Fig. 7.9. Accuracy vs. Loss for a) training and b) testing

7.8.4.2 Model Evaluation

The results obtained from the experiment are tabulated in Table 7.2. The average accuracy for the considered ten sampled datasets is 78.43%, and the average prediction error is approximately 21%, which makes the prediction model consistent. The forecasting error estimations are shown in Fig. 7.10.

Table 7.2. Evaluation metrics for ten sampled datasets

Dataset	Precision	Recall	F1-score	Accuracy	Error
D_1	0.75	0.81	0.78	0.79	0.21
D_2	0.7	0.78	0.74	0.77	0.23
D_3	0.73	0.93	0.82	0.79	0.21
D_4	0.78	0.77	0.77	0.77	0.23
D_5	0.77	0.75	0.76	0.79	0.21
D_6	0.69	0.8	0.74	0.79	0.21
D_7	0.75	0.79	0.77	0.79	0.21
D_8	0.73	0.84	0.78	0.79	0.21
D_9	0.72	0.86	0.78	0.78	0.22
D_10	0.75	0.8	0.77	0.8	0.2

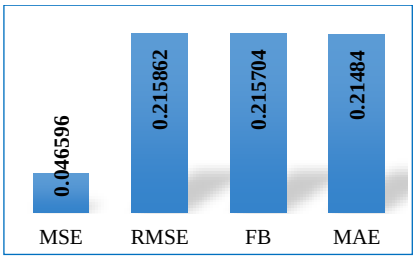


Fig. 7.10. Forecasting error estimates

7.8.5 SMD Selection

The criterion for crowdworker selection is that the predicted availability duration must be greater than the job execution duration. For the selection, the confidence of the prediction model is taken into account. The minimum confidence threshold for selection criterion depends on the particular application, job execution time, etc. For the sake of simplicity, we set this threshold heuristically to 0.74 since the system precision using forecast bias is found to be 0.7371 (refer Section 7.8.3).

7.9 Prediction Using Convolutional LSTM and GRU

In the above-presented approach, we just used the readily available ConvLSTM2D model that is specially meant for 2D spatial data and requires advanced transformations to work with time-series data. The limitation of the model is that it does not have the required flexibility. The only available tuning option for the model was the selection of the hyper-parameters and the transformation technique. This opens up the scope for exploring other options which are more generalized and flexible, which would lead to better model performance. In view of that, in this section, we aim to propose a CNN-based prediction model using the most two popular deep learning methods for time-series data – LSTM and GRU.

7.9.1 Data Preparation

Since we used the time-series data as the raw data, they needed to be converted into a suitable form so that the convolutional filters could be applied to extract the features. The steps that we performed before sending the data to the LSTM/GRU model are elaborated in the following subsections.

Before feeding the time-series data into the LSTM/GRU model, we needed to prepare the data that would be suitable for applying convolutional filters for feature extraction. The user mobility data are time-series data that contain multiple

transactions for in-time and out-time for each user. One transaction represents a pair of in-time and out-time data. These data in the raw form usually have very fewer attributes that can be used in a prediction model. For example, by considering only the duration attribute as a difference of in-time and out-time, the prediction performance may not be sufficient; hence, some form of feature extraction methodology needs to be introduced, as presented in [481]. Using these initial features and a static feature extraction technique may not give the best results. To improve the results, we needed to use a dynamic feature extraction technique that requires the data to be in an image form. The tasks carried on to transform the raw user mobility time-series data into frame-by-frame image data are discussed in the following subsections.

7.9.1.1 Data Frame Creation

To represent the users' mobility, we pursued the following steps for creating the required data frames, as shown in Fig. 7.11.

a) The data frames, each representing one week's data of users' mobility, were created. Each frame has two channels - channel 1 and channel 2, representing the in-time and out-time records of the users, respectively.
b) The data frames have UxD dimensions, where U (number of users) = 50 and D (number of days) = 7.
c) The total number of frames for in-time and out-time is calculated by $2*T_d/D$.
d) Each cell in channel 1 contains the in-time of the user on a particular day. Similarly, the out-time is recorded in channel 2.
e) A user might have multiple entries on a single day. In that case, we normalized the entries by keeping only the entry for the longest duration (for example, if U1 entered four times on D1 and the durations are of 4, 23, 37, and 57 minutes, only the entry for 57 minutes is considered). We adopted this approach to implement a fair share policy so that one SMD would be given a job in only one session. Furthermore, a deep learning model works better with data that is consistent and has less deviation. In the case of MCC, the data may contain session information for multiple short and inconsistent sessions. To avoid fitting inconsistent data, we selected the longest continuous session duration to

be the final session if the gap between the sessions is smaller than a particular threshold time value. Algorithm 1 presents the procedure of calculating the longest continuous session duration. Here, S_n denotes the n^{th} session, IN and OUT represent the in-time and out-time for the respective session, and λ is the threshold criteria for merging two sessions. In our experiment, we considered λ = 0.05, i.e., 5% of the entire duration.

Algorithm 1: Selecting Longest Continuous Session Duration
Input: Raw session data
Output: Updated session data
while (S_{n+1}) if $(S_{n+1}^{IN} - S_n^{OUT}) < \{(S_{n+1}^{OUT} - S_{n+1}^{IN}) + (S_n^{OUT} - S_n^{IN})\} \times \lambda$ then concatenate (S_{n+1}, S_n) end while

7.9.1.2 Data Normalization

Each of the cells in the channels contains time values that are not appropriate for direct input to the prediction model. For this reason, we represented the collected user mobility data (time-series data) as image data. To convert the in-time and out-time frames from temporal data to image intensity data, we needed to normalize the frames. Each cell in a channel represents the time values. Typically, the channel intensity values of an image range between 0-255. Hence, we normalized the time values for both the channels between 0 and 255, as shown in Fig. 7.12.

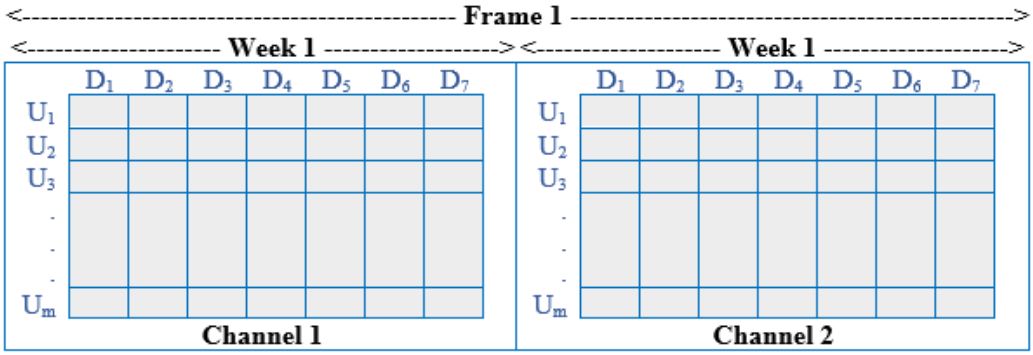


Fig. 7.11. A sample frame for in-time and out-time

The pixel-wise normalization of the time-series data (x) into image intensity (y) is achieved by applying a linear equation, as shown in Eq. 7.26.

$$y = 10.625x \tag{7.26}$$

A sample of data normalization based on input data is shown in Fig. 7.13. The

transition of darker to lighter shades indicates the increasing hour of a day, and the black colour indicates the unavailability of the particular user on that particular day. For example, from the figure, it can be observed that U1 was absent on Saturday and Sunday.

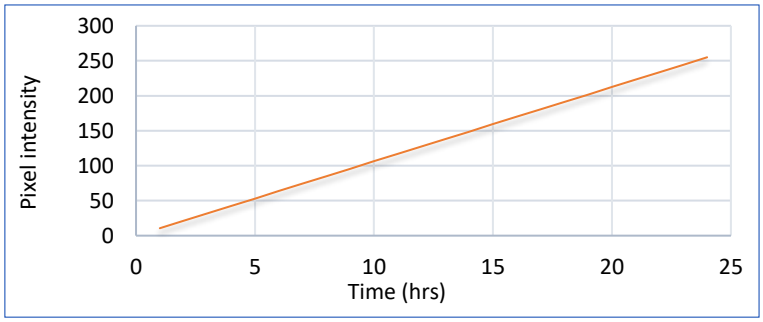


Fig. 7.12. Mutual linear normalization of time and pixel intensity

7.9.2 Feature Optimization

To increase the model accuracy and training and inference speed, we needed to optimize the feature sets. Feature optimization includes extracting new features from the input data and removing the unwanted features. Feature optimization improves the model's performance and makes it more interpretable. The followed steps for feature optimization are discussed in this section.

7.9.2.1 Feature Extraction

A machine learning model execution is just on par with the provided data, and the data is as great as the way it is prepared for the model. This means that a model will perform decently if we have a decent quality of data with sufficient features. Therefore, we need to identify some features from the data that might help in improving the model's performance.

Typically, in the time-series datasets, the features (e.g., length of time-series, period, mean value, standard deviation value, etc.) are not sufficient for prediction modelling and cannot be used straightforwardly. The format of the existing data features may not be suitable for direct analysis and comparison. The new features are generated by reformatting, combining, and transforming the original features. This makes the data suitable for modelling and increases the model's training and prediction accuracy.

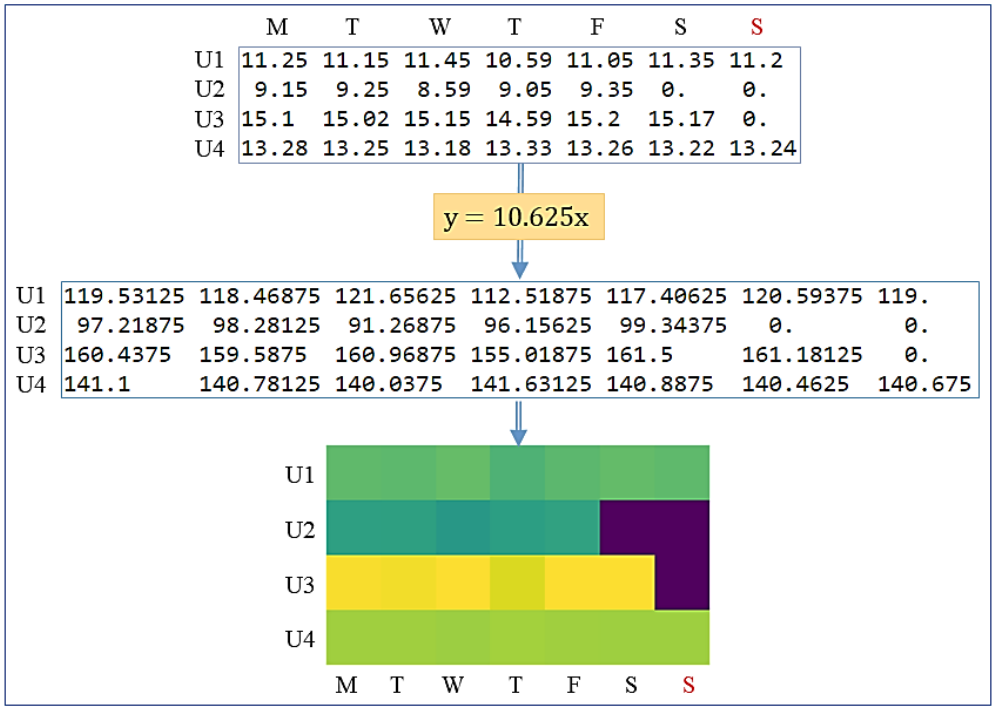


Fig. 7.13. A sample of data normalization based on input data

7.9.2.1.1 Issues with Popular Feature Extraction Methods

There are various methods for feature extraction, which are used depending on the type of the data and problem. PCA, GLCM, etc., are the popular feature optimization methods for dimension reduction in image data. These methods have been proven to work well in time-series predictions and image classifications. However, these methods are not suitable for the problem addressed here because the resource availability prediction in MCC is a generalized time-series prediction problem without any prior knowledge of the important features.

Further, PCA requires some hyperparameter tuning to generate quality features, which is not trivial. GLCM can extract only certain known features and is highly dependent on the characteristics of the data. However, our problem demands a dynamic and generalized feature extraction methodology that is not affected by the data size and quality.

7.9.2.1.2 Need for Convolutional Feature Extraction

In our dataset, except the in-time and out-time of the users, no other information is available. This means there are not sufficient features to model the users' availability pattern. To elaborate further, let us consider the in-time and out-time of

three randomly chosen users over a period of 30 days, as shown in Fig. 7.14. It can be observed that there is a high variance in the in-time and out-time patterns for all users. It also varies day-wise for each individual user. It implies that even if a user's availability seems to follow a pattern, it might not hold true throughout the considered period. This inconsistency could be either intentional or driven by several factors that are not apparently visible from the raw dataset.

However, these nonobvious features might provide some valuable information. But it is impossible to unearth these features manually. For this, we needed some automated and dynamic feature extraction mechanism, which would extract the useful features from the dataset.

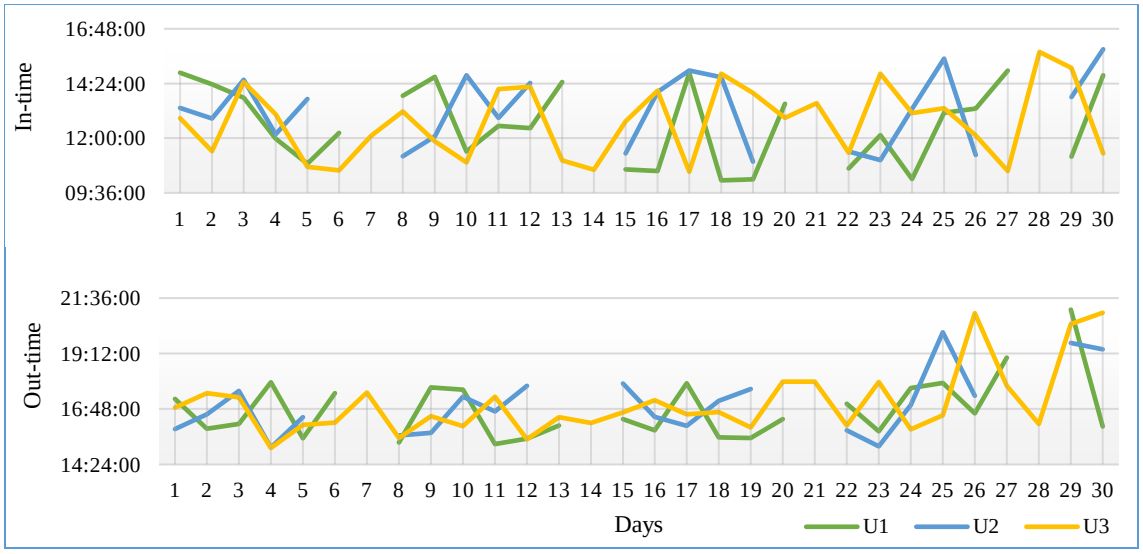


Fig. 7.14. The in-times and out-times of three sample users over a period of 30 days

We found the convolutional feature extraction method as a suitable option for our problem. In many of the dynamic feature extraction problems, CNN has popularly been used. CNN is a supervised classification model comprised of two major segments: a) a convolutional feature extractor and b) a SoftMax classifier. In a traditional CNN, feature optimization (extraction and selection) is automatic. But when using only the convolutional feature extractor, we need a separate feature selection model. A convolutional feature extractor is known for its capability to generate dynamic and new features. Therefore, we transformed our time-series data so that convolutional feature extraction can be applied.

7.9.2.1.3 The Convolutional Feature Extraction Process

In this section, we present the details of the convolutional feature extraction method designed specifically for the problem presented in this chapter. The input to the considered convolutional feature extraction model is shown in Fig. 7.15. We considered the stride or the window size as 1, i.e., the data frame window slides for each day, as shown in Fig. 7.16.

A frame represents the number of values considered in a single instance of the model. The architecture for convolutional feature extraction is shown in Fig. 7.17.

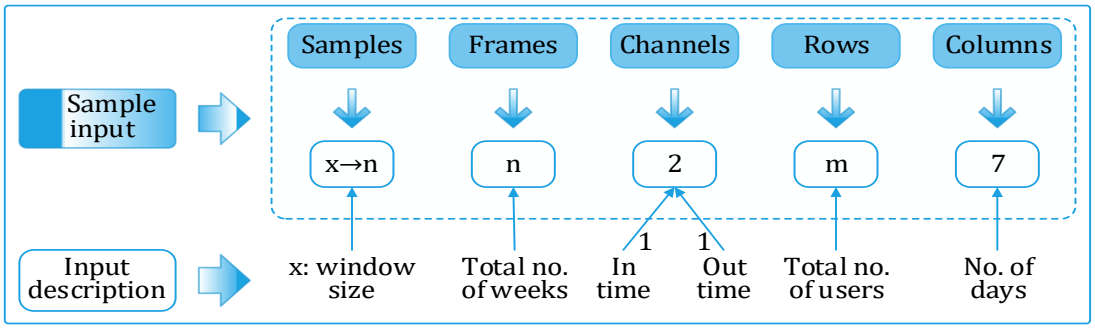


Fig. 7.15. Input parameters for the considered CNN model

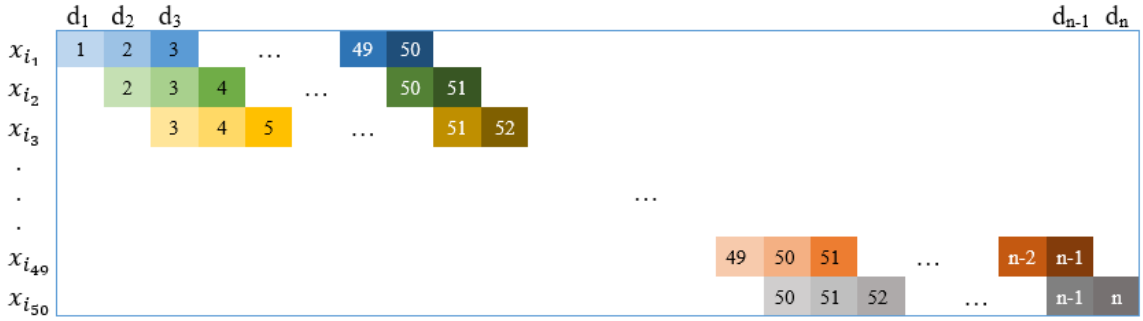


Fig. 7.16. Distribution of the frames for training the feature extractor model

To train the model, we needed to feed the data into it. We did not have a much larger dataset, which was generated by acquiring mobility data for only a few months. Hence, we fed the training data into the model serially, adding one day at the end of the window and removing one day from the front in each iteration.

The distribution of the frames to train the feature extractor model is shown in Fig. 7.16. In the figure, x_{i_j} represents the input frame for a particular user j, while i is the set of days available as prediction input. Here, we considered the value of i as 50 (however, the value of i would vary according to the total number of available samples). Since we considered the mobility data of 50 users (m), the maximum

value of j in this problem would be 50, and j would iterate for all the users, i.e., 50 times (m = 50). Here, $d_n$represents the data instance for a particular day n. Since we considered two datasets consisting of user mobility data for 120 and 150 days, the maximum value of n would be either 120 or 150.

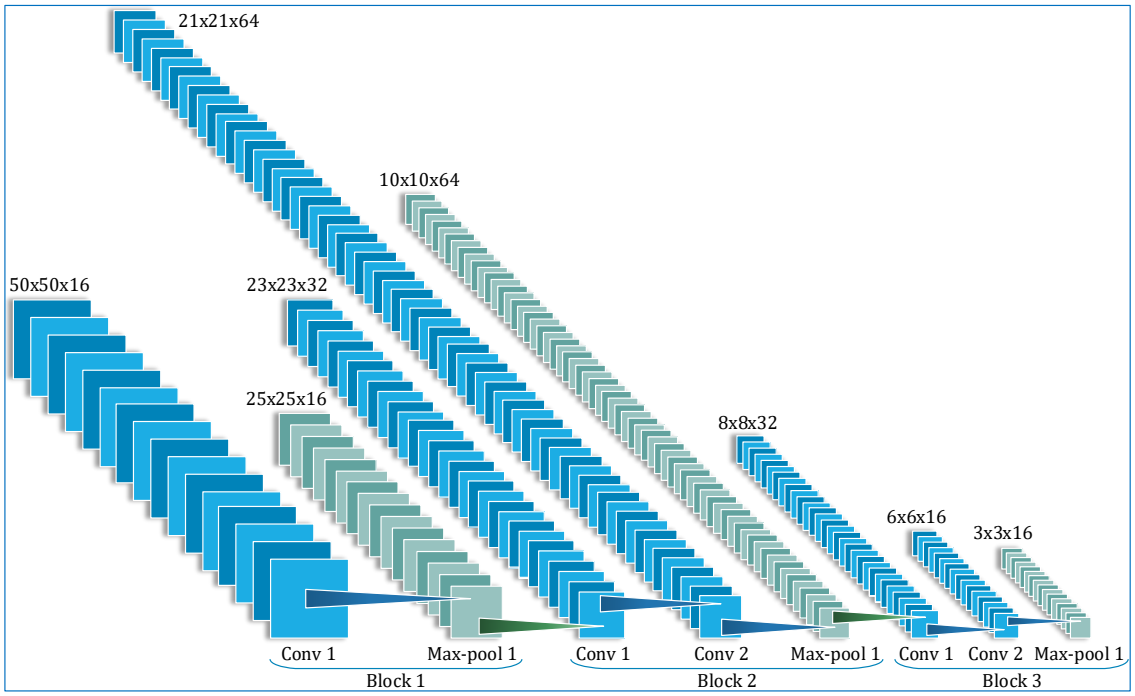


Fig. 7.17. Convolutional feature extraction architecture

After creating the input frames, we proceeded to the feature extraction phase. A novel feature extraction model is developed specifically for this work, as presented in Fig. 7.18. The proposed CNN model contains the following five segments:

- **Weekly mobility data:** Each week of mobility data for all the users in the considered duration is represented by one data frame. These frames are the input to the feature extraction model.
- **Channels for in-time and out-time:** Each frame is split into two channels; one for in-time and the other for out-time for all the users. The subsequent functions were repeated for each channel separately.
- **Frame-by-frame training:** For training the convolutional feature extractor, the frames in a group of 50 were arranged in a single block. In the next timestep, a single stride of each frame was made for further predictions till all the frames were considered, as shown in Fig. 7.16.

- **Model:** This is the CNN model for convolutional feature extractor without the classifier, as shown in Fig. 7.15. The model is architected using three blocks of varied convolutional and max-pooling layers, as shown in Fig. 7.17. We considered a filter of dimension 3x3. Block 1 comprises a single convolutional layer with a dimension of 50x50 and 16 filters and a max-pooling layer of dimension 25x25. Block 2 comprises two convolutional layers with a dimension of 23x23 and 21x21, along with 32 and 64 filters, respectively. It also has a max-pooling layer of dimension 10x10. In block 3, there are two convolutional layers, 8x8 and 6x6, along with 32 and 16 filters, respectively.
- **Feature extraction:** The extracted features from each input data frame were stored in a vector form, which was fed into the LSTM and GRU prediction models.

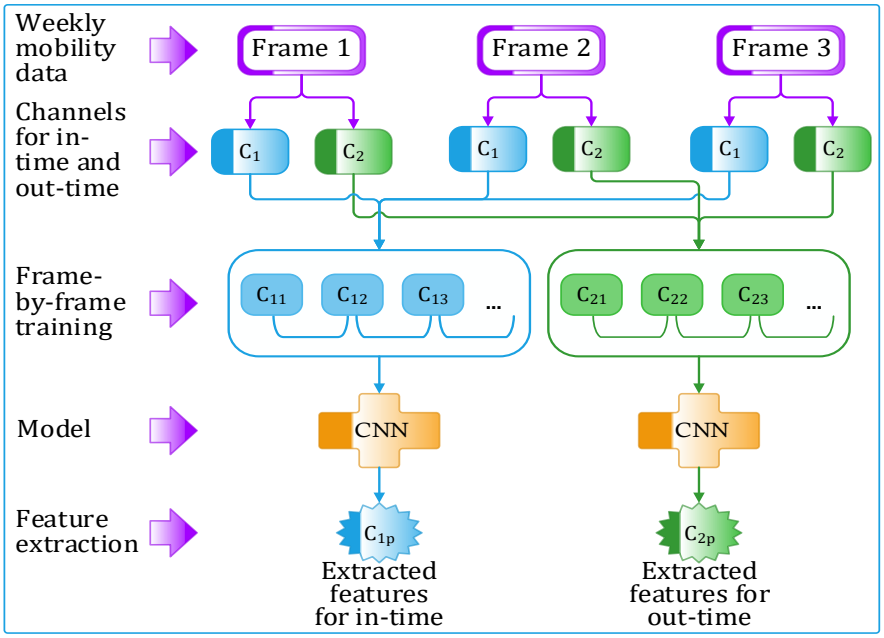


Fig. 7.18. Feature extraction for in-time and out-time using CNN

7.9.2.2 *Feature Selection*

Not all the features in the dataset are really useful. Irrelevant and redundant features increase the training time, decrease the accuracy, and make it complex to interpret. That is why, for model construction, it is important to select only those features that are essential and can represent all the features. Feature selection is used to select relevant features from the dataset by eliminating the redundant or irrelevant features or the strongly correlated features in the data without losing

much information. The primary reasons for using feature selection and the popular regression methods are mentioned in Fig. 7.19. The features that contribute most to the desired prediction or output are generally retained. Even after convolutional feature extraction, the model may contain some features that may cause performance degradation due to multi-collinearity. Feature selection not only removes multi-collinearity and improves the prediction accuracy but also reduces training time, simplifies the model for better interpretation, and improves the chances of generalization, thus, avoiding overfitting.

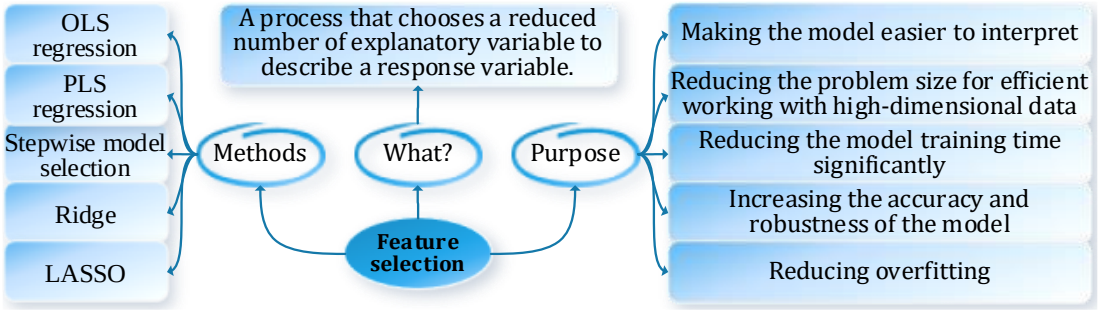


Fig. 7.19. Purpose of feature selection and the popular regression methods

Though there are a few regression methods for feature selection, as shown in Fig. 7.19, we avoided the traditional methods such as OLS regression, stepwise model selection, and PLS regression, etc., due to their sensitiveness to random errors. In the case of multi-collinearity in the input values, Ridge and LASSO methods perform effectively. However, we preferred LASSO because the major problem with Ridge is that though it shrinks the coefficients nearly to zero but not exactly to zero. Hence the ridge regression fails to provide an unambiguous and easily interpretable sparse model, especially when the number of predictors is large [721]. On the other hand, LASSO offers a better prediction accuracy and model interpretability by eliminating the irrelevant variables/coefficients that are not associated with the response variable. If there is a high correlation in a set of predictors, LASSO picks only one among them while shrinking the others exactly to zero. The leftover non-zero values are selected to be used as features in the model. This method leads to a reduction in variance without increasing the bias much. This is especially beneficial when the dataset consists of a small number of observations and a large number of features. The cost function of LASSO is defined by Eq. 7.27.

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - (w_0 + \sum_{j=1}^{X} x_{ij}w_j))^2 + \lambda \sum_{j=1}^{X} |w_j| \quad (7.27)$$

where, m is the total number of training samples or instances in the dataset, X is the total number of features, y_i represents the value of target variable for i^{th} training example, $x_{ij}$is the i^{th} observation for j^{th} feature, w_0 is the intercept term, w_j represents the weight of the j^{th} feature, and λ is the tuning parameter that controls the feature reduction. The larger λ becomes, the more feature coefficients shrink to zero. Also, as λ increases, bias increases, and variance decreases. Here, the goal is to minimize the error function $\sum_{i=1}^{n}(y_i - (w_0 + \sum_{j=1}^{m} x_{ij}w_j))^2$, subject to the regularization term $\lambda \sum_{j=1}^{m} |w_j|$.

After applying the convolutional feature extraction, we had a total of 46,384 features in our considered user mobility dataset. After using LASSO on this feature set, the total number of features was reduced to 4,976.

7.9.3 Prediction Method

In this section, we present the proposed CLSTM and CGRU models.

7.9.3.1 Convolutional LSTM and GRU Modelling

To model the CLSTM and CGRU, we used two layers of the LSTM and GRU networks, respectively, with an input of frame groups with 50 samples, as shown in Fig. 7.16.

The layered representation of the CLSTM/CGRU prediction model is shown in Fig. 7.20. The objective of both models is to maximize the conditional probability of the convolutional features at the current timestep (C) over the input (N), for which the prediction is to be made, at the next timestep, as given in Eq. 7.28. This implies that the model optimizes the current prediction based on N. The timestep and the input frames can be modified during the training phase to check for improvements. In our experiment, the number of input vectors is quite low; therefore, we proceeded with a single stride over the input vectors for each timestep.

$$p(C|N) = \prod_{j=1}^{m} p(\mathrm{x}_{(\mathrm{i+j})_\mathrm{j}}, N) \quad (7.28)$$

where, m is the total number of users, $x_{i_j}$is the input frame for user j, and i = 50.

The input to the initial LSTM/GRU cell is the convolutional feature vector of the input data at timestep t, while the current LSTM/GRU cell output at timestep t is q_t, and the hidden states are h_t. The input to the next LSTM/GRU cell is the output of the previous LSTM/GRU cell, which then passes into a SoftMax layer for classification, as shown in Eq. 7.29.

$$p(x_t|(x_{t+1})) = SoftMax(W_f h_t) \tag{7.29}$$

where, W_f is a learnable parameter, and x_t and x_{t+1} are two adjacent input vectors to the model.

The output hidden states at the current timestep for CLSTM and CGRU are generated using Eq. 7.30 and Eq. 7.31, respectively.

$$h_t = LSTM\ (x_t, c_{t-1}, h_{t-1}) \tag{7.30}$$

$$h_t = GRU\ (x_t, h_{t-1}) \tag{7.31}$$

7.9.3.2 Training

Training a model is used for making the model learn the trainable parameters and tuning the hyperparameters. The objective of the training phase is to decrease the error in the training dataset d of size m. The training objective function T is defined by Eq. 7.32, where p_i is the predicted output and a_i is the actual output.

$$T = \sum_{i=1}^{m} -\log p\ (p_i|a_i) \tag{7.32}$$

7.9.4 Experiment, Results, and Analysis

This section presents the details of experimental environment and the results and analysis.

7.9.4.1 Experimental Setup

The hardware specifications of the system used in training and testing the prediction model is as follows:

- Operating system: Windows 10 Professional
- CPU: AMD® Ryzen™ 7-3700X Processor
- RAM: 32 GB DDR4
- GPU: NVIDIA GeForce® GTX 1080 Ti

The Windows version of the Python (64-bit) with IPython notebook [722] was used to build the models. Several important APIs including TensorFlow [723], NumPy packages [724], SciPy [725], scikit-learn [726] and Matplotlib [727] were used in the experiment. NVIDIA CUDA Version 9.1 [728] for Windows environment was used to avail GPU (graphics processing unit) computing.

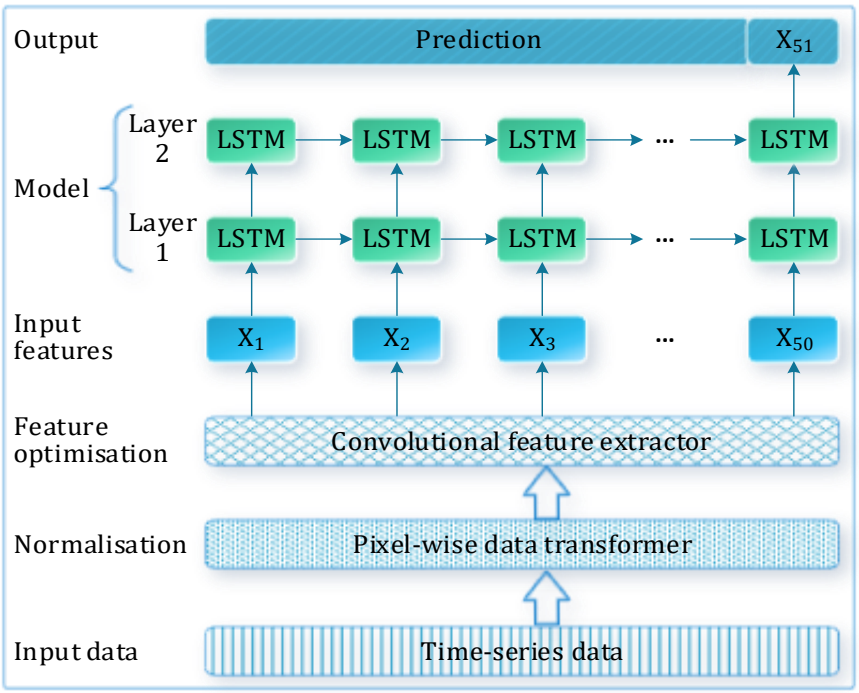


Fig. 7.20. Layered representation of the CLSTM prediction model

7.9.4.2 Training and Testing Split

To conduct a prediction experiment and evaluate the model performance, the complete dataset is generally split into two parts as training and testing sets. This ensures that the model trains on the known data and is able to perform predictions properly for unknown data, which is validated using the test set. Though in this problem, the dataset is too small (150 data instances for each user) to split into two sub-datasets, we had to do it because no other data was available for validating the accuracy of the model in case of unfamiliar data. Hence, the existing data was split into training and testing sets in the ratio of 7:3, as it is found that 70% as the training set can sufficiently represent the data patterns. This splitting ratio is used throughout the experiment. Further, the sequence of the entire data is maintained properly after splitting.

7.9.4.3 Experiment and Result Analysis Consideration

As discussed in Section 7.7.2, we used two datasets for understanding the model performance for a lower volume of data. This is important in this problem due to its implementational and usage nature.

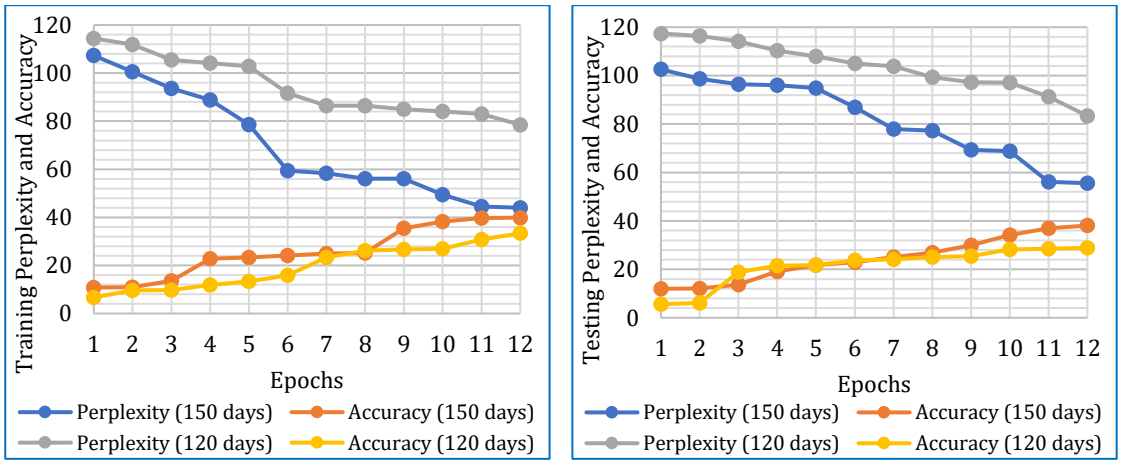
To evaluate our proposed model's performance, we considered using the performance measurement metrics as mentioned in Section 7.6 over 20 epochs as the training model perplexity and accuracy did not improve after that. We used training and testing statistics for evaluation, which shows the perplexity vs. accuracy graph for each of the datasets. For comparison, we considered using GRU with 12 epochs without convolutional features and measured its efficacy over CGRU.

7.9.4.4 Prediction Results Using Conventional GRU

The training and testing statistics (accuracy and perplexity) of the GRU prediction model are shown in Fig. 7.21. The training and testing accuracy for 150 days are 39.9 and 38.1, respectively, and for 120 days, 33.4 and 28.9. The accuracy improvements of CLSTM over GRU are 133.18% and 121.2% for 120 and 150 days, respectively.

7.9.4.5 Prediction Results Using Convolutional GRU

The training and testing statistics (accuracy and perplexity) of the CGRU prediction model are shown in Fig. 7.22. The training and testing accuracy for 150 days are 79.7 and 76.8, respectively, and for 120 days, 69.8 and 66. The accuracy improvements of CLSTM over GRU are 2.11% and 7.48% for 120 and 150 days, respectively.



(a) (b)

Fig. 7.21. Statistics of GRU for two datasets of (a) training and (b) testing

7.9.4.6 *Comparing CGRU with Traditional GRU*

Fig. 7.23 shows the improvements of the testing accuracy with respect to (a) the number of days of data used and (b) the prediction model used. With an increment of 30 days of data, the GRU model shows a significant improvement of 15.47% over CGRU. This is because the performance of GRU with less amount of test data is very bad. A slight increase in the data volume makes the performance significantly better. Whereas, since the performance of CGRU is already far better than GRU, the increase in data amount does not make much difference in the performance of the CGRU model. For the same reason, when comparing the GRU and CGRU models for 120 and 150 days of data, GRU lacks significantly.

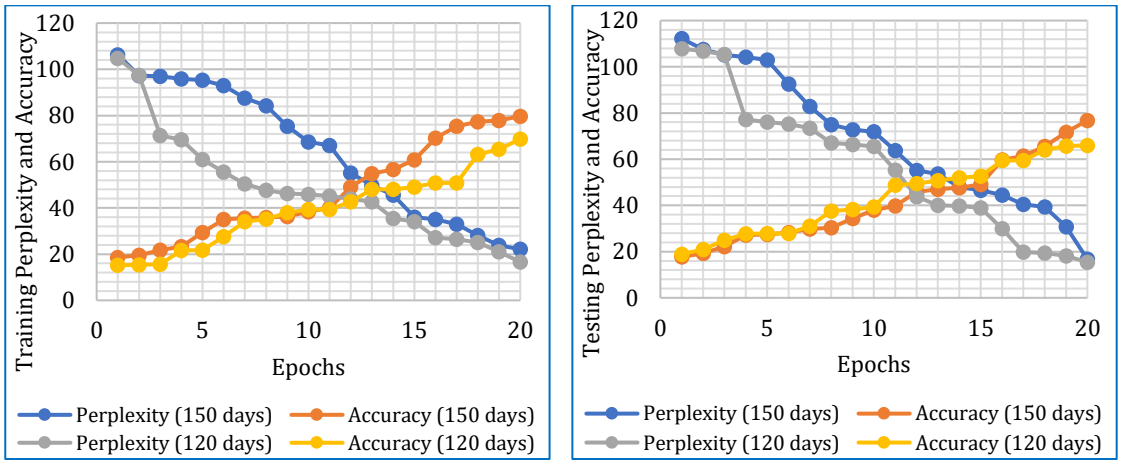


Fig. 7.22. Statistics of CGRU for two datasets of (a) training and (b) testing

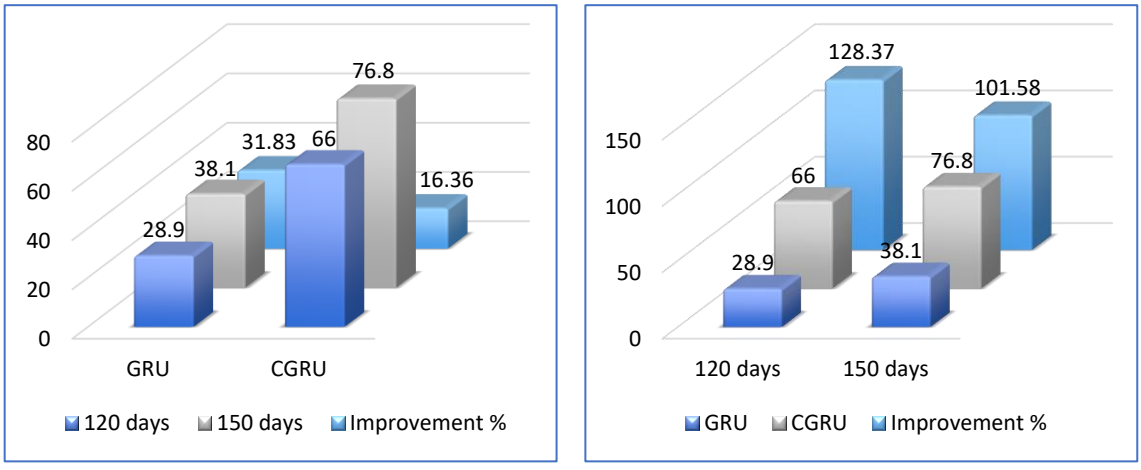


Fig. 7.23. Improvement percentage of testing accuracy with respect to (a) number of days of data used and (b) prediction model used

These statistics prove that CGRU outperforms GRU in both cases when the volume

of data is low and high. So, the CGRU-based prediction model is suitable for crowd computing applications where sufficient crowd data is not available.

7.9.4.7 Prediction Results Using Conventional LSTM

The training and testing statistics of the LSTM prediction model are shown in Fig. 7.24. The training and testing accuracy for 150 days are 45.6 and 44.35, respectively, and for 120 days, 41.9 and 32.89. The accuracy improvements of CLSTM over LSTM are 104.9% and 90.03% for 120 and 150 days, respectively.

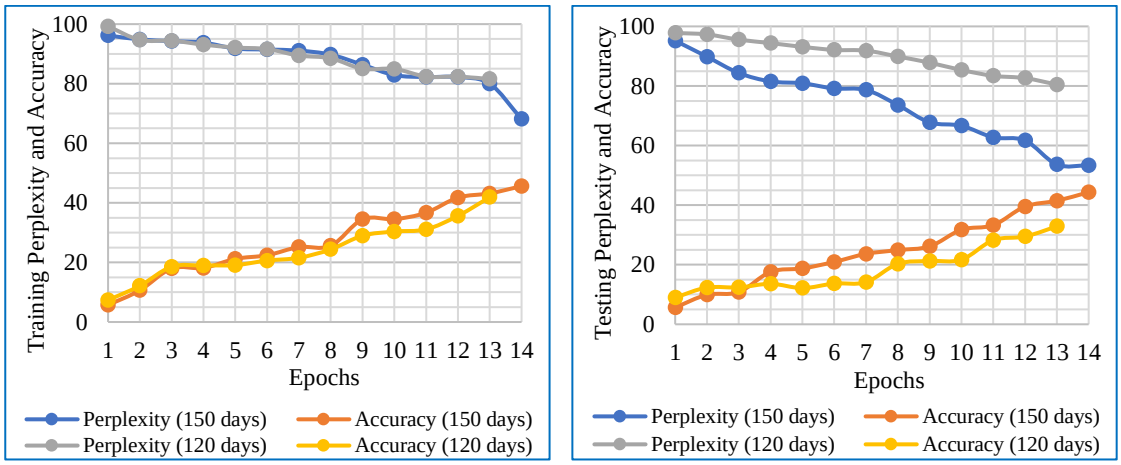


Fig. 7.24. Statistics of LSTM for two datasets of (a) training and (b) testing

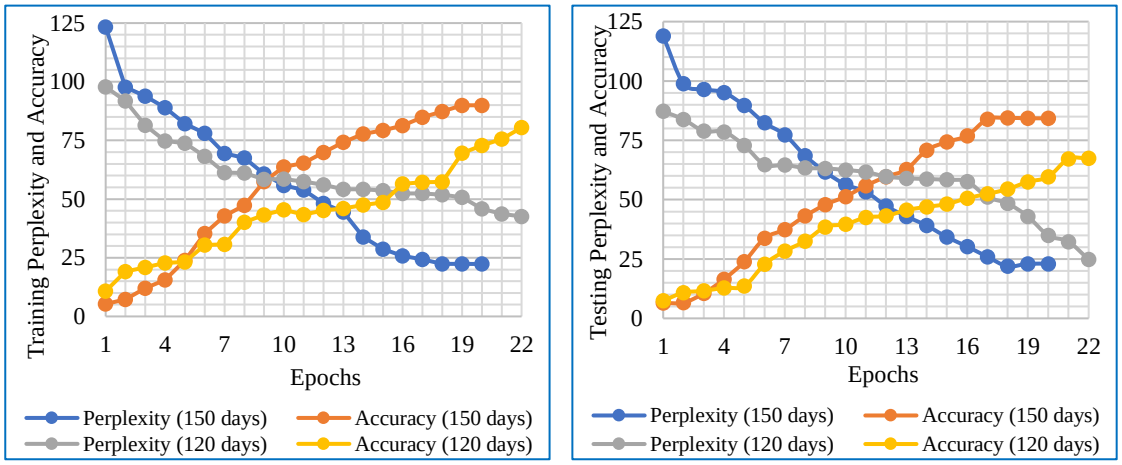
7.9.4.8 Prediction Results Using Convolutional LSTM

To evaluate the proposed CLSTM model's performance, we considered evaluating these metrics over 22 epochs because the training model's perplexity and accuracy did not improve after 22 epochs. The perplexity vs. accuracy graph for training and testing is shown in Fig. 7.25. It is observed that for 150 days of data, the achieved training accuracy of the model over 22 epochs is 89.97%, and the testing accuracy is 84.28%, while for 120 days, it is 80.44% and 67.39%, respectively. A difference of 5.69% and 13.05% are seen between the training and testing models in the two cases, which signifies that the model is not overfitting.

7.9.4.9 Comparing CLSTM with Other Methods

The performance of the proposed CLSTM was compared with the other three prediction models. Fig. 7.26 shows the accuracy comparison between the proposed CLSTM model and the other compared models. It is observed that CLSTM has significantly higher accuracy over GRU and LSTM. However, there is not much

difference between CLSTM and CGRU in terms of accuracy. This suggests that when the traditional LSTM and GRU are combined with our proposed convolutional feature extractor, they perform considerably better. This proves the efficacy of the proposed model.



(a) (b)

Fig. 7.25. Statistics of CLSTM for two datasets of (a) training and (b) testing

(a) (b) (c)

Fig. 7.26. Accuracy comparison between (a) GRU and CLSTM (b) LSTM and CLSTM and (c) CGRU and CLSTM

To compare the model's sensitiveness towards the input data size, we checked the model's accuracy by varying set sizes. It is observed from Fig. 7.27 that all the models perform better with the larger data size. However, for the traditional LSTM and GRU models, the accuracy improvement with larger data set is comparatively greater than CLSTM and CGRU. It is always desirable to have a higher improvement percentage, but the final attained accuracy value must also be considered. This implies that even if the traditional models have the highest improvement

percentage, their final attained accuracies are too low compared to the convolutional models. Furthermore, the accuracy improvement of CLSTM is much higher than CGRU, from which we can expect to have further higher accuracy for CLSTM with a larger data set.

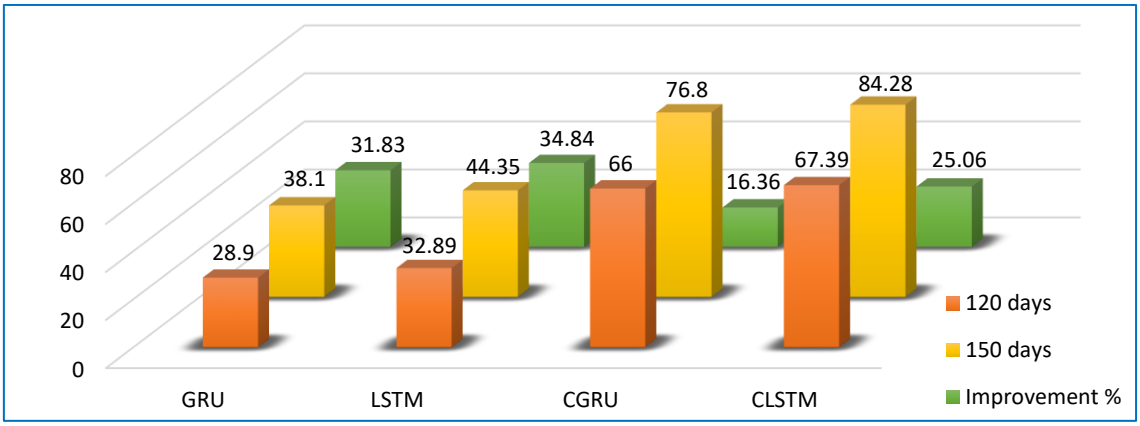


Fig. 7.27. Improvement percentage of testing accuracy of each model with respect to the number of days of data used

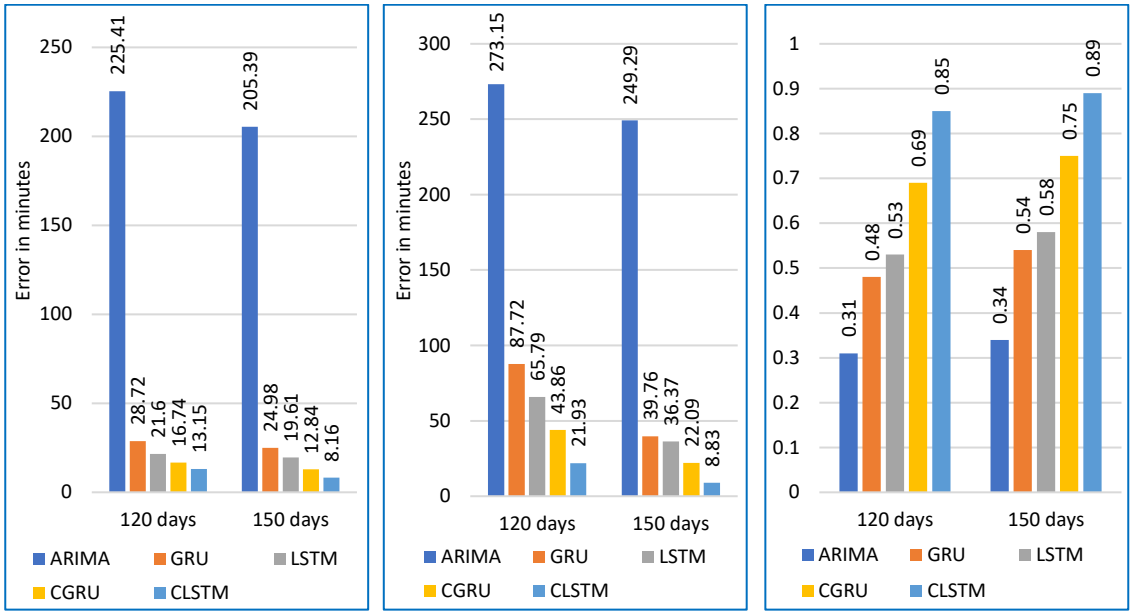
We further compare CLSTM with other prediction methods: ARIMA, LSTM, GRU, and CGRU in terms of the prediction error estimation. In Section 7.2, we categorically stated that the traditional statistical prediction methods would not work well for this problem. However, to prove our claim, in this comparison, we included ARIMA, the most popularly time-series prediction method, along with other deep learning based prediction methods.

Fig. 7.28 shows the error estimation results in terms of MAE, RMSE, and R^2. The MAE and RMSE errors are calculated in minutes, while R^2 is presented in percentage. It can be observed that for each error estimation, CLSTM produces least errors than other methods. It was also observed that not only the CLSTM but other models also performed better if the size of the training dataset increased.

It can be further discerned that ARIMA exhibits the worst error estimations. This can be due to the fact that ARIMA cannot handle datasets with missing values. In this, the missing values should be handled by some fillers. Furthermore, ARIMA is suitable for short-time prediction because due to the absence of memory, the prediction window is very limited. That is why it fails in long-term prediction.

Since ARIMA generates a very high degree of prediction error which is

unacceptable for our problem, we did not consider it for further comparative analysis.



(a) (b) (c)

Fig. 7.28. Error comparison of CLSTM with ARIMA, GRU, LSTM, and CGRU based predictions: (a) MAE (b) RMSE, and (c) R2

7.9.5 SMD Selection

After the initial ranking, the top-ranked (as per other criteria) SMD is forwarded to the prediction model along with its in-time and the job duration. The minimum required out-time is calculated by adding current time and job duration. The CLSTM model predicts the out-time against the in-time of the SMD and forwards it to the selection module. Now, if the predicted out-time is more than the minimum required out-time, then the selection module selects the SMD.

7.10 Discussion

In this section, we present a brief discussion on the crucial aspects of the proposed CLSTM/CGRU methods for availability prediction.

7.10.1 GRU vs LSTM

GRU and LSTM are popular deep learning methods used in sequence modelling and time-series analysis and predictions. GRUs are sometimes preferred over LSTM because GRUs are more straightforward and easier to modify. Also, since GRU uses fewer training parameters, it not only needs much less memory, but also

is faster to train and execute than LSTM [729]. GRUs are generally preferred when we require a decent degree of accuracy but do not have sufficient computing resources to train data samples with longer sequences.

However, there are some issues with GRUs compared to LSTM, such as the GRUs cannot regulate the amount of memory content that is to be forwarded to the next unit, as they do not store cell state. Whereas LSTMs are capable of regulating the amount of information that is to be forwarded to the next unit. Nevertheless, we tested our model with both LSTM and GRU to compare the effectiveness of these models for this particular problem.

7.10.2 Concern Over Space and Time Cost of LSTM

LSTM models provide satisfactory accuracy in predicting or forecasting. However, there is a general concern over LSTM models is that they usually involve high processing and memory costs because of linear layers present in each cell.

Nowadays, the space requirement issue is not a big deal considering the availability of cheap and compact memory. Similarly, the processing requirement can be minimized by exploiting the parallel execution capability of the modern many-core GPUs, which are powerful enough to carry out computing-intensive jobs quickly.

Furthermore, the complexity of the machine learning model mainly contributed to its training process. During the testing, when the model is already built, the runtime is minimal. Hence, the time requirement of implementing the LSTM-based availability prediction process would be negligible, and the proposed system can be deployed for real-time purposes.

In our proposed MCC system, the prediction model runs on a dedicated MCC coordinator. Moreover, as we considered a local MCC environment, the number of SMDs or users is limited. Also, the time span of data gathering is not much wide. Therefore, the data volume is not exceptionally huge. As a result, the hardware requirement would not be too extensive. In our experiment, we leveraged the parallel processing capability of NVIDIA GPU using CUDA API, and it was found quite acceptable.

However, if the MCC is implemented in an environment where such a powerful

system is not present, or if the data size is hugely voluminous such as in a smart city scenario, the cloud services can always be availed. Today's commercial cloud services such as AWS, Google, etc., not only offer on-demand highly scalable GPU and TPU (tensor processing unit) stacks but they are also proffered with viable and affordable prices. Therefore, the LSTM model can be trained on the cloud, leveraging the powerful heterogeneous hardware environments to achieve significant speedups.

7.10.3 CNN for Temporal Data

It is observed that LSTM performs quite satisfactorily with higher accuracy in many forecasting applications while combined with CNN. In our experiment also, we observed significant performance improvement by using CLSTM compared to only LSTM. It is known that LSTM works well on temporal data, whereas CNN is designed to exploit the spatial correlation in the data and works well on the data having spatial features; they are not generally capable in efficient handling of complex and long temporal dependencies [730]. While in our SMD availability problem, the dataset is only temporal in nature. Even then, the CLSTM provides much better performance than LSTM. In our dataset, there is some inherent inconsistency that cannot be reflected using traditional LSTM. So, to avail the advantages of CNN, the sequential data is needed to be mapped somehow into a spatio-temporal pattern.

To the best of our knowledge, besides extracting the new features using CNN, the technicality behind mapping temporal or sequential data to spatio-temporal data for prediction performance improvement is still not quite obvious. However, from Fig. 7.14, it is intuitive that identifying the shape of the users' in-time and out-time patterns may significantly improve the performance of the predictive model. The obvious reason behind this assumption is that the mobility pattern shape can capture the hidden features associated with the dataset. If we treat the users' in-time and out-time patterns as an object, then acquiring the object's shape would allow us to extract the unknown features, as in the case of image analysis.

7.11 Limitations and Further Scopes

Though the experimental results indicate satisfactory accuracy, there is a scope for improvement of this work. Since the job completion time of different devices will be different, the variance of the completion time over various devices could be measured separately to make the model more accurate and justified.

Further, we considered only the longest duration of a user's presence time in the network for each day. This somehow curbs the effectiveness of the crowdworker selection. It may happen that a session duration of a crowdworker is sufficiently larger than a job to be assigned. However, since this session is not this crowdworker's longest session on that day, it would not be listed as the probable candidate crowdworker, even it fulfils all other criteria adequately. Increasing the depth and, in turn, the dimension of the data frame may allow the model to consider all the sessions.

The prediction may further be enriched by considering the difference in the predicted out-time and the job duration to get a confidence value that may be used to rank the SMDs based on availability in the absence of any prior ranking scheme, where the minimum confidence threshold for selection criterion will depend on the particular application, job execution time, etc.

7.12 Summary

In this chapter, we designed a model for predicting mobile devices' availability for a local MCC where people join the MCC regularly and stay connected for varying periods. Before submitting the job to a crowdworker, the probability of the considered crowdworker being available until the job execution is finished is evaluated. If the job execution time is greater than the predicted availability duration, an alternative crowdworker is considered. This will improve the QoS of MCC by minimizing the job reassignment.

We presented a novel dynamic feature extraction method where the features are unknown. Based on the extracted and selected features, we experimented with a convolutional prediction model applied on LSTM and GRU. Utilizing the real user mobility traces for a Wi-Fi AP deployed in a research lab, a convolutional LSTM (CLSTM) and a convolutional GRU (CGRU) based prediction methods were

applied to predict the out-time of the user for each in-time. The prediction model was implemented on two datasets of different volumes.

Our experiment showed that introducing the proposed convolutional feature extractor to the LSTM and GRU prediction models exhibit significant improvement in the prediction accuracy compared to the traditional LSTM and GRU-based prediction models. The proposed model competes favourably against another models, viz. ARIMA, popularly used in time-series prediction. The proposed CLSTM and CGRU models give satisfactory performance accuracy not only when the dataset is large enough but also for the small-sized dataset. It is observed that with an increase in dataset size, the performance of the model improves significantly. Also, with the increase in the dataset, the error estimation of the model gets better. However, compared to the CGRU, CLSTM model exhibited better performance. Thus, we can conclude that the proposed CGRU model can be a feasible resource availability prediction model in mobile crowd computing both with small- and large-scale user mobility data.

8

Mobility-Aware Service Provisioning in P2P MCC

"Movement is everything, direction is what matters." --- Manfred Hinrich

8.1 Introduction

Due to their usefulness, flexibility and capability to be an all-purpose device, SMDs have been integrated with our daily life. With the increased demand for diverse pervasive and ubiquitous computing and other online services, people are becoming greatly dependent on their SMDs for more effective and functional services to satisfy these needs. Accordingly, various utility and imperative applications are being developed that are supposed to make our lifestyle easier. Furthermore, many applications involve real-time inputs such as sensor data and crowdsourced data which might big in size. As a result, they often comprise computing-intensive jobs which demand significant computing capacity. In summary, many sophisticated mobile applications and services demand certain hardware and software resources requirements.

But all the SMDs do not boast equivalent functionality and resource capacity. The SMD market is crowded with a huge variety of devices incorporating a diverse range of resources. Some of them are very high-end devices with greater capacity, and thereby can afford to host various types of applications and service. Whereas, some of the SMDs that are aimed only for basic purposes, might have inferior configuration and capacity; hence, they may not afford to have/run the resource-demanding applications and services are low in resources [249]. If they want to, they need to offload their job to the cloud, which is known as mobile cloud computing [451]. But, accessing cloud services are not always viable or desired. In the context of this chapter, following are some of the issues with the cloud:

- It may happen that due to some reason (e.g., unavailability of internet connection or network infrastructure failures, mobile data used up and there is no Wi-Fi, etc.) cloud is not accessible.

- A real-time application may not tolerate the latency required to obtain a cloud service.
- A reluctant user may not want to use paid cloud services.
- With limited wireless bandwidth, sending compute and data-intensive jobs to the cloud causes network and performance bottleneck for mobile cloud services.

In these scenarios, P2P MCC (PMCC) can be a worthwhile alternative. In PMCC, a resource-low mobile device can lend resources from a peer or neighbouring resource-rich mobile device. We perceive the idea of PMCC as a collection of heterogeneous SMDs that are locally connected (usually, through WLAN, Bluetooth, hotspots, etc.) and are capable of and willing for sharing their resources or services among themselves, when in need.

The proposed service lending scheme using PMCC can be used in several real-life scenarios and has a wide range of significant applications. PMCC will play a critical role for anytime-anywhere computing service provisioning through impromptu collaboration. These services may include educational programs, multimedia services, business applications, instant messaging, etc. PMCC may also be useful for hiring network services such as routing and packet forwarding to other nodes. Moreover, ad-hoc environments such as military battlefields and disaster management areas also can make use of PMCC.

However, such a local PMCC is prone to become fragile due to the users' mobility. Availability of a service provider within a dynamic environment as PMCC cannot always be guaranteed. The resource availability is greatly affected by user mobility. It may happen that a service provider agreed to lend its resources but exits before the assigned job is finished. For an efficient PMCC, the resource consuming and providing SMDs should ideally be together or in contact until the resource is fully availed.

The unavailability of the designated service provider can be tackled by reassigning the job to another service providing SMD, but if the SMDs move in and out of the PMCC very frequently, resource switching and the associated overhead, will

increase accordingly and that will hamper the QoS of PMCC in terms of throughput and reliability. Therefore, service provisioning in PMCC must be supported with an accurate mobility prediction mechanism.

In a traditional mobility prediction approach, the mobility of each SMD is assessed individually, and if there is a change in mobility, it is assumed that the device might be unavailable. In this chapter, we try to assess the relative mobility of the peer SMDs. We argue that instead of focussing on the absolute mobility or stability, of the SMDs, it is more practical to ensure the relative stability between the service consuming and providing SMDs.

We define relative stability as an attribute of a mobile node that determines its stability, which says whether it is stationary or in motion with respect to the nodes in its neighbourhood, not its absolute movement (that which is determined by GPS) with respect to its geographical surroundings. We further categorise relative stability based on the time fractions of the relative stability as follows:

- **Continuous relative stability:** A group of SMDs stay together for a certain duration without being separated in between.
- **Discrete relative stability:** The togetherness of the SMDs is not continuous rather discrete, i.e., a pair or group of SMDs meet at some point of time, get separated, and after a while meet again.

In this chapter, we demonstrate two such scenarios. In the first case, presented in Section 8.5, we consider a single cluster of peer SMDs which continuously stay together for some duration and share resources. In the second case, presented in Section 8.6, we consider a multi-cluster PMCC providing an inter-cluster service provisioning.

Overall, in this chapter, we aim to achieve the following objectives:

- Lay out a service provisioning model for PMCC.
- Predict the relative stability of the SMDs in a single-clustered PMCC with respect to their neighborhood.
- Find a group of mobile users who tend to stay together longer.
- Find out relative mobility and stability among the service requester, provider,

and the carrier in a multi-clustered PMCC based on their individual mobility pattern.

- Based on the mobility pattern, predict the contact times between these three entities.
- Specifically, we attempt to find answers to the following questions:
  a) Which are the users stay together within an AP for a certain period of time? When and how long?
  b) Whether they remain together when they go to another AP.
  c) If b) is not true then is there any pattern that user (u_1) belongs to a group (g_1) in AP_1 but belongs to g_2 in AP_2 steadily?

8.2 Single- and Multi-cluster PMCC

Single-cluster PMCC: In a single-cluster PMCC, the SMDs are connected through only a particular small-range communication network. The scale and coverage of the cluster is limited to the capacity of the network for supporting multiple devices simultaneously and its range. A resource requester can avail computing resources from one or multiple SMDs. If a single SMD has sufficient resources and is willing to cater the service demand, then the job is directly sent to it. If multiple SMDs are required, the job is divided and distributed among the service providers. A typical single-cluster PMCC model is shown in Fig. 8.1.

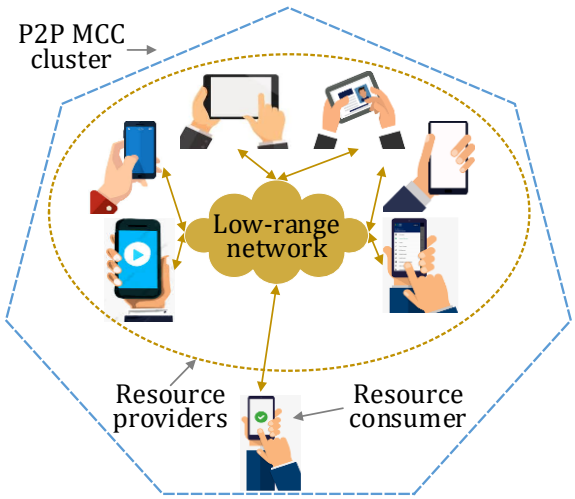


Fig. 8.1. A typical single-cluster P2P MCC model

Multi-cluster PMCC: In this model, there are multiple isolated PMCC clusters, typically belongs to the same organisation. In our experiment, a cluster is

represented by an AP and each cluster is disjoint, i.e., an SMD is connected to only one AP at a time. Within a cluster, one leader is selected or elected. All SMDs within the cluster forward the service request to the leader. If the requested service is not available in the cluster, then searching will be done in another cluster, typically in the clusters that are immediate neighbour to the original cluster. If all the clusters are dense in nature, i.e., all the clusters are placed near to each other, then it is easy to find the service across the clusters; but if the clusters are sparse and placed far from each other and also not directly connected, as shown in Fig. 8.2, then the mobility property plays a crucial role in searching as well as getting the service. In such cases, an SMD is selected, within the cluster, that has the highest mobility in the requesting cluster. It goes to another cluster that might have the requested service, collect the service, and get back to the requesting cluster and provide to the requesting node.

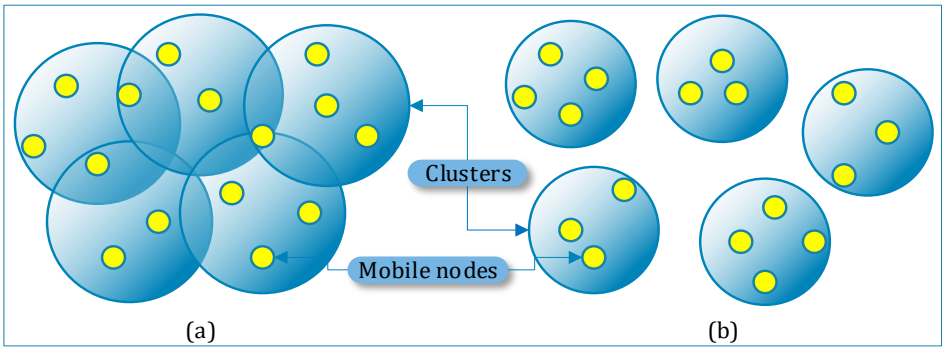


Fig. 8.2. Clustering in (a) dense network and (b) sparse network

8.3 Service Lending Scenarios

Let us theorise an PMCC environment where a low-end SMD can get/hire the service, that it requires but does not have, from one SMD that has the concerned service. To make it simple, software and hardware capabilities can be delivered and consumed as services, and if an SMD (or its user) needs such service, it may contact one nearby SMD which has the service and is willing to give/lend that. The SMD may provide the service on profit (gets something in return) or non-profit (doing social service) basis. To understand and visualise the usefulness of PMCC better the let us consider the following real application scenarios:

Scenario 1: Akash is on a group tour with his friends. He is on the train. Suddenly he receives an urgent email from his office. He is asked to complete the financial

calculations attached with the mail immediately. Obviously, Akash is not happy but also knows that he has to do it anyhow. As the job is computing-intensive, Akash faces the following two problems:

a) He has not brought his laptop with him and neither his friends. All of them are carrying only smartphones. But none of these devices is capable enough to carry out the computing-intensive calculations individually.
b) As he is in a moving train, there is no continuous internet connectivity. And so, he cannot use the cloud.

So, Akash decided to utilize PMCC and requests his friends to lend their SMDs' resources. The PMCC application on his smartphone divides the computation task into a number of small tasks and sends each small task to the SMDs of Akash's friends. Each SMD executes its part and sends the result back to Akash's smartphone. Once, all the results are accumulated, they are assembled to have the final result. Akash sends the result to his office and moves on to enjoy his holidays without any tension.

Scenario 2: John, while sitting in the student canteen, downloads a research article that seems to be related to his project. But after downloading, he finds that the document is not opening on his SMD as it is in some unknown format. John has a pdf reader application installed in his device, and if the document could be converted to pdf format, he could read it. But he does not have any such conversion app. So, he searches the network for a buddy, having the conversion service. And thank goodness, he found one such and got the file converted to pdf. John is elated and dives into the paper without delay.

Scenario 3: Lilly goes to a shopping mall to buy a laptop. The salesman shows her a couple of models from different companies. Though the salesman tries his best to explain to her about the features of the products, Lilly wants to be sure about the advantages and disadvantages, in a comparative manner, of the products the salesman showed her. Her SMD is not able to help her because she does not have any such benchmarking applications installed. She also tries to visit the websites of the companies of the products using her SMD, but due to the inferior cellular

connection, she cannot manage to open all the websites either. In a PMCC environment, she requests for a benchmarking service to a service provider (SP), in her vicinity. The SP agrees to help her and makes the comparison for her and sends her the comparison result. Satisfied with the best one (according to her requirements) she bought the laptop and went home happily.

Scenario 4: Merry intends to fill up the library subscription form while sitting in the college library. But the form is in pdf format, and she does not have the edit option in her pdf reader version (the pdf reader might be a pirated one or with a limited licence). So, she tries to convert it into a Word document. But again, she does not have the conversion software. So, she searches for an SP which can provide this conversion service within the network. Unfortunately, there was no SP available in the network. So, a service carrier (SC) comes to her rescue, which goes out and searches for the required service in other networks. Once it finds the service, gets from that SP, returns to the previous network, and delivers the converted Word document to Merry. But interestingly, Merry is not aware of all these processes. The PMCC system took care of all these, transparently.

8.4 UCSD Dataset

To implement our system for finding relative mobility, we used the UCSD trace dataset[42], obtained from the University of California, San Diego. Marvin McNett and Geoffrey M. Voelker of the University of California, San Diego conducted a study of the movements of as many as 275 students of the university for the period from 22nd September 2002 to 8th December 2002 – a total of 78 days [731]. The trace data was collected for 402 APs within a range of approximately 130 m X 130 m square area. In the trace collection, the users were given a Jornada PDA which recorded the trace information with the help of a data collection tool called WTD, an in-house developed software. The WTD software sampled and collected the trace data before uploading them periodically to a server. The users are identified through the MAC addresses of their devices' wireless network cards. It is assumed

[42] http://sysnet.ucsd.edu/wtd/data_download/wtd_data_release.tgz

that there is a one-to-one mapping between users and wireless network cards. However, the mapping is anonymous, i.e., there is no mapping between usernames and MAC addresses.

After a user's device was powered on, the WTD software collected the following data periodically during the 11-week trace period:

- The signal strength of each detected AP.
- MAC address of each AP that was detected.
- Current AP association.
- The version number of the WTD program.
- Type of device (Jornada 548 or 568).
- Power state the device is running on (connected to an AC socket or using battery power).

It is to be noted that the WTD program recorded the MAC address of all APs that were detected by the PDA, not only the AP the PDA was associated with at the time of sampling. This is evident from the recorded data contained in the trace file *wtd.csv* (described later in this section). A sampling of data was done with a time interval of 20 seconds, i.e., the WTD software recorded data after every 20 seconds. During data collection, when the local data file reached a critical size, the WTD software contacted the data collecting server and uploaded the file at the next opportunity. A feature of the WTD software was that it automatically checks the server for new updates and downloads and installs them, so that it may adapt to unexpected problems if necessary.

Key terms: The key terms used in the trace data analysis are as follows:

- **User:** A user is someone who has a mobile device connected to one of the AP in the context.
- **AP session:** An AP session is a contiguous time period for which a user's device was associated with a particular AP in the UCSD campus.
- **User session:** A user session is a contiguous time period in which a user's device is powered on and is able to detect the signals of nearby APs. A user

session includes the user's movements among different APs.

Contents: The dataset contains three files, namely, *ap_locations.csv*, *README*, and *wtd.csv*. The files *ap_locations.csv* and *wtd.csv* are trace files containing the data recorded by the UCSD researchers and *README*, a text file, is the metadata file of these two trace files. The files *ap_locations.csv and* wtd.csv contain comma-separated values comprising four and seven fields, respectively, as shown in Fig. 8.3 and Fig. 8.4, respectively. The trace file *wtd.csv* was processed to generate the set of files required for the mobility study.

AP_ID	Unique identifier assigned to the AP.
X_COORDINATE	X-coordinate (in ft) of AP w.r.t. campus coordinate system.
Y_COORDINATE	Y-coordinate (in ft) of AP w.r.t. campus coordinate system.
Z_COORDINATE	Z-coordinate (in ft) of AP w.r.t. campus coordinate system.

Fig. 8.3. Fields of ap_locations.csv and their descriptions

USER_ID	Unique identifier assigned to the user.
SAMPLE_DATE	The date the sample was taken by the WTD software.
SAMPE_TIME	The time the sample was taken by the WTD software.
AP_ID	Unique identifier assigned to the detected AP.
SIG_STRENGTH	Strength of AP signals received by the SMD.
AC_POWER	Whether the SMD used AC power (1) or battery (0).
ASSOCIATED	Whether the SMD was associated with this AP (1) or not (0).

Fig. 8.4. Fields of wtd.csv and their descriptions

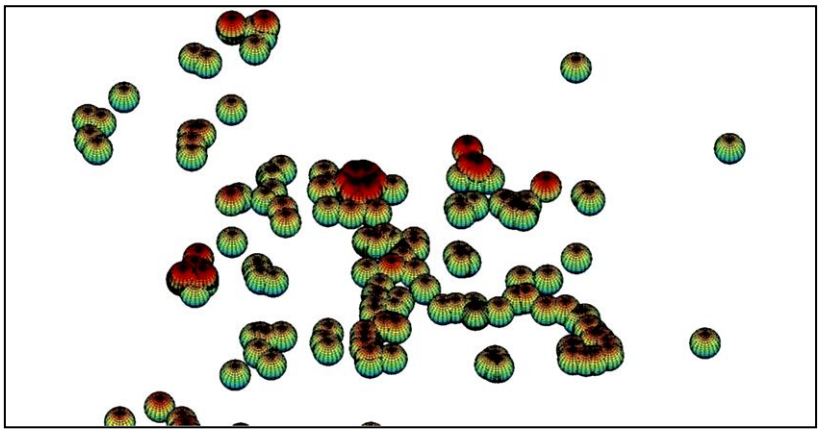


Fig. 8.5. Snapshot of the 3D view of APs

It is to be noted that the trace file *wtd.csv* and the metadata file *README*, both do

not mention the field SAMPLE_DATE, probably due to the human error or as per the researchers' discretion. A snapshot of the 3D view of APs are shown in Fig. 8.5, and the snapshots of the files *ap_locations.csv* and *wtd.csv* are shown in Fig. 8.6 and Fig. 8.7, respectively.

```
AP_ID,X_COORDINATE,Y_COORDINATE,Z_COORDINATE
9,1696306,259247,21
11,1697337,259292,8
12,1697773,261665,76
14,1697181,261181,21
15,1697716,260425,8
16,1698295,261388,60
17,1697800,261537,74
18,1697850,261510,76
19,1697632,260411,8
20,1696233,260940,3
23,1697849,260370,8
24,1696306,259298,21
26,1696398,260163,0
27,1697983,259999,8
28,1696547,261535,0
29,1697132,260955,0
30,1697432,260954,13
31,1695281,261754,8
32,1697513,260573,8
33,1696561,261193,21
```

Fig. 8.6. Snapshot of the file ap_locations.csv

```
USER_ID,SAMPLE_TIME,AP_ID,SIG_STRENGTH,AC_POWER,ASSOCIATED
123,09-22,00:00:00,359,8,0,0
123,09-22,00:00:00,363,5,0,0
123,09-22,00:00:00,365,11,0,1
191,09-22,00:00:00,355,31,0,1
101,09-22,00:00:00,353,8,1,0
101,09-22,00:00:00,362,30,1,1
129,09-22,00:00:00,369,31,0,1
156,09-22,00:00:00,360,19,1,1
184,09-22,00:00:02,352,29,0,1
184,09-22,00:00:02,364,12,0,0
211,09-22,00:00:02,366,31,1,1
211,09-22,00:00:02,369,18,1,0
35,09-22,00:00:05,353,31,1,1
154,09-22,00:00:05,362,23,0,1
66,09-22,00:00:06,363,25,1,1
66,09-22,00:00:06,364,4,1,0
149,09-22,00:00:07,354,7,0,0
149,09-22,00:00:07,355,10,0,0
149,09-22,00:00:07,360,29,0,1
108,09-22,00:00:07,368,31,1,1
260,09-22,00:00:07,361,11,0,0
238,09-22,00:00:08,352,5,1,0
238,09-22,00:00:08,364,21,1,1
```

Fig. 8.7. Snapshot of the file wtd.csv

8.5 Predicting Continuous Relative Stability in a Single-cluster PMCC

In this section, we present a relative stability prediction method for the mobile devices in a single-cluster PMCC.

8.5.1 Resource Availability Problem in PMCC

It should be ideal to assign the job to the SMD, which is supposed to be stable and would stay within the PMCC for a longer time (till the job is finished). In order to select one or multiple SMDs as stable resource providers, predictions about the future mobility behaviour of each of the participating nodes are crucial. Particularly, the following information are needed:

a) For how long the SMD is available in the PMCC cluster?

b) What is the mean/average period of availability?

c) What is the probability of the service requester and the provider being in contact until the request is fulfilled?

8.5.2 Relative Topological Stability

In many situations, it is observed that though a pair or a group of mobile users change their topological position, relatively they stay together, i.e., they generally move together. Fig. 8.8 depicts such a scenario. The figure shows the mobility traces of a number of SMD users (A, B, C, D, E, F, G, H, I, J, K, L, M, N, and O) from the APs AP_1 to AP_4 along with time t_1 to t_4. It can be seen from the figure that a group of users (C, G, I, J, and O) always move together. Which means, though their topological positions are changed they are relatively stable. This is the exact case in the scenario described in Section 8.3. Here, though Akash and his friends are in the mobile state, they are relatively static as they are moving together.

In these circumstances, i.e., when the resource seeker and provider move together, there is no need for handoff or switching of the resource provider. Here, mobility does not affect PMCC topology and operation. The primary concept of this approach is that even the SMDs are mobile, if they are relatively static, i.e., they move together, resource exchange can be done. Finding such a group of SMD users

enables us to select the resource providers which minimizes the chance of incomplete job execution and job reassignment in PMCC.

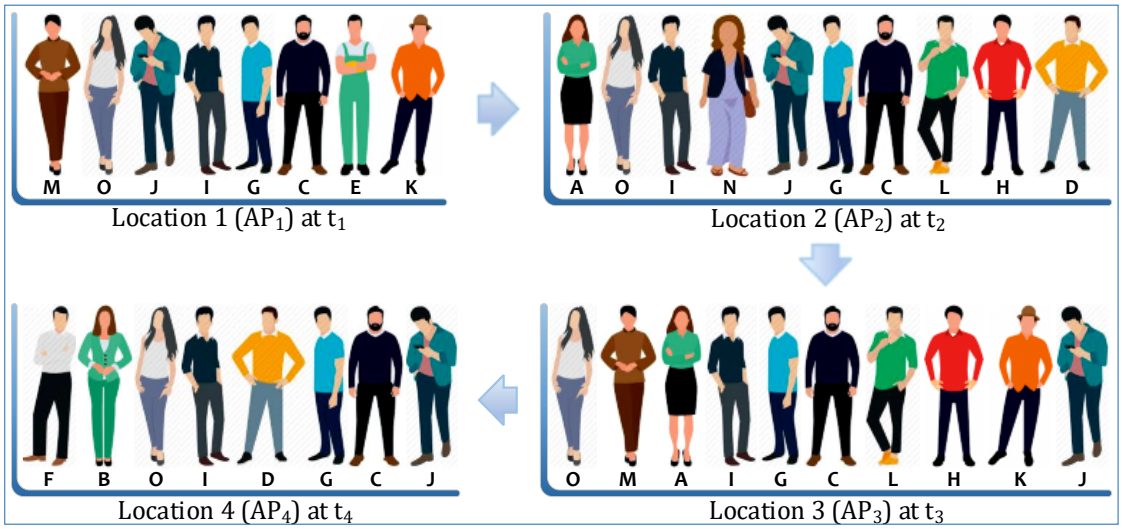


Fig. 8.8. Relative stability is maintained between a group of mobile users, although their topological positions are changed

The proposed mobility prediction algorithm should identify the users whose probability to be relatively static is higher, i.e., they stay together most often across different network segments and locations.

8.5.3 Experiment and Validation

In this section, we introduce and elaborate the concept of relative stability with details of experiment done. Results are analysed using statistical measures and validation techniques.

8.5.3.1 Calculating Relative Stability

The relative stability of an SMD is measured by its stability (or dynamism) with respect to its neighbourhood. Each SMD monitors its neighbourhood at different time intervals. Let, x denotes an SMD and NH be its set of neighbourhood SMDs. The mobility of an SMD at t_1 is calculated using Eq. 8.1, where $NH_{t_1}^x$ = list of neighbours of x at time t_1, $NH_{t_1-1}^x$ = list of neighbours of x at time $(t_1 - 1)$ and $M_{t_1}^x$ is the mobility of x at time t_1.

$$M_{t_1}^x = \frac{|NH_{t_1-1} \cap NH_{t_1}|}{|NH_{t_1}|} \tag{8.1}$$

The relative stability, over a time span T, is calculated using Eq. 8.2, where, $RS_{t_1}^x$ = relative stability of x at time t_1 which is stored for the last S_T time steps.

$$RS_{t_1}^{x} = \frac{1}{S_T}\sum_{i=t_1-x} M_{t_1}^{x} \tag{8.2}$$

The algorithm for relative stability calculation is given in Algorithm 8.1.

Algorithm 8.1: Relative Stability Calculation
Input: Node neighbour list database of x.
Output: Relative stability $\boldsymbol{RS}_{t_1}^{x}$ of x.
1. Find current neighbour list $\boldsymbol{NH}_{t_1}^{x}$. 2. Calculate the change in the neighbourhood using Eq. 8.1. 3. Store the obtained values in a linear list. 4. Calculate relative stability using Eq. 8.2. 5. Return relative stability $\boldsymbol{RS}_{t_1}^{x}$.

The mobility at a particular time t_1 is considered as a temporal process. The spatiotemporal prediction tool is not considered because the node mobility at a particular time instance t_1 depends on the neighbour list at $t_1 - 1$ and t_1, for i = 1 to n, where n is the number of intervals.

8.5.3.2 Short-term Relative Stability Assessment

This section describes our study of relative stability made on a short-term basis. The descriptions of the variables which were used in our analysis are given in Table 8.1. Here, A and N are partially dependent variables, depending on independent variables U, T and D, whereas S is a completely dependent variable, depending on N.

For computing the short-term mobility, the value of relative stability was calculated from 20 users selected randomly from the dataset, and the relationship between their mobility and the obtained relative stability was analyzed. The selected users had the following USER_ID's: 10, 12, 22, 30, 35, 18, 24, 2, 7, 70, 77, 98, 135, 142, 208, 255, 263, 270, 240, and 210.

Since the relative stability prediction algorithm requires information about the neighbourhood of an SMD for the calculation of its relative stability, the original trace file was processed to obtain the neighbourhood information of all users. Each day was divided into 96 intervals, each of 15 minutes duration. The connection information of each user during these intervals was recorded. The connection information answered the following two questions:

a) To which AP a particular user was connected during a particular 15-minute

interval on a particular day?

b) What is the set of neighbours of a particular SMD (user) during a particular 15-minute interval on a particular day?

For each user, a particular session of two-hours duration on a particular day during which he/she was connected to some AP was selected. The set of neighbours of a user U_i for a particular time interval was defined as the set of users, other than U_i itself, who were connected to the same AP as the user U_i during the particular time interval.

Table 8.1. Notations used in the relative stability analysis

Variable	Dependency	Description
U	Independent	USER_ID of users from trace dataset.
T	Independent	Time interval of 15-minute duration, where T=1 denotes 00:01 am – 00:15 am, T=2 denotes 00:16 am – 00:30 am, and so on. T lies in [1, 96].
D	Independent	Day of observation selected from the trace period, where D=1 denotes 22nd September 2002, D=2 denotes 23rd September 2002, and so on. D lies in [1, 78].
$A^{j}_{i,k}$	Partially dependent on U, T and D	AP_ID of the AP to which user with U=j is associated with during time interval T=i on day D=k; its value is "NC" if the user is not connected to any AP.
$N^{j}_{i,k}$	Partially dependent on U, T and D	Neighbourhood set of users with U=j during time interval T=i on day D=k; its value is "NC" if the user is not connected to any AP.
$S^{j}_{i_1,k_1,i_2,k_2}$	Dependent on N	Relative stability of user with U=j calculated over the time span: time interval T=i_1 on day D=k_1 to time interval T=i_2 on day D=k_2. R lies in [0, 1]. It is to be noted that if $k_1 = k_2$, $S^{j}_{i_1,k_1,i_2,k_2}$ can be written as $S^{j,k}_{i_1,i_2}$.

Table 8.2 shows the neighbourhood information of four sample users during busy hours (10 am to 5 pm) and lazy hours (5 pm to 10 am), on a weekday and a weekend, using the notation defined in Table 8.1. It is expected to have more SMDs in the busy hours while very few or nil in lazy hours.

The detailed information about the chosen session for a set of representative users and relative stability calculated over the session is given in Table 8.3, which also shows the relative stability of a set of representative users, calculated on a short-term basis.

Table 8.2. Sample user's information

Time interval (T)	Lazy/ busy hour	USER_ID (U)	Weekend neighbourhood (D=8)	Weekday neighbourhood (D=25)
00:01 – 00:15 (1)	Lazy	16	$N_{1,8}^{16} = \{\}$ (empty neighbourhood)	$N_{1,25}^{16} = \{3, 24, 127, 225\}$
12:45 – 13:00 (52)	Busy	16	$N_{52,8}^{16} = \{50, 64, 191, 235, 264\}$	$N_{52,25}^{16}$ = NC
00:01 – 00:15 (1)	Lazy	103	$N_{1,8}^{103}$ = NC	$N_{1,25}^{103} = \{165, 207, 238\}$
12:45 – 13:00 (52)	Busy	103	$N_{52,8}^{103} = \{171, 185, 193, 238\}$	$N_{52,25}^{103}$ = NC
01:00 – 01:15 (5)	Lazy	39	$N_{5,8}^{39} = \{32, 128, 164, 166, 220\}$	$N_{5,25}^{39}$ = NC
09:45 – 10:00 (40)	Lazy	39	$N_{40,8}^{39} = \{32, 164, 166, 208, 244\}$	$N_{40,25}^{39}$ = NC
01:00 – 01:15 (5)	Lazy	65	$N_{5,8}^{65} = \{235, 264\}$	$N_{5,25}^{65} = \{16, 47, 64, 212, 243\}$
09:45 – 10:00 (40)	Busy	65	$N_{40,8}^{65}$ = NC	$N_{40,25}^{65} = \{\}$ (empty neighbourhood)

Table 8.3. Selected session and relative stability information of a set of representative sample users

USER_ID	Day	Session (T_1–T_8)	Relative stability	Mobility/stability behaviour
10	18	1 – 8	$S_{1,8}^{10,18} = 0.8333$	Static; high relative stability
12	13	2 – 9	$S_{2,9}^{12,13} = 0.95$	Static; high relative stability
22	31	36 – 43	$S_{36,43}^{22,31} = 0.25$	Dynamic; low relative stability
30	2	7 – 14	$S_{7,14}^{30,2} = 0.875$	Static; high relative stability
35	32	61 – 68	$S_{61,68}^{35,32} = 0$	Dynamic; low relative stability

It is observed that the users who have a higher value of relative stability are relatively more static compared to the users who have a lower value of relative stability. By relatively static, we mean that a user is static with respect to its neighbours, irrespective of its actual geographical location. Detailed analysis reveals that the users who have relative stability value close to '1' have unchanging AP connection, whereas the users who have relative stability value close to '0' have frequently changing AP connection. From Fig. 8.9, we intuitively choose 0.5 as the threshold value for determining relative stability of the users, separating them into two groups, namely static users and dynamic users. A user whose relative stability is above 0.5 is regarded as static, and one whose relative stability is below this threshold value is regarded as dynamic.

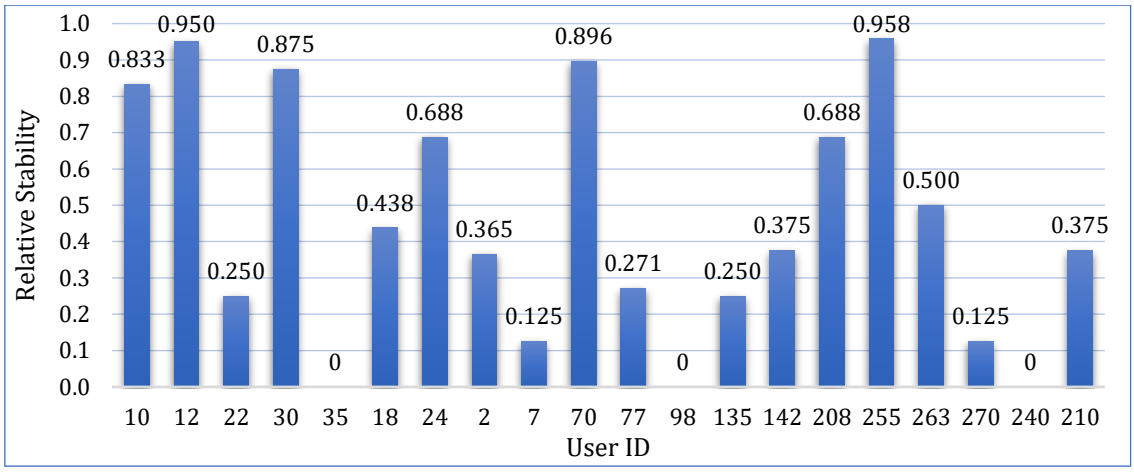


Fig. 8.9. Relative stability of 20 sample users

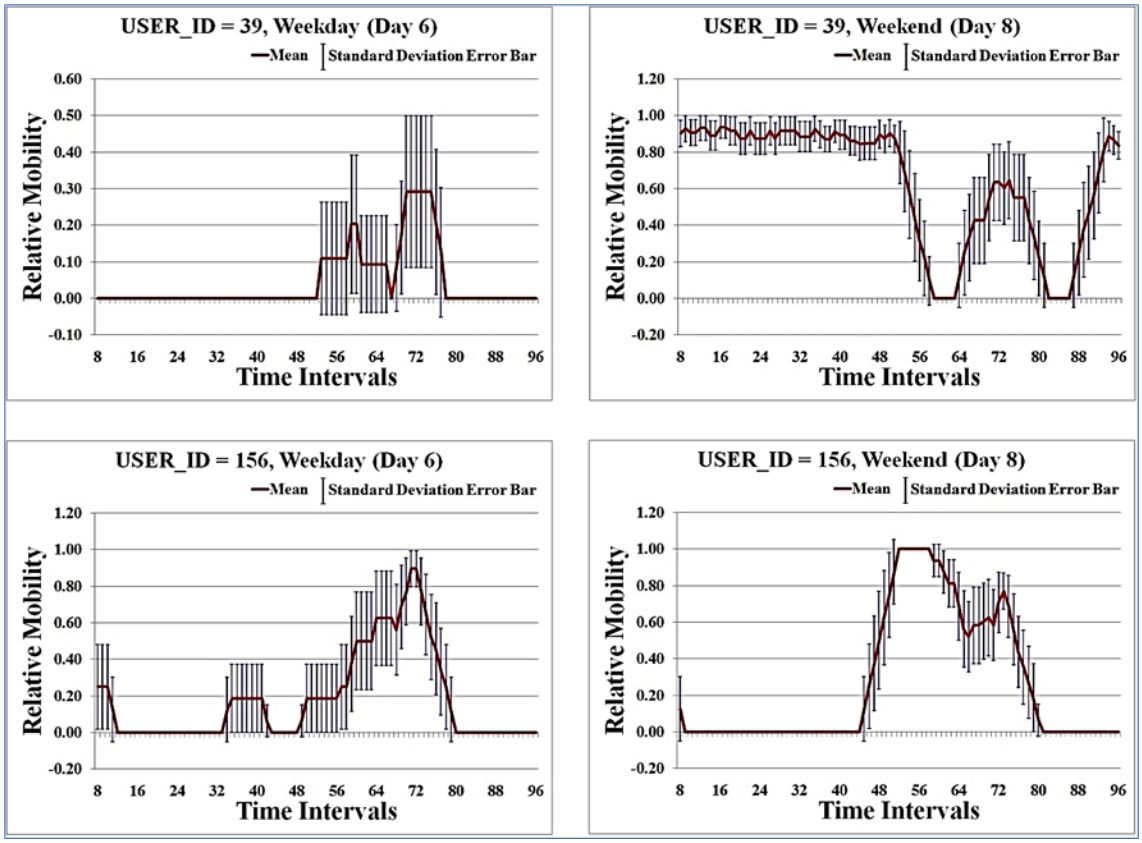


Fig. 8.10. Relative stability/mobility analysis for two users (USER_ID's 39 and 156) on weekday and weekend

Furthermore, a study of the relative stability of two sample users (USER_ID's 39 and 156) on a weekday and a weekend in terms of the mean and standard deviation of the neighbourhood change (according to Eq. 8.1) was performed; the related graphs are shown in Fig. 8.10. The mean and standard deviation values are calculated by taking spans of two hours (eight 15-minute intervals) throughout the day. The mean and standard deviation values corresponding to an interval t are the mean and standard deviation of the neighbourhood change values over a time span

t' to t, where $t' = t - 7$. Observing the weekday and weekend graphs of the two users, we come across different scenarios, as mentioned in Fig. 8.11.

It is to be noted that a user getting disconnected (going out of range of all APs/turning off his or her SMD) is regarded as a neighbourhood change, and the neighbourhood change value at the disconnection interval is assumed to be zero.

From the above observations, it is concluded that the short-term mobility study is not sufficient to determine a user's inherent behaviour, the reason being that when the standard deviation value becomes greater than the mean value of the neighbourhood change over a given timespan, the user's behaviour is unpredictable, even if it exhibits a high value of relative stability. To determine the inherent mobility behaviour of users, a logistic regression analysis of their relative stability for the entire trace period is needed, as described in the next section.

Case 1 *Stability Ratio < Standard Deviation and Stability Ratio < Threshold Value*

- The user's behaviour is unpredictable.
- There is no fixed group of neighbours with which the user stays connected.
- For example, such behaviour is exhibited by user 39 during intervals 66 to 73 on day 6.

Case 2 *Stability Ratio < Standard Deviation and Stability Ratio >= Threshold Value*

- The user behaviour is erratic.
- The user stays connected to a fixed group of neighbours for some intervals but it changes its neighbourhood at other intervals.
- For example, such behaviour is exhibited by user 156 during intervals 53 to 60 on day 6.

Case 3 *Stability Ratio > Standard Deviation and Stability Ratio < Threshold Value*

- The user's behaviour is predictable and it is dynamic.
- The user changes its neighbourhood in a similar pattern leading to an almost constant low value of neighbourhood change during the timespan.
- For example, such behaviour is exhibited by user 156 during intervals 69 to 76 on day 8.

Case 4 *Stability Ratio > Standard Deviation and Stability Ratio >= Threshold Value*

- The user's behaviour is predictable and it is static.
- The user stays connected with almost the same fixed group of neighbours throughout the timespan leading to an almost constant high value of neighbourhood change. Such a user can be regarded as static.
- For example, such behaviour is exhibited by user 156 during intervals 56 to 63 on day 8.

Fig. 8.11. Different possible scenarios based on Fig. 8.10

8.5.3.3 Long-term Mobility Prediction Using Logistic Regression Analysis

A long-term study of mobility behaviour of all users is necessary to determine the effectiveness of our proposed prediction algorithm when users' neighbourhood records are taken over a long period of time. It indicates the degree of correctness

of our method when a large database of users' movements is analyzed. If both short-term and long-term prediction results are consistent for a particular user, the user can be predicted to be static or dynamic with a high chance of the prediction being correct.

The proposed algorithm was applied on the UCSD dataset and the average value of relative stability, termed as average stability ratio (ASR), was calculated for all the users in the dataset over the entire trace period of 78 days (taking the average of 78 daily values of relative stability for each user), after which a logistic regression model was derived to predict whether a user is static or dynamic. Out of the 275 users in the trace, 240 users were used for the analysis (the other users were filtered out as they remained active in zero or very few sessions during the entire trace period, leading to an insignificant value of the standard deviation of neighbourhood change). This set of users was divided into two sets for training and testing. Records of the first 160 users were used as the training set to derive the logistic regression model, and of the remaining 80 users were used as the test set to validate the model.

Before performing logistic regression, we needed a method to classify a user as static or dynamic. We took the dependent variable as static or dynamic and one independent variable of ASR. The descriptions of variables are summarised in Table 8.4 and Table 8.5. There is no definite method for defining the status of a node as static or dynamic. We used a method that is simple and objective, i.e., if the coefficient of variation (CV), defined by Eq. 8.3, of the relative stability of a user is less than 1, it is classified as *Static*, and otherwise, it is classified as *Dynamic*.

$$CV = \frac{Standard\ deviation}{Average\ stability\ ratio} \tag{8.3}$$

As the dependent variable or outcome is a dichotomous one, we have taken *Static* = 0 and *Dynamic* = 1 to classify the nodes, as shown in Table 8.4.

8.5.3.4 Results and Analysis

In this section, we analyse the results using logistic regression and empirical tests. We also observe the classification accuracy. Furthermore, the model's performance is validated through goodness fit. We also compare the performance of the model

for short- and long-term relative stability predictions.

Table 8.4. Dependent variable based on the coefficient of variation

Degree of relative stability	Type of user based on ASR	Criterion	Internal value
High relative stability	Static	CV < 1	0
Low relative stability	Dynamic	CV > 1	1

Table 8.5. Independent variable

Variable name	Description
Avg_SR	ASR of a user, calculated for the full trace period of 78 days

8.5.3.4.1 Logistic Regression Analysis

Logistic regression [732] [733] has popularly been used for predicting the outcome of an event, or the probability of the presence or absence of a characteristic in a considered context is based on the values of a set of predictor variables. Precisely, it is the most popular modelling approach for binary outcomes. Since, in our prediction requirement, there are only two possible outcomes, either static or dynamic, the logistic regression is the automatic choice for prediction modelling. The basic characteristics of logistic regression are shown in Fig. 8.12.

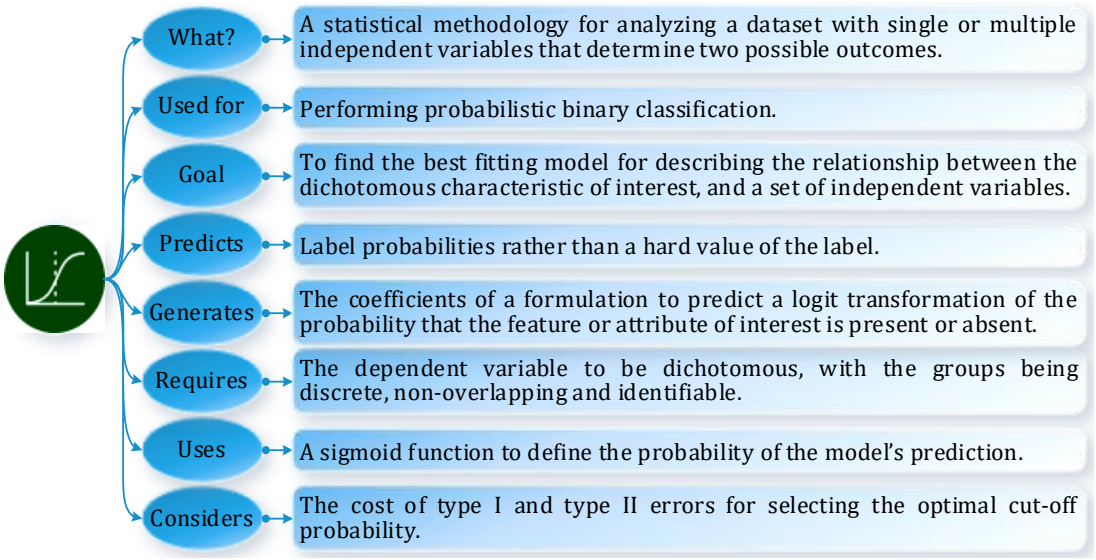


Fig. 8.12. Basic characteristics of logistic regression

Compared to the linear regression model, it is more appropriate to model the situation where the dependent variable is dichotomous (a 0/1 variable). In a linear regression model, the input of the dependent variables might be greater than 1 or less than 0. Therefore, it is impractical to model probabilities with linear regression since the boundary of the probability of an event should be within 0 to 1. To overcome this, the logistic regression model adopts a generalized form of the linear

model and extends it by mapping the real numbers to the 0-1 range [734].

The relationship between the regression coefficient (z) and the probability (p) of an event of interest (i), in a logistic regression model, is described by the link function, given in Eq. 8.4. Here, the dependent variable (z_i) is a logit, which is a log of odds, in other words, it is the value of the unobserved continuous variable for the i^{th} case, and p_i is the probability that the i^{th} case experiences the event of interest and z_i and is defined by Eq. 8.5.

$$logit(p_i) = z_i = ln\ \frac{p_i}{1-p_i} \tag{8.4}$$

$$p_i = \frac{e^{logit(p_i)}}{1+e^{logit(p_i)}} = \frac{e^{zi}}{1+e^{zi}} = \frac{1}{1+e^{-zi}} \tag{8.5}$$

In Fig. 8.13, the predicted variable p_i and the explanatory variable z_i are represented through the vertical and horizontal axes, respectively. The odds ratio (z value) of each independent variable in the logistic regression model can be estimated using the regression coefficients, as given in Eq. 8.6.

$$logit(p_i) = z_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} \tag{8.6}$$

where, x_{ij} is the j^{th} predictor for the i^{th} case, β_j is the j^{th} coefficient, and k is the number of predictors.

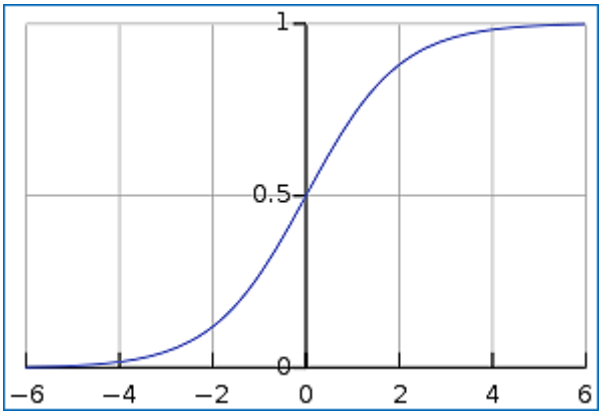


Fig. 8.13. Link function p_i

Here, βs are the regression coefficients that are evaluated by means of an iterative maximum likelihood method. Logistic regression analysis does not consider the prior probabilities of failure as well as the restrictive assumptions regarding the normal distribution of independent variables or the equal dispersion matrices [735] [736]. Though, due to the subjectivity of the choice of these misclassification costs

in practice, most researchers aim to minimize the total error rate, implicitly assuming equal costs of type I and type II errors [736] [737].

8.5.3.4.2 Empirical Testing

To test the proposed model, we opted for the Wald statistic that is used to assess the significance or contribution of the coefficients in the model [738]. It is defined by Eq. 8.7.

$$Significance = \frac{(Regression\ coefficient)^2}{(Standard\ error\ of\ the\ coefficient)^2} \tag{8.7}$$

Wald statistic is based on Chi-square distribution with degree of freedom (DF) is 1; i.e., the number of independent parameters is one. The parameter for which the Wald statistic is maximum is the best one that contributes to predicting the outcome properly. Maximization of Wald statistic means the minimization of the standard error of the corresponding parameter.

The estimated results of the logistic regression model for the ASR of the users in the total sample are shown in Table 8.6. The maximum likelihood estimation for user classification is used to estimate the final logistic regression, as expressed in Eq. 8.8, where, z is defined as in Eq. 8.6. Here, the event (i) is *Static*; hence, p_i denotes the probability of the outcome *Static*.

$$z = -6.455 + 21.293 \times ASR \tag{8.8}$$

Table 8.6. Estimated results of the logistic regression model of the ASR of the mobile users

	Coefficient of the predictor (β)	Standard error (SE)	Wald $((\beta/SE)^2)$	Degree of freedom	Significance
ASR	21.293	3.453	38.019	1	.000
Constant	-6.455	.981	43.251	1	.000

8.5.3.4.3 Classification Accuracy

The accuracy of the proposed model has been diagnosed using classification table (also known as confusion matrix), as shown in Table 8.7, which gives a statistical estimation of the number of observations that have been predicted correctly. Classification tables help to evaluate the correctness of the model by cross tabulating the actually observed successes with the predicted number of successes and actually observed failures with a predicted number of failures. For each case, if the predicted probability of a category is greater than the user-specified cut-off, the

predicted response of that category is considered as 0. The cut-off value was considered as 0.5.

Table 8.7. Classification table of the proposed model

Observed performance		**Predicted performance**	
		Static	*Dynamic*
		Positive prediction	Negative prediction
Static	Positive observation	TN	FP
Dynamic	Negative observation	FN	TP

8.5.3.4.3.1 Classification of Training Dataset

Table 8.8 shows the comparison of the observed and the predicted mobility behaviour of the users and to the extent that it can be correctly predicted on the training dataset. Dynamic users can be classified 95.4% correctly, while 78.4% of Static users can be properly classified. The overall prediction is 90.0% correct. The cut-off value can be changed to notice the change/improvement in the predicted correct response.

Table 8.8. Classification table for training dataset

Observed performance	**Predicted performance**		**Correct prediction percentage**
	Static	*Dynamic*	
Static	40	11	78.4
Dynamic	5	104	95.4
Overall correct prediction percentage			90.0

8.5.3.4.3.2 Classification of Evaluation Dataset

Table 8.9 shows the comparison of the observed and the predicted mobility behaviour of the users and to the extent that it can be correctly predicted on the evaluation dataset. *Dynamic* users were classified 95.6% correctly, while 88.6% of *Static* users were predicted accurately. The overall prediction is 92.5% correct.

Table 8.9. Classification table for evaluation dataset

Observed performance	**Predicted performance**		**Correct prediction percentage**
	Static	*Dynamic*	
Static	31	4	88.6
Dynamic	2	43	95.6
Overall correct prediction percentage			92.5

8.5.3.4.4 Validation

In model validation, the prediction performance of the model is observed by checking against independent data. Typically, the validation process includes collecting an independent dataset and validation of the results on it. To validate our model,

as shown in Table 8.9, we took 80 test users, and the results were validated using various evaluation metrics, as listed in Table 8.10. The performance measures of the prediction model are compared for the training and evaluation datasets and shown in Fig. 8.14.

Table 8.10. Performance measure details for prediction model evaluation

Evaluation metrics	Significance	Ideal value	Calculation
True positive rate (TPR) or sensitivity or recall	Indicates how many static SMDs, out of all the static SMDs, have been predicted correctly.	Close to 1	TP/(TP+FN)
True negative rate (TNR) or specificity	Indicates how many dynamic SMDs, out of all the dynamic SMDs, have been predicted correctly.	Close to 1	TN/(TN+FP)
False positive rate (FPR) or fall-out (Type-I error)	Indicates how many dynamic SMDs, out of all the dynamic SMDs, have been predicted incorrectly.	Close to 0	FP/(TN+FP)
False negative rate (FNR) or miss rate (Type-II error)	Indicates how many static SMDs, out of all the static SMDs, have been predicted incorrectly.	Close to 0	FN/(TP+FN)
Positive predictive precision (PPP) or positive predictive value (PPV)	Indicates how many SMDs, out of all the predicted static SMDs, are actually static.	Close to 1	TP/(TP+FP)
Negative predictive precision (NPP) or negative predictive value (NPV)	Indicates how many SMDs, out of all the predicted dynamic SMDs, are actually dynamic.	Close to 1	TN/(TN+FN)
False discovery rate (FDR)	Indicates how many SMDs, out of all the predicted static SMDs, are actually dynamic. In other words, it is the probability of making any Type-I error at all.	Close to 0	FP/(FP+TP)
False omission rate (FOR)	Indicates how many SMDs, out of all the predicted dynamic SMDs, are actually static. In other words, it is the probability of making any Type-II error at all.	Close to 0	FN/(FN+TN)
Accuracy (ACC)	Determines the overall predicted accuracy of the model. In other words, it measures the fitment of the model.	Close to 1	(TN +TP)/ (TN+FN+TP+FP)
Balanced accuracy (BA)	Gives a balanced accuracy calculation for the imbalanced test set, like ours. It is calculated as the average of TPR and TNR for each class (here, static and dynamic).	Close to 1	(TP/(TP+FN)+ TN/(TN+FP))/2
F1-score	Provides a better understanding of the performance of the model than ACC, especially in case of an uneven class distribution, like ours. It is calculated as the harmonic mean of TPR and PPP.	Close to 1	2TP/ (2TP+FP+FN)

As the validation approach we followed the tests for goodness of fit. The goodness of fit asserts how well a statistical model and the sample data fit to a set of observations. It helps to identify if there is any discrepancy between observed values and

the values predicted from the considered model in a normal distribution case. Although several methods are there for testing the goodness of fit of the logistic regression, in this work, we used the popular Hosmer and Lemeshow test (HL test) [735] that provides useful information about the calibration of the model. If the p-value produced by the HL test for goodness of fit is small, the model is considered as a poor fit. Typically, if it is less than 0.05, then the model is rejected; otherwise, it is passed.

The HL test is fundamentally a chi-square goodness of fit test for grouped data, where the samples are divided into g groups according to their predicted probabilities. Typically, the value of g (number of groups) is chosen as 10. The observations with the lowest 10% predicted probabilities are placed in the first group, the next lowest 10% are placed in the second group, and so on. The grouping is done on the estimated parameter values (z_i), for the probability that stability is predicted correctly for each observation in the sample, based on each observation's covariate values, as defined by Eq. 8.4.

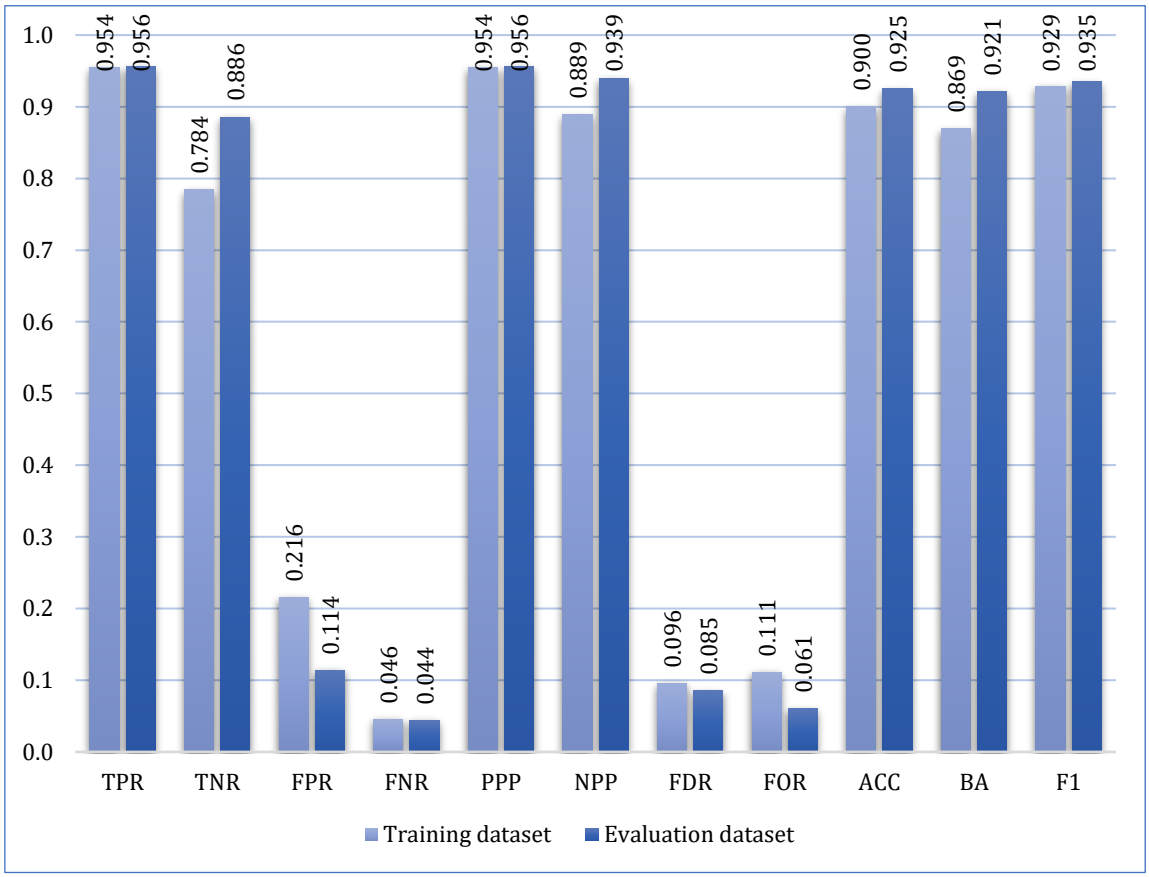


Fig. 8.14. Performance measures for training and evaluation datasets

Chi-square statistic is calculated by HL test is given by Eq. 8.9.

$$X_{HL}^2 = \sum_{j=1}^{10} \frac{(O_j - P_j)^2}{P_j(1 - P_j/n_j)} \sim X_8^2 \tag{8.9}$$

where, n_j = number of observations in the j^{th} group; O_j = number of observed cases in the j^{th} group; P_j = number of predicted cases in the j^{th} group; X^2 = standard Chi-square.

The degree of freedom and the p-value are calculated using Eq. 8.10 and Eq. 8.11, respectively, where k is the number of regrouping (i.e., 10 in HL test) of the observed and predicted cases of two dependent variables.

$$DF = k - 2 \tag{8.10}$$

$$p\text{-}value = \int_0^{11.053} f(X^2)\, d(X^2) \tag{8.11}$$

The observed significance level for Chi-square value is found to be 0.199, as shown in Table 8.11, which indicates that the null hypothesis of the model is acceptable, i.e., the observed values are close to the predicted values. The acceptance of the null hypothesis ensures that the hypothetical model fits the observations reasonably satisfactory. The calculated Chi-square value of this model at the level of significance below 0.05 indicates that logistic regression is very meaningful and our null hypothesis for goodness of fit, i.e., observed mobility are equal to the predicted mobility tested by the HL test.

Table 8.11. Hosmer and Lemeshow test results

Chi-square	DF	p-value
11.053	8	.199

8.5.3.4.5 Comparison of Short- and Long-term Relative Stability

Previously, the instantaneous relative stability of 20 sample users from the trace dataset was calculated and analysed. The relative stability of two additional users throughout a particular weekday and weekend was analysed as well. Later, the overall ASR of all users (240 users out of 275 users of the entire trace dataset) was calculated, and a logistic regression model was derived to predict the mobility of the users in terms of *Static* or *Dynamic*. The significance of the logistic regression analysis is that it shows us the inherent nature of the users. The short-term relative

stability and its analysis may show a result different from the overall nature of a particular user. For an exemplar demonstration, Table 8.12 summarises the analysis information of two users.

Table 8.12. Analysis of two sample users

USER_ID	Instantaneous relative stability	Behaviour (by intuition)	Classification according to logistic regression model	Remarks
24	0.6875 (day: 61, intervals: 42-49)	Static	Dynamic	Mobility cannot be predicted convincingly; behaviour is misleading.
255	0.9583 (day: 45, intervals: 29-36)	Static	Static	Mobility can be convincingly predicted because of consistent behaviour.

It is observed from the table that the user with USER_ID 24 has a relative stability value of 0.6875 on the 61st day of the trace period during intervals 42-49, giving us the impression that it is nearly static. Logistic regression analysis classifies it into the category of *Dynamic*. Due to this difference in the two results, the exact tendency (to be static or dynamic) of the user in the 50th interval cannot be determined convincingly. On the other hand, the user with USER_ID 255 has a relative stability value of 0.9583 on the 45th day of the trace period during intervals 29-36. Logistic regression analysis shows that it is indeed static on a long-term basis, classifying it as static analysing its ASR. This user can be convincingly predicted as static in the 37th interval. For users that satisfy a consistent behaviour with both the instantaneous and regression analysis, i.e., if their results in both the analysis match, prediction about their stability can be made convincingly.

8.6 Predicting Discrete Relative Stability in a Multi-cluster P2P MCC

In this section, we present a service provisioning scheme in a multi-cluster PMCC by predicting the relative stability of the service seekers and the service providers. Since this approach involves latency, it particularly suits the delay-tolerant applications, which may afford to wait for the required service.

8.6.1 System Model

In this section, we present the system model of a multi-cluster PMCC and the details of the proposed service provisioning scheme.

8.6.1.1 Key Components

Before diving into the details, let us be familiar with the key components of the proposed system. Some of the crucial terms we encounter throughout the article are as follows:

Service seeker (SS): An SMD (or SMD user) which needs a service that it does not have.

Service provider (SP): An SMD which has the service that is requested by a SS and it (or the mobile phone user) is willing to lend/provide the service.

Service carrier (SC): An SMD which, if required, carries the request (with necessary details) of an SS to an SP and gets back the result/service to the SS.

Access point (AP): Usually, a Wi-Fi router to which the SMDs are connected. Also, each AP forms an individual cluster comprising the SMDs connected to it. The APs are not connected to each other.

Reference node (RN): It is one of the most resourceful and highly stable SMDs in the cluster. It acts as the leader of the concerned cluster and is responsible for keeping the information of all the peers within the cluster so that when it receives a service request form an SS, it can rightly suggest the most suitable SP or SC to the SS, regarding services that could be provided by the SMDs in the network. Under each AP, at any point of time, an RN should be present.

8.6.1.2 Proposed Service Provisioning Scheme

The working of the proposed system is stepwise described in the following:

1. As mentioned earlier, under each AP, an RN is maintained, which holds the information of all the peer SMDs in the same network.
2. When needed, an SMD refers to the concerned RN for probable SP.
3. If the requested service is available within the own network (AP) then:
    i. RN suggests SS the most suitable SP.
    ii. SS requests the SP for the service.
    iii. SP provides the service to the SS.

This scenario is depicted in Fig. 8.15.

4. If the service is not available in the network, the RN suggests a suitable SC under it that has a higher probability of moving to another network where the service may be available and come back soon. If the SC is willing to do this job, then:
    i. The SC takes along the request with the required data/inputs to another network and requests the RN of that network (remote RN) to suggest the most suitable SP (remote SP) for the job.
    ii. As per the suggestion from the remote RN, the SC requests the suggested SP for the desired service.
    iii. SC gets the service, returns to the previous AP (home network), and hands over the service to the SS.

This scenario is depicted in Fig. 8.16.

5. If the service is not found in the other network also, any one of the following measures can be adopted:
    i. The searching cycle might go on in other networks.
    ii. SS can try later to get the service.
    iii. The request is aborted.

The workflow of the whole system is presented in Fig. 8.17.

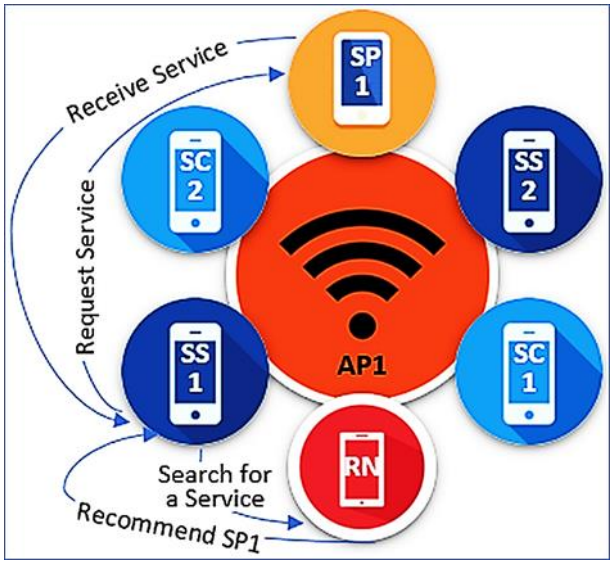
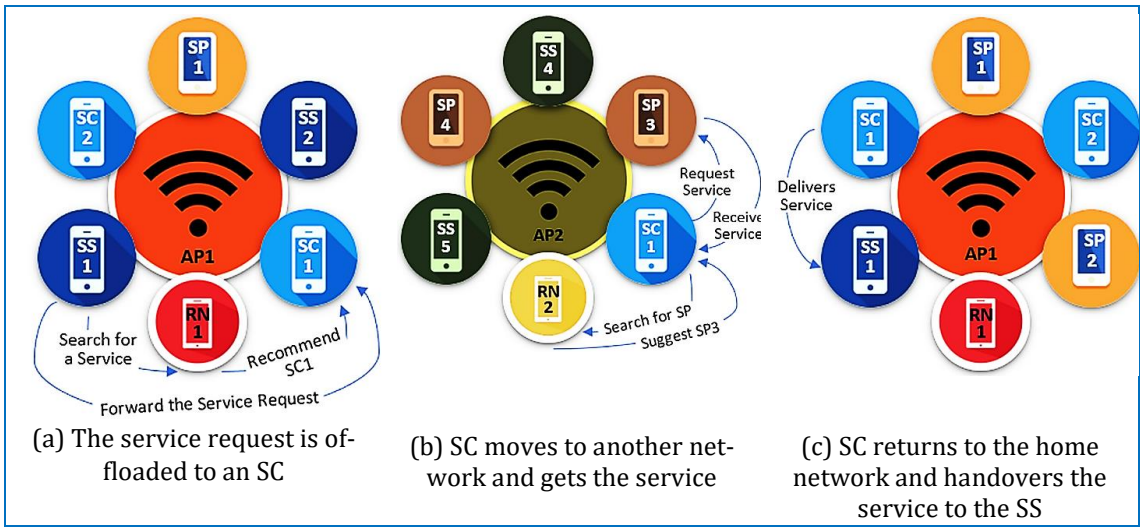


Fig. 8.15. Service is available within the network



(a) The service request is offloaded to an SC

(b) SC moves to another network and gets the service

(c) SC returns to the home network and handovers the service to the SS

Fig. 8.16. Service is not available within the network

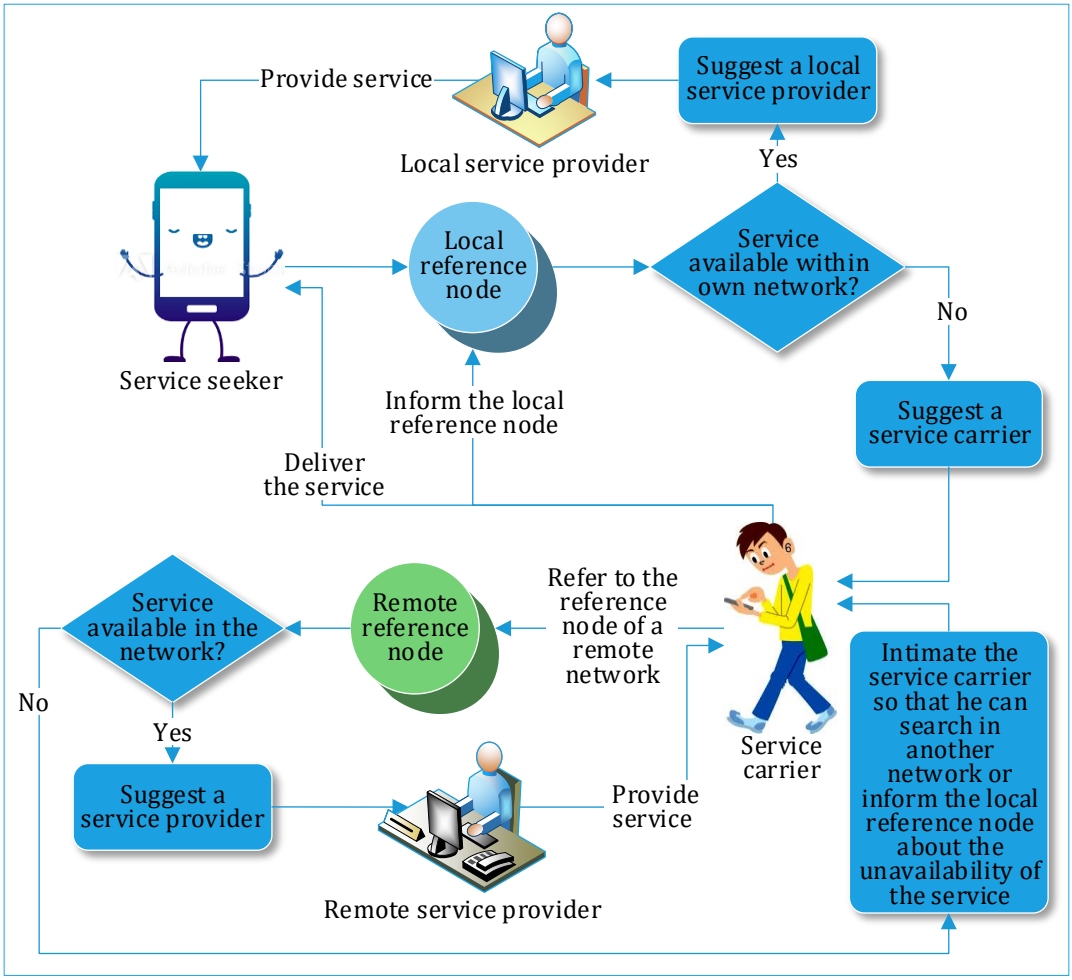


Fig. 8.17. The workflow of the proposed service provisioning system

8.6.1.3 Issues and Challenges in the Proposed System

Some of the following mentioned issues are crucial and need to be addressed for successful implementation of the proposed system for service provisioning:

Synchronisation between SS and SC: Since the nodes in an MCC are typically

dynamic in nature, it is crucial to synchronise the presence of SS and SC in the same network. The absence of which may result in the following significant problems (also depicted in Fig. 8.18):

- **Liveness problem:** The SS assigns the job of getting the service from a different network to SC (Fig. 8.18(a)). But SC may not come back with the service to the SS network, as illustrated in Fig. 8.18(b). It is treated as liveness problem. The liveness problem can be avoided if an SMD is selected as SC that has high mobility.
- **Availability problem:** On the contrary, it may happen that the SC returns with the service but finds that SS is not present in the network, as illustrated in Fig. 8.18(c). To avoid this availability problem, we require a mechanism that makes synchronization between the SS and SC and so that SS remains in the network when SC comes back with the service.

Selecting a node with high mobility: We just mentioned that a highly mobile SC could help in avoiding the liveness problem. Also, an SC with high mobility has a better chance to get a service in different networks at different times. Because, if the service is not available in one network, a highly mobile SMD can move to other networks and find the service. That implies, the search for service is widely propagated across several networks. But identifying the node that is highly mobile is not straight forward.

(a) SS1 requests SC1 to get the service

(b) SS1 waits for SC1 to come back with the service, but SC1 does not return within a finite time period (the carrier is not live)

(c) When SC1 returns with the service, SS1 is not present (unavailable) in the network

Fig. 8.18. Availability and Liveness problems

Calculating the probability and the probable time of meeting SS with SC again: In addition to being highly mobile, an SC should be selected based on the contact frequency with the SS. This requires calculating the number of times the SC comes in contact with the SS.

Calculating the overall contact period of SC with the SP: It is very crucial to know the approximate time for which the SP will remain in contact with the SC because if the stay-time of the SP in the network (while providing the service) is less than the amount of time required for serving the request, the purpose is not fulfilled. So, the contact period between SP and SC should be larger than that of service providing time.

Deciding on the optimal waiting time of SS for SC to bring the service: The SS cannot wait indefinitely for the SC to get the service for it. This might result in an indefinite lock in for a certain service. So, there should be a timeout value for SS as well as SC to wait or search for the particular service.

Permissible time an SC can spend in other networks while carrying a pending service request: When carrying a service request form an SS an SC cannot stay in other networks for a very long or very small duration. If SC stays for an abnormally long time outside of the network, the SS may starve for the service, and if the SC stays in the external network (from it is supposed to get the service) for a very small duration, it may not get enough time to receive the service. Therefore, finding out the optimum time duration an SC can stay in another network is very important.

Weighing the reliability of SC: How to be sure that the designated SC will surely get the service? Due to several reasons (intentional or circumstantial, SC may not deliver the service to the SS, even after agreeing to do that. Though exact assessing is not trivial, an approximation should be made on the reliability of SC in getting the service assuredly. To tackle this problem, the SC can be incentivised so that they will find interest in delivering the service to SS successfully.

8.6.1.4 Essential Criteria of the Key Components

From the above discussions, now we are able to get a clear depiction of the

desirable properties of the key components mentioned in Section 8.6.1.1.

SS: Ideally, should be stable in the network, so that whenever SC returns to the network, it should be available to receive the service.

SC: Since SC plays a crucial role in getting the requested service while selecting an SMD as an SC, the following crucial factors should be considered:

- A node with high mobility should be chosen as an SC since the highly mobile node has a high frequency of changing its position and comes in contact with different networks and thus, having a high probability that it will search the service providing node faster.
- In spite of the high mobility, the SC should regularly be in contact with the SS for service delivery.
- The SC should be present in the network of the SP for sufficient duration so that it can receive the service fully and successfully.
- The SC should have sufficient memory to carry the service request and the service itself.

SP: Ideally should be stable in the network. If not, then, it should be stable at least for the contact period with SC so that the service can be provided completely to SC.

RN: The RN is also an important entity which should have the following characteristics:

- Ideally, should be very much stable in the network.
- It should be active enough to retrieve the information of the SMDs whenever they enter the network.
- To store the information of all other SMDs, the RN must have sufficient hardware and software resources.
- It should have adequate decision-making capability for suggesting SP and SC.

8.6.2 Experiment and Validation

In the experiment, our goal is to obtain the followings:

a) Average time (in seconds) after which a user connects to an AP.

b) Average duration (in seconds) a user remains connected to an AP.

c) A set of four users who are simultaneously connected a particular AP.

Above-mentioned first (a) and second (b) types of information are useful to find out the movement of the users that at which time the user will come again in contact with the AP and how much time a user spends. And the third information will help to find the expected future location (AP) of a group of users, i.e., when they will be in contact with that particular AP.

To attain the above-mentioned goals, we performed the following operations on the original *wtd.csv* file in steps:

1. Eliminate all the users with associated value '0' and create a separate file which includes all the users with associated value '1'.
2. Find out the average time gap (in seconds) after which a user comes back and reconnects to the AP, he/she was connected to earlier. We calculate separate average time connection per AP for every user in the network and name these files on the name of users, for example, the file of user with ID '1' is saved as *1.avg.txt*.
3. Find out at least four users who are connected to a particular AP at the same time. For example, four users - user '32', user '54', user '97' and user '21' are connected to AP '245' at time 13:12:54, 13:12:57, 13:12:46, and 13:13:00 respectively. Here, the maximum time difference is 14 second, which is not much, hence ignored; i.e., we consider that they are connected at the same time frame.
4. To find such combinations, we write a program in Python and identify a set of four users who are connected at the same time with a particular AP.
5. Calculate average connection time (ACT) (the average time a user is connected to a particular AP) of the users.

The details of each operation are discussed in the following subsections.

8.6.2.1 Data Preprocessing

First, we filter according to the ASSOCIATED value available from the downloaded *wtd.csv* file. This value is used to calculate/identify the neighbours of a node (user).

To make it clear, suppose three nodes at a time are in the range of an AP but the first node is only connected to this AP and the second and the third nodes are not connected or connected to some other AP. Then, they cannot be treated as neighbours of each other. To eliminate entries with ASSOCIATED value 'o' (i.e., not connected), we write a program in Java which use MySQL in the backend to access the file. Snapshot of the output is shown in Fig. 8.19.

8.6.2.2 Calculate the Time Gap After a User Returns to the Network Again

After doing away with the unwanted values (nodes that are not connected), we aim to estimate the average time period after which a user comes back to the contact (ASSOCIATED) with the particular AP. A Python program was written for this, which considered the details of the connected nodes (users) during the data tracing period, i.e., for 78 days. Fig. 8.20 gives an idea of the average time after which a user gets connected to a particular AP. This is important to get rid of the availability problem. Based on this information, the SMD that is highly mobile and has a high possibility of meeting with the SS after getting the service from the SP is selected as SC.

```
USER_ID,SAMPLE_DATE,SAMPLE_TIME,AP_ID,SIG_STRENGTH,AC_POWER,ASSOCIATED
191,22-Sep,0:00:00,355,31,0,1
156,22-Sep,0:00:00,360,19,1,1
101,22-Sep,0:00:00,362,30,1,1
123,22-Sep,0:00:00,365,11,0,1
129,22-Sep,0:00:00,369,31,0,1
184,22-Sep,0:00:02,352,29,0,1
211,22-Sep,0:00:02,366,31,1,1
35,22-Sep,0:00:05,353,31,1,1
154,22-Sep,0:00:05,362,23,0,1
66,22-Sep,0:00:06,363,25,1,1
149,22-Sep,0:00:07,360,29,0,1
108,22-Sep,0:00:07,368,31,1,1
238,22-Sep,0:00:08,364,21,1,1
68,22-Sep,0:00:09,360,29,1,1
172,22-Sep,0:00:11,3,31,1,1
194,22-Sep,0:00:11,360,26,0,1
89,22-Sep,0:00:12,39,10,0,1
185,22-Sep,0:00:14,364,10,1,1
187,22-Sep,0:00:15,365,31,0,1
167,22-Sep,0:00:16,53,12,0,1
147,22-Sep,0:00:16,360,3,1,1
147,22-Sep,0:00:16,360,4,1,1
```

Fig. 8.19. Snapshot of the dataset after eliminating not connected entities

AP_ID	AVG(in sec)
354	1177.28571429
356	564.153846154
357	8623.93478261
70	210.090909091
364	1258.27272727
363	3421.375
236	153.8
174	653.8
50	616.75
435	357.0
119	53.6666666667
90	41.3333333333
446	1959.5
63	122.3

Fig. 8.20. Snapshot of ACT of a particular users for a number of APs

8.6.2.3 Identifying a Group of SMDs Connected to an AP Simultaneously

To continue our experiment, we find a group of four users who are connected to an AP at the same time. An instance of such set is shown in Table 8.13. It can be observed that a group of four users are connected to AP '354' on 24-sept between 13:00:51 to 13:01:08.

Table 8.13. Sample group of four users

USER_ID	SAMPLE_DATE	SAMPLE_TIME	AP_ID	ASSOCIATED	ACT (in secs.)
208	24-sept	13:00:51	354	1	180
242	24-sept	13:00:59	354	1	40
41	24-sept	13:01:03	354	1	1481
232	24-sept	13:01:08	354	1	2493

8.6.2.4 Selecting the Reference Node

From Table 8.13, we select an SMD as an RN. To find the RN, we calculated the ACT of the SMDs, which indicate how long an SMD is available in the network and how many occasions and for how long they are disconnected. This is important for a node to active in the network to provide or access services.

We divided a day into 12 sessions of 2 hours each and track the movement of the SMDs for each session. For example, the first session of 2 hours' starts from 00:00:00 to 02:00:00, and we calculate ACT of User 42, User 208, User 232, and User 242 in this time interval. Similarly, the ACTs of these users are collected for all sessions, as shown in Table 8.14. Using these ACT, we calculate average time and standard deviation.

The mean (μ) of ACT, collected for different sessions, and the standard deviation (σ) were calculated using Eq. 8.12 and Eq. 8.13, respectively, where s, x_i and N

denote no. of sessions, values, and total no. of values, respectively.

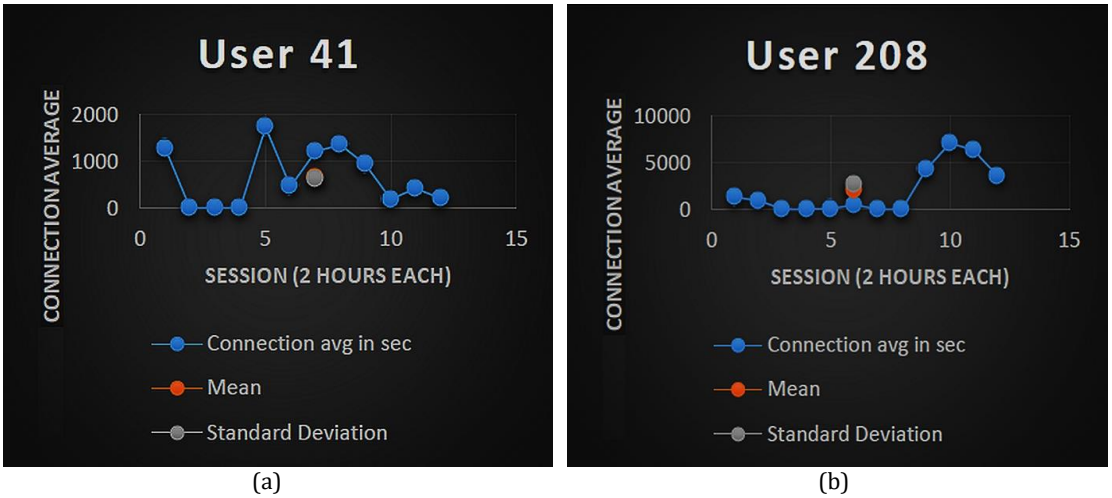
$$\mu = \frac{1}{N}\sum_{i=1}^{s} x_i \tag{8.12}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{8.13}$$

Using the mean value and the standard deviation, we spot the user (SMD) that stays in the network for most of the time. Within the considered four users, we find out that User 41 is more frequently available in the network. For comparative demonstration, ACT graphs of User 41 and User 208 are shown in Fig. 8.21.

Table 8.14. ACT of four users for a particular AP

Session	Connection time (in secs.)			
	User 41	User 208	User 232	User 242
1	1280	1260	0	0
2	0	930	0	0
3	0	0	0	0
4	0	0	0	0
5	1730	80	20	0
6	453.33	455	1760	0
7	1213.33	0	0	0
8	1345	0	0	0
9	933.33	4220	0	0
10	180	7060	0	0
11	410	6320	0	0
12	200	3540	0	0



(a) (b)

Fig. 8.21. Average connection time graph of (a) User 41 and (b) User 208

From Fig. 8.21(a), it can be observed that the ACT of User 41 is close to its mean and standard deviation, which shows that there is less variation in the ACT. Whereas, in Fig. 8.21(b), we can see that there is a large difference between

standard deviation and ACT of User 208, and the variation of ACT is more than User 41. So, we consider User 41 as a better choice to be an RN.

8.6.2.5 Calculating Relative Stability

Another decisive criterion for choosing an SMD as an RN is relative stability (RS). To calculate the RS of each user, we used the algorithm proposed in [739], as shown in Eq. 8.14. The RSs of the four users in consideration are shown in Table 8.15.

$$s_t^x = \frac{1}{s}\sum_{i=t-x}^{t} m_t^x \tag{8.14}$$

where, $s_t^x \Rightarrow$ relative stability, $m_t^x \Rightarrow$ current neighbour list and is defined by Eq. 8.15.

$$m_t^x = \frac{|N_{t-1} \cap N_t|}{|N_t|} \tag{8.15}$$

Table 8.15. The relative stability of four users

Time interval	Relative stability			
	User 41	User 208	User 232	User 242
8	0.666667	0.125	0	0
16	0.416667	0.45833333	0	0
24	0.375	0	0	0
32	1	0	0	0
40	0.479167	0.0833333	0	0
48	0	0.2833333	0	0
56	0.15625	0.125	0	0
64	0	0.0833333	0	0
72	0	0	0.3563	0
80	0	0	0.25	0
88	0.041667	0	0	0
96	0.59375	0	0	0

The RS graphs for User 41, User 208, User 232, and User 242, obtained from Table 8.15, are shown in Fig. 8.22.

From Fig. 8.22(a), it can be observed that the RS for User 41 is varying very close to the standard deviation and mean, which are almost equal. In the case of User 208 also, the RS is varying around standard deviation, as can be seen in Fig. 8.22(b). But the difference between mean and standard deviation is much higher for User 232, and this user is not active/relatively stable for most of the time, as can be seen in Fig. 8.22(c), hence, not suitable as the RN. User 242 is not active at this duration (Fig. 8.22(d)). So, comparing these 4 graphs, we found that User 41 and User 208 are the two choices to be selected as the RN. But the ACT of User 41 is better than

User 208; therefore, we select User 41 as the RN.

Fig. 8.22. Relative stability graph of (a) User 41 (b) User 208 (c) User 232 (d) User 242

8.6.2.6 Calculate the Latency of the Users in Returning to the AP

Now, we estimate when a user returns to the previous AP after it left. It is important because if a SS leave the AP, it should come back as soon as the service is available which is brought by an SC. Similarly, for an SC also, it is crucial to return with the service as soon as possible. To calculate the return time, we use the ACT of the users. For this calculation, we used Table 8.16, which includes all the values required to calculate. An instance of the four considered users is shown in the table. It can be seen that the SCs (User 208 and User 242) returns to the network after 0.085 days.

Fig. 8.23 shows the arrival time of the four users when they return after leaving the A; in this case, AP 354. User 41 and User 232 are SSs, which require services and User 208 and User 242 are SCs, which are mobile in nature and supposed to bring

service to the requesting node.

Table 8.16. Date and time of future connection at AP 354

SS	SC	Time after which SC returns in the network	Predicted next contact day and time (approx.)	SCs connect with the SSs (connected/not connected)
User 41, User 232	User 208, User 242	0.085 days	24-Sept 13:30:00	Yes

The estimated time of returning of User 208 and User 242 to the AP is at 13:30:00, as shown in Table 8.16. It is assumed that they bring the service along with them. Hence, at that time, the SSs, i.e., User 41 and User 232, should be present in the network to receive the requested services.

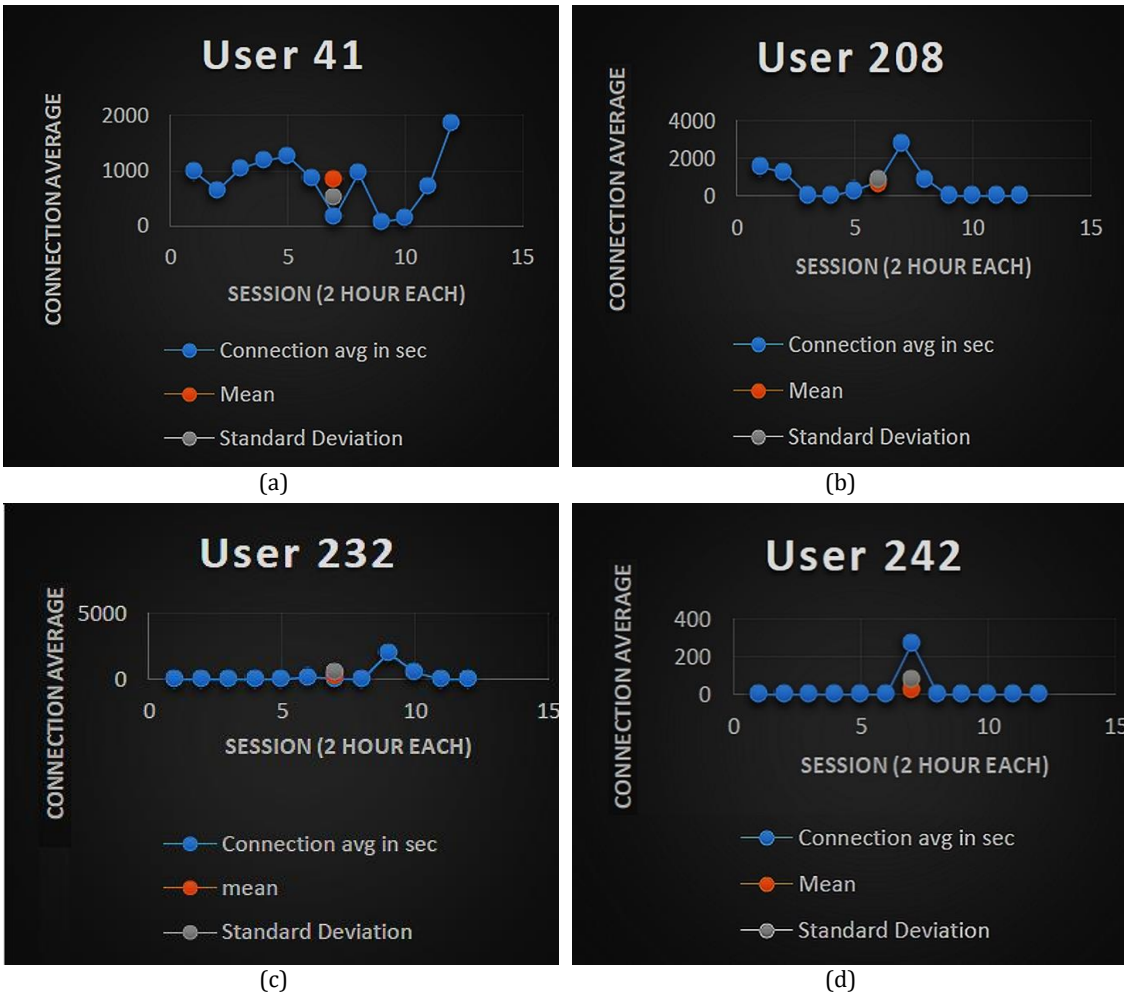


Fig. 8.23. Predicted arrival time of (a) User 41 (b) User 208 (c) User 232 (d) User 242

The standard deviation is used to measure variation between the ACT of the sessions. From Fig. 8.23, it is evident that though the users seem to be roaming around the campus across the sessions, they all are connected to the particular AP (354) between session 6 and 7, which indicates time duration 12:00:00 to 14:00:00, and

this is in range of our predicted time (13:30:00).

8.6.2.7 Validation

Now let us check how our proposed approach for estimating the relative mobility works in a different scenario. Let us consider that User 41 and User 208 are the SSs (which request for services that are not available in the network) and User 232 and User 242 are the SCs. The estimated time of returning of User 232 and User 242 to the AP is at 16:00:00 on 25-Sept, as shown in Table 8.17.

Table 8.17. Future date and time of connection

SS	SC	Time after which SC returns in the network	Contact day and time (approximate)	SCs connect with the SSs (connected/not connected)
User 41, User 208	User 232, User 242	1.183	25-Sept 16:00:00	No (User 41 and User 208, both are not connected with AP 354)

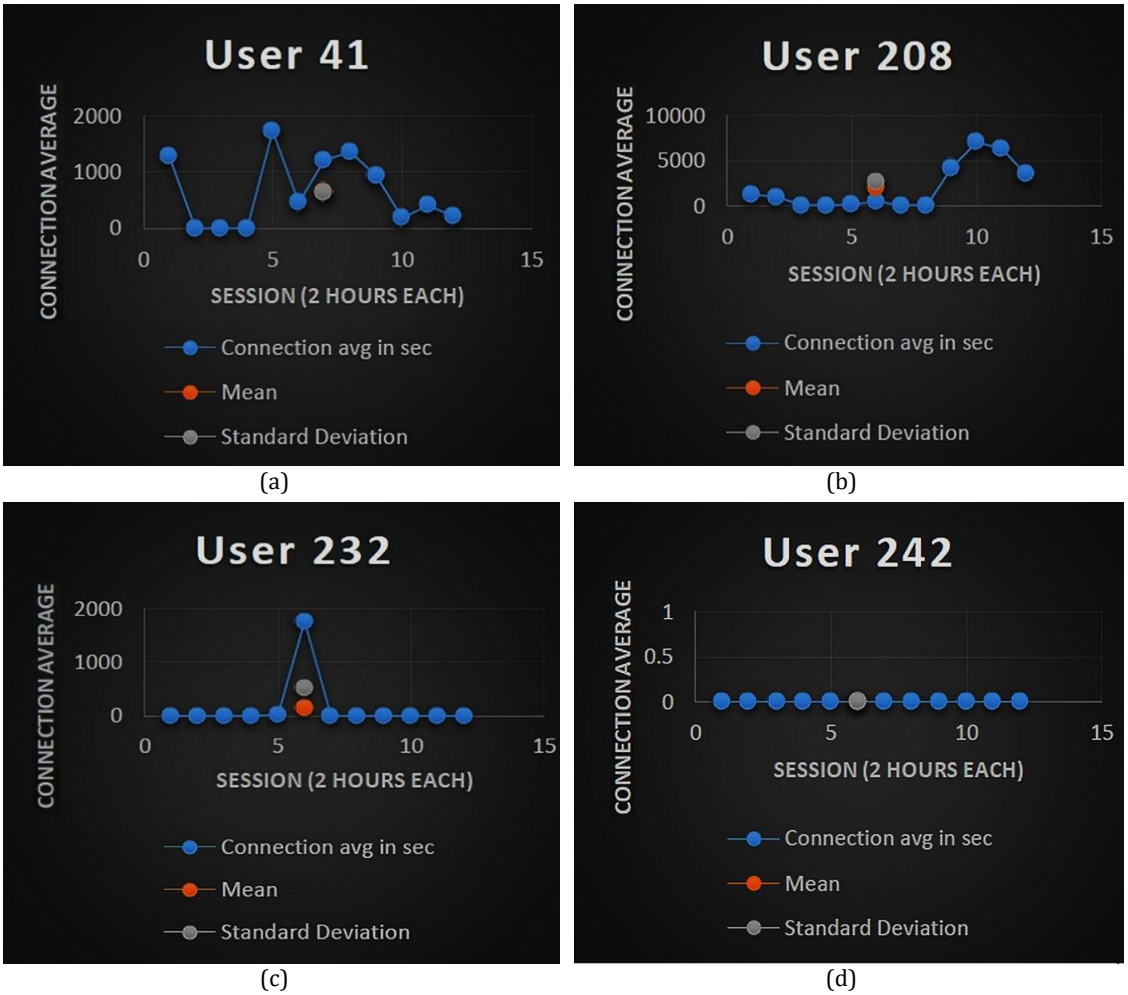


Fig. 8.24. Nodes did not arrive on the predicted time

Now, let us check if the SCs can deliver the service to the SSs at the time of contact. As we have seen, according to our estimation, the predicted time of contact is 25-Sept at 16:00:00 pm (approx.), which belongs to session 8 (14:00:00-16:00:00). From Fig. 8.24, we can see that at that time, User 208 and User 41 are not active; and User 242 is not active on 25 Sept, whereas User 232 is active on that session but connected to some other AP which are not in the range of the SS. So, SSs are not able to get the requested service from the SCs on the predicted time.

In the first case (Fig. 8.23), users arrived at the predicted location on time, but in the second case (Fig. 8.24), the same does not happen. So, using these two scenarios, we can conclude that to predict the future time on which SS and SC may communicate, a fixed pattern is required.

8.7 Limitations and Further Scopes

The results show that the proposed algorithm can be used to efficiently predict the behaviour of the nodes in a PMCC. However, it does not make any prediction about how long a node in the network is going to remain static once it has been predicted to be static. If a node is found to be static at a particular location, its duration of stay at that location is a criterion that should be considered while making the decision about selecting a stable SMD in PMCC. Further research will be carried out to predict the duration of stay of a static node to bring more improvement in the mechanisms of service management in PMCC.

Another shortcoming of the experiments is that the dataset that is used in the experiment is not wide enough; it is only of 78 days. And also, the traces do not follow a fixed pattern whereas our proposed model is based on a fixed mobility pattern because, in real life, most people do their task in a fixed pattern like they wake up in the morning, take breakfast, go to their work, stay at the workplace till evening, return back to their home and, generally, follow the same pattern, except holidays. So, experimenting with a dataset of fixed mobility patterns would have yielded a better prediction.

8.8 Summary

In this chapter, we presented the concept of P2P MCC where mobile devices share

resources and services between themselves. Being a local system, connected through short-range communication, for effective implementation, the service consumers and the providers need to be together (connected) continuously or at least periodically. In this regard, we introduced relative stability that defines the probability of two mobile devices being topologically in sync. We considered two PMCC scenarios – one comprising only a single cluster of SMDs and another of multiple clusters.

In the first case, we found the mobility pattern of a group of mobile users, connected to different Wi-Fi APs, from the UCSD dataset, created using real user traces. Two studies were conducted on the dataset using the proposed algorithm. The first study applied the algorithm to calculate relative stability for 20 sample users from the trace, choosing time spans of 2-hour duration. These values gave the instantaneous behaviour of the nodes. In the second study, the average relative stability of a user was calculated taking its neighbourhood information during a trace period of 78 days. A logistic regression analysis of the average stability of all users from the trace was performed. The regression model was derived using a training set of 160 users and was evaluated using a test set of 80 users.

The general observation is that the short-term grouping does not necessarily reflect the true behaviour in the long term. It is observed that the result of the instantaneous analysis and that of the regression analysis, which takes into account the overall mobility of users for a long time period, do not match in case of some users. These users can be considered to be erratic in their movements on the campus in consideration, and therefore their behaviour is unpredictable. Based on their relative stability, the users were classified into two categories: static and dynamic. If a user's instantaneous relative stability and its classification derived by the regression model indicate similar behaviour, its mobility, i.e., whether it is static or dynamic, can be predicted with a high probability of accuracy.

In the second case, we presented an inter-cluster service provisioning scheme where a service can be availed from another cluster in case the service is not available in the base cluster. The mismatch in the contact time between the service consumer, provider, and the carrier (that acts as a mediator to exchange the service

between the provider and the consumer) will prohibit in exchanging the service. Hence, it is very important to know when they will be in contact with each other so that the service can be exchanged. We applied a mobility prediction algorithm on a dataset of real mobility traces to extract different mobility-related information. These information are used to select the service carrier and the provider so that the availability (consumer waits for the service indefinitely) and liveness (the carrier returns with the service but cannot find the consumer) problems can be averted.

9

MCC as Edge Computing: A Proof-of-Concept

"Anything that won't sell, I don't want to invent. Its sale is proof of utility, and utility is success." --- Thomas A. Edison

9.1 Introduction

In Chapter 1, we discussed the environmental effects of increasing computing demands. If we do not check on our device consumption model and carry on at the current pace, we might have to face serious consequences [20]. On the other side, the global warming has led to an increase of average temperature. Not only that, the erratic behaviour of the environment has caused unusual snowing and chilling at many places in the world [2]. This has triggered the use of HVAC systems worldwide in a huge scale. Going along the smart building and smart city trends the traditional HVAC systems also have transformed as smart systems. Typically, these smart systems generate huge amount of data that need to be processed quickly for taking swift action. This requires sufficient computing and networking resources which has further aggravated the ICT and environment scenario.

In this chapter, we present a proof-of-concept of using MCC as a sustainable and feasible edge computing infrastructure that is generally used for real-time computations.

9.1.1 Edge Computing

Cloud computing has revolutionised the landscape of scalable high-performance computing by offering utility computing resources on demand [740]. The pay-as-you-go policy of computing resources at affordable prices made cloud computing an instant hit among not only the small/mid/large-scale organisations but also individual users. This ubiquitous accessibility of elastic computing resources has paved the way for numerous innovative consumer-centric networked applications [741]. However, the latency involved in availing the cloud services, which is extremely crucial for real-time applications, has led the researchers to think about provisions of in-network computing resources [637]. In this approach, the

computing services are deployed at and accessed from somewhere in between the cloud and the user. One such solution is known as edge computing, where a set of small-scale computing resources is deployed at the edge of the external network [74]. Edge computing has let minimising the response time and the overall latency significantly compared to the cloud [636] [742]. Considering various application scenarios and infrastructural establishments, different forms of edge computing have been proposed [637]. Among them, the most prominent edge architectures are cloudlet [743] [744], fog computing [745] [746], and MEC [747] [748].

9.1.2 Crowdsourced Edge Computing

Considering the large-scale implementation of IoT, location and context-aware services, and virtual and augmented reality supported applications, edge computing will be demanded almost everywhere. But deploying the edge infrastructure on this vast scale is immensely challenging in terms of financial and ecological costs, coordination, and management. One worthwhile solution is to attain crowdsourced edge computing by utilising the nearby idle computing nodes owned by home users, organisations, or internet and mobile service providers as local computing endpoints [749]. Especially, in light of the emergence of DIY networking [750], in which SMDs and SBCs are leveraged to form ad-hoc networks quickly and efficiently, crowdsourced edge computing is certainly practicable with great potential [152] and is believed to democratize the computation [751].

9.1.3 Edge Computing through MCC

Thanks to the availability of enough SMDs almost everywhere, an MCC-powered edge computing (MCC-edge) can be a unique opportunity to realise crowdsourced edge computing. In an MCC-edge, public-owned SMDs can be leveraged to materialise an ad-hoc edge computing system ubiquitously. MCC-edge would be truly suitable for the time-constrained real-time applications as the computing nodes are in very close proximity to the data sources and as well as data sinks. The advantages of MCC-edge over other edge computing approaches are listed in Table 9.1. Considering the potential of MCC-edge, proper designing and deployment of it can be a crucial catalyst in realisation of smart cities and smart infrastructures.

Table 9.1. Advantages of MCC-edge over other edge computing approaches

Criteria	Cloudlet	Fog	MEC	MCC-edge
Dedicated infrastructure	Yes	Yes	Yes	No
HPC	No	No	No	Depends on the number of SMDs
Computing nodes per site	Single	Multiple	Single	Many
Elasticity	No	No	No	Yes
Dependency on corporate service	Yes	Yes	Yes	No
Upfront investment	High	Moderate	High	Negligible
Operational and maintenance cost	High	High	High	Negligible
Environment-friendly	No	No	No	Yes
System development	Generic	Generic	Generic	Tailormade
Responsibility	Control and compute	Control and compute	Control and compute	Only compute

9.1.4 Smart HVAC and MCC-Edge

The HVAC (heating, ventilation, and cooling) system is responsible for heating, cooling, and continuing air flow in the building, maintaining the ideal comfort level of the occupants in terms of temperature, moisture, and fresh air [12]. An ideal comfort level not only enhances the living of the occupants but also tittivates their functioning and behaving. In a commercial building scenario, it has been observed that the ambient environment and comfort level have a great impact on the efficiency and output of the employees. Considering the advantages, HVAC systems have become popular and have been an integral part of modern buildings.

9.1.4.1 HVAC: A Major Energy Consumer

HVAC is the largest energy consumer of modern buildings. Generally, HVAC systems account for nearly 40% of total electricity consumption either in residential or office buildings. Fig. 9.1 shows the energy consumption of the commercial buildings and HVAC, particularly. The continuous increase in global average temperature, as well as increasing population, are leading to more and more use of AC systems [752].

9.1.4.2 Smart HVAC: Reducing the Energy Consumption

Sensor-connected smart HVACs not only reduce the energy consumption substantially by optimising the configurations and operations but also maximise the user experience by adapting to the ambient conditions as and when required. For

example, if the occupancy sensor senses that there is no one in the room, the heater or the cooler and the lights can be turned off automatically. Similarly, if a particular area of the floor experiences a sudden steep increase in occupancy, the ventilation system should inflow more fresh air immediately. Besides, sensor-enriched HVACs reduce facility staff's onsite visits by sending an auto alert for equipment damage, inoperability condition, efficiency dropping, replacement requirements, etc. This not only lowers the maintenance costs but also improves operating life and energy efficiency. Data collected from the sensors attached to different HVAC components can give the facility workers and managers real-time insight into the proper functioning. The key benefits of smart HVACs are summarised in Fig. 9.2.

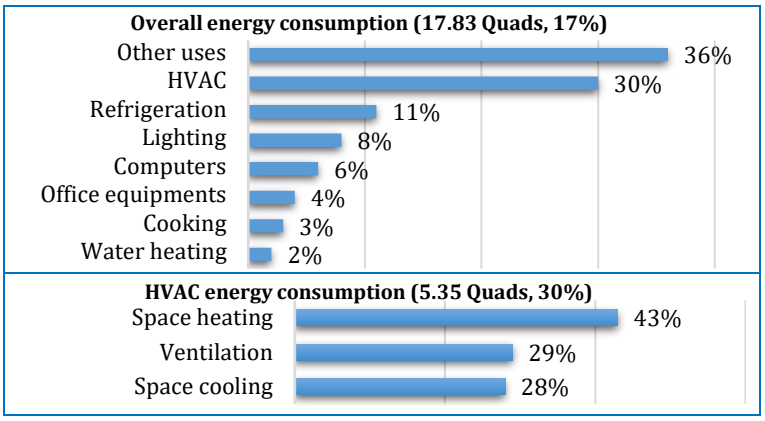


Fig. 9.1. Energy consumption in commercial buildings [753]

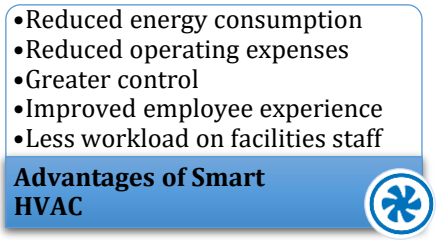


Fig. 9.2. Major advantages of a smart HVAC system in the context of an office building

9.1.4.3 Edge Computing for Smart HVAC

Depending on the building size, there might be hundreds to thousands of HVAC data points from which sensory data are captured and collected. And, depending on the data capturing frequency, annual data generation might be in terabytes to exabytes scale. To process, HVAC data of a standard mid-size commercial building, significant computing infrastructure is needed. It can be achieved through the cloud; otherwise, the building authority should have its own infrastructure. In practice, the first option is more sensible. But the following issues negate the

advantages of the cloud:

i. Cloud inherently suffers from high latency. Sending the data to the cloud and getting the results require some time which might not be suitable for non-delay-tolerant applications.

ii. To send the huge amount of data generated by different components of HVAC to the cloud requires significant bandwidth. Transmitting redundant or excessive data incurs network and energy costs.

iii. Cloud data centres consume huge energy, causing significant carbon emissions. It is estimated that by 2025, the data centre will share 25% of total global energy consumption [39]. And, they already account for nearly 2% of total global greenhouse gas emissions.

To handle the above-mentioned issues, a local processing facility nearer to the data source is required. This is known as edge computing [30], which can be advantageous in the following ways:

i. Processing smart HVAC data locally to generate real-time alarms for the events that require immediate attention. The typical round-trip times of the order of tens of milliseconds for cloud servers can be reduced to less than 10 ms by using a nearby server, deployed in the room or within the building premise [743] [754].

ii. Preprocessing smart HVAC data to reduce the data volume before sending it to the cloud for further analysis such as prediction, pattern finding, etc. Preprocessing at the edge will not only reduce the traffic but also lessen the load on cloud servers which in effect may lead in declining the requirement of the number of servers at the data centre. This will help the environment greatly.

In summary, local computing lets more in-hand information, intelligence, and local controls for HVAC devices and systems, which allows instant anomaly detection and quicker response while dwindling the environmental hazards. In [755] and [756], the significant differences between local processing and cloud processing in terms of time and energy are argued.

9.1.4.4 MCC-Edge for Smart HVAC

Usually, HVAC systems are installed in residential and office buildings where there is a high probability of a number SMDs being available in the building premises. Appropriate and viable policies can be framed to employ these personal devices in the on-premise infrastructure-less edge computing system. This local MCC-edge in the building might be dedicated to handling the HVAC operations, or it might be general-purpose, i.e., used for other building operations (e.g., analysing security camera data). The smart HVAC data would be sent to the SMDs, which would process those data as per pre-set instructions and return the results for actionable control of the HVAC in real-time.

9.1.5 Chapter Objective

In this chapter, we aim to achieve the followings:

- Demonstrate the viability and usability of MCC as a sustainable alternative to conventional vendor-based proprietary edge computing solutions.
- Design a MCC framework for real-time processing of the HVAC data generated from various sensors
- Maintaining the ideal comfort level in the room by using automated AC control and fault detection by processing the sensor data.
- Compare MCC-edge with commercial edge and cloud computing services in terms of cost, energy consumption, latency and environmental impacts.

9.2 Considerations for the Proof-of-Concept

In this section, we elaborate the concept of MCC, the considered HVAC scenario and the idea of MCC-edge enabled smart HVAC. A proof-of-concept typically entails several considerations to build up the initial prototype. Such considerations for the proposed system are also declared. For the proof-of-concept, we considered a local MCC with a client/server model, as described in Chapter 4.

9.2.1 Use Case Scenario

To demonstrate the viability of MCC as an edge computing solution, we considered a smart HVAC scenario. HVAC systems can be of two types: a) self-contained individual units or b) centralised systems. Fig. 9.3 shows types of HVAC and AC

systems. Small and medium-sized buildings generally use AC units for individual rooms. In our experiment, we considered split AC which is much common nowadays. We aimed to employ MCC for dynamic temperature control and error reporting for each room and the whole building. We assume that each room is fitted with the right-sized AC, not under or over-sized, so that our algorithm for maintaining the ideal dew point works uniformly for each AC. An undersized AC would not lower the temperature as expected, and an oversized AC would not lower the humidity. For experimental purposes, we chose to deploy the prototype MCC application at an institutional building in Kolkata, India. Since Kolkata falls in the tropical region, we considered the related parameters accordingly, as discussed in the following subsections.

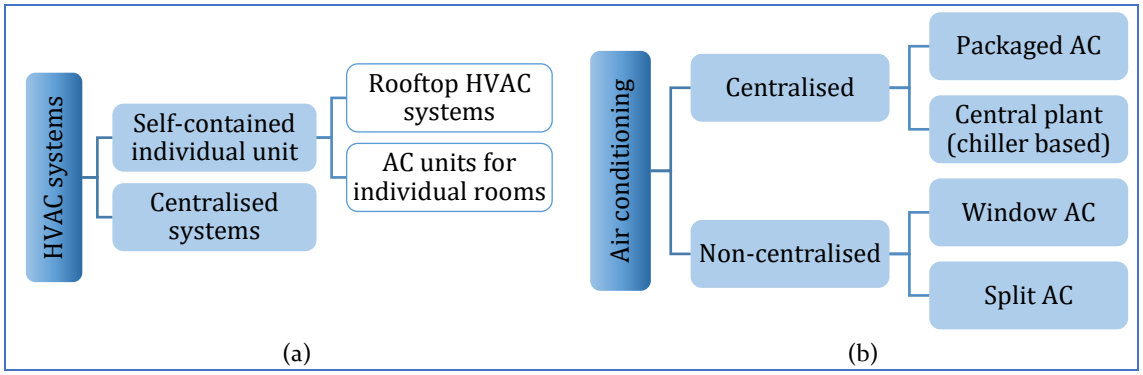


Fig. 9.3. Types of (a) HVAC systems and (b) air conditioning systems

9.2.2 Overview of the MCC-Edge Enabled HVAC System

Fig. 9.4 depicts a basic overview of the MCC-edge enabled HVAC system. The data generated by the temperature and humidity sensors and occupancy sensors of the individual rooms are read, processed, and analysed through MCC, and based on the outcome, the AC is controlled through suitable controllers. The purposes of the major components of the automated AC controlling are shown in Fig. 9.5. Here, for sensor data processing, we used public-owned SMDs, and the analysis is done on a Raspberry Pi based SBC, which acts as a local coordinator (LC) of MCC. The room temperature is dynamically regulated for the ideal comfort level of the occupant(s) by increasing and decreasing the AC temperature as per requirement. Furthermore, if an unusual increase or decrease of temperature is detected in a particular room, an error notification is sent to the facility staff. Similarly, the AC will be turned off if there is no one in the room (detected by the occupancy sensor). The

key inputs and the utilisation of the MCC-driven HVAC system are shown in Fig. 9.6.

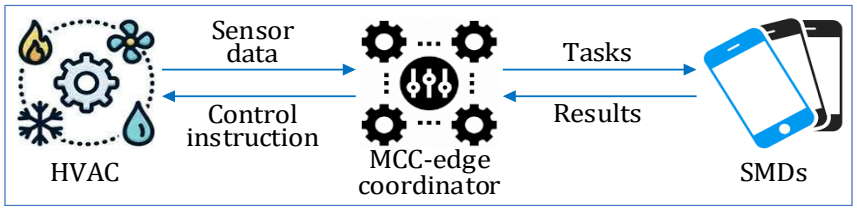


Fig. 9.4. Overview of the MCC-edge enable HVAC

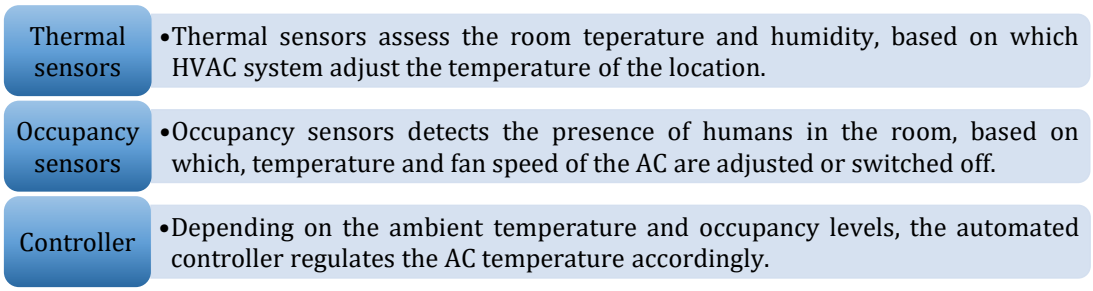


Fig. 9.5. Major components of the automated AC controlling in a typical smart HVAC system

Fig. 9.6. Key inputs and the purpose of the MCC-driven smart HVAC system

9.2.3 General Considerations

Since this work is only a preliminary experiment, to make the system implementations straightforward, we made some up-front considerations. However, for a full-fledged implementation, they need to be suitably incorporated. The assumptions are listed below and further discussed in Section 9.5.5.

- The SMD users voluntarily agreed to participate in MCC by installing the MCC client application on request.
- All the SMDs which have the MCC client application installed fulfil the minimum requirement of the fixed hardware resources such as CPU, GPU, etc.
- Each task sent to the crowdworkers has uniform computation requirements, input and output data size, and format.
- The crowdworker is available till the allocated job is finished. After completion, it returns the result to the coordinator.

- There is no cap on the number of tasks a crowdworker is assigned to.

9.3 HVAC Control Data Calculation

For impulsive adjustments of the room's environment as per the occupants' ideal comfort levels, we considered the following parameters:

a) **Dry-bulb temperature:** It is the ambient air temperature in the room that can be measured directly using a thermometer. It indicates the heat content in the atmosphere. Since we used a humidity sensor, we did not require to measure the wet-bulb temperature, the adiabatic saturation temperature.

b) **Relative humidity:** Relative humidity indicates "how close the air is to be saturated". It is the ratio between the specific humidity of the sample air (the amount of moisture in the air) and the specific humidity of saturated air (the amount of moisture that the air can hold at that temperature) of the same quantity. Table 9.2 shows the corresponding comfort levels for different humidity levels.

Table 9.2. Relative humidity and respective comfort levels

Humidity level (%)	Comfort level
Less than 15	Very dry
15 ~ 30	Dry
30 ~ 40	Comfortable
40 ~ 50	Ideal
50 ~ 60	Comfortable
60 ~ 80	Uncomfortable
More than 80	Very uncomfortable

c) **Dew point:** The temperature at which air becomes saturated with moisture, i.e., it reaches 100% relative humidity, is called the dew point. This measurement indicates "how does the air feel" and is always lower or equal to the air temperature. The higher the dew point, the muggier it would feel. Dew point gives a good measure for balancing temperature and relative humidity. A comfortable level of dew point indicates the right balance of temperature and humidity. Therefore, instead of assessing temperature and humidity individually, it is a good practice to assess the dew point of a room and try to keep it at a comfortable level. Fig. 9.7 depicts a sample representation of dewpoint with respect to temperature and relative humidity. Fig. 9.8 demonstrates the change in dew point with respect to increasing temperature and decreasing

relative humidity. Considering the weather of Kolkata, we set the comfort levels against different dew points, as shown in Table 9.3.

Temperature (°C)	Relative humidity (%)																		
45	6.4	12.4	16.9	20.4	23.4	26.0	28.3	30.3	32.2	33.9	35.4	36.9	38.3	39.5	40.7	41.9	43.0	44.0	45
44	5.6	11.6	16.0	19.6	22.6	25.1	27.4	29.4	31.3	32.9	34.5	35.9	37.3	38.6	39.8	40.9	42.0	43.0	44
43	4.9	10.8	15.2	18.8	21.7	24.3	26.5	28.5	30.3	32.0	33.6	35.0	36.3	37.6	38.8	39.9	41.0	42.0	43
42	4.1	10.0	14.4	17.9	20.9	23.4	25.6	27.6	29.4	31.1	32.6	34.1	35.4	36.6	37.8	39.0	40.0	41.0	42
41	3.4	9.3	13.6	17.1	20.0	22.5	24.7	26.7	28.5	30.2	31.7	33.1	34.4	35.7	36.9	38.0	39.0	40.0	41
40	2.6	8.5	12.8	16.2	19.1	21.6	23.8	25.8	27.6	29.2	30.8	32.2	33.5	34.7	35.9	37.0	38.0	39.0	40
39	1.9	7.7	12.0	15.4	18.3	20.8	22.9	24.9	26.7	28.3	29.8	31.2	32.5	33.8	34.9	36.0	37.1	38.1	39
38	1.1	6.9	11.2	14.6	17.4	19.9	22.1	24.0	25.8	27.4	28.9	30.3	31.6	32.8	33.9	35.0	36.1	37.1	38
37	0.3	6.1	10.3	13.7	16.6	19.0	21.2	23.1	24.9	26.5	28.0	29.3	30.6	31.8	33.0	34.1	35.1	36.1	37
36	-0.4	5.3	9.5	12.9	15.7	18.1	20.3	22.2	23.9	25.5	27.0	28.4	29.7	30.9	32.0	33.1	34.1	35.1	36
35	-1.2	4.5	8.7	12.0	14.8	17.3	19.4	21.3	23.0	24.6	26.1	27.4	28.7	29.9	31.0	32.1	33.1	34.1	35
34	-1.9	3.7	7.9	11.2	14.0	16.4	18.5	20.4	22.1	23.7	25.1	26.5	27.8	28.9	30.1	31.1	32.1	33.1	34
33	-2.7	2.9	7.1	10.4	13.1	15.5	17.6	19.5	21.2	22.8	24.2	25.5	26.8	28.0	29.1	30.1	31.1	32.1	33
32	-3.4	2.1	6.2	9.5	12.3	14.6	16.7	18.6	20.3	21.8	23.3	24.6	25.8	27.0	28.1	29.2	30.2	31.1	32
31	-4.2	1.3	5.4	8.7	11.4	13.7	15.8	17.7	19.4	20.9	22.3	23.7	24.9	26.1	27.1	28.2	29.2	30.1	31
30	-5.0	0.5	4.6	7.8	10.5	12.9	14.9	16.8	18.4	20.0	21.4	22.7	23.9	25.1	26.2	27.2	28.2	29.1	30
29	-5.7	-0.3	3.8	7.0	9.7	12.0	14.0	15.9	17.5	19.0	20.5	21.8	23.0	24.1	25.2	26.2	27.2	28.1	29
28	-6.5	-1.1	2.9	6.1	8.8	11.1	13.1	15.0	16.6	18.1	19.5	20.8	22.0	23.2	24.2	25.2	26.2	27.1	28
27	-7.3	-1.9	2.1	5.3	7.9	10.2	12.3	14.1	15.7	17.2	18.6	19.9	21.1	22.2	23.3	24.3	25.2	26.1	27
26	-8.0	-2.7	1.3	4.4	7.1	9.4	11.4	13.1	14.8	16.3	17.6	18.9	20.1	21.2	22.3	23.3	24.2	25.1	26
25	-8.8	-3.5	0.5	3.6	6.2	8.5	10.5	12.2	13.9	15.3	16.7	18.0	19.2	20.3	21.3	22.3	23.2	24.1	25
24	-9.5	-4.3	-0.4	2.8	5.4	7.6	9.6	11.3	12.9	14.4	15.8	17.0	18.2	19.3	20.3	21.3	22.3	23.2	24
23	-10.3	-5.1	-1.2	1.9	4.5	6.7	8.7	10.4	12.0	13.5	14.8	16.1	17.2	18.3	19.4	20.3	21.3	22.2	23
22	-11.1	-5.9	-2.0	1.1	3.6	5.8	7.8	9.5	11.1	12.5	13.9	15.1	16.3	17.4	18.4	19.4	20.3	21.2	22
21	-11.8	-6.7	-2.8	0.2	2.8	4.9	6.9	8.6	10.2	11.6	12.9	14.2	15.3	16.4	17.4	18.4	19.3	20.2	21
20	-12.6	-7.5	-3.7	-0.7	1.9	4.1	6.0	7.7	9.3	10.7	12.0	13.2	14.4	15.4	16.4	17.4	18.3	19.2	20
19	-13.4	-8.3	-4.5	-1.5	1.0	3.2	5.1	6.8	8.3	9.8	11.1	12.3	13.4	14.5	15.5	16.4	17.3	18.2	19
18	-14.1	-9.1	-5.3	-2.4	0.2	2.3	4.2	5.9	7.4	8.8	10.1	11.3	12.4	13.5	14.5	15.4	16.3	17.2	18
17	-14.9	-9.9	-6.2	-3.2	-0.7	1.4	3.3	5.0	6.5	7.9	9.2	10.4	11.5	12.5	13.5	14.5	15.4	16.2	17
16	-15.7	-10.7	-7.0	-4.1	-1.6	0.5	2.4	4.1	5.6	7.0	8.2	9.4	10.5	11.6	12.6	13.5	14.4	15.2	16
15	-16.5	-11.5	-7.8	-4.9	-2.5	-0.4	1.5	3.2	4.7	6.0	7.3	8.5	9.6	10.6	11.6	12.5	13.4	14.2	15
	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100

Dry	Ideal	Comfortable	Manageable	Uncomfortable	Very uncomfortable	Extremely uncomfortable	Health hazards

Fig. 9.7. Temperature, relative humidity, and dew point chart[43]

[43] https://www.mrfixitbali.com/images/articleimages/dew-point-chart-full.pdf

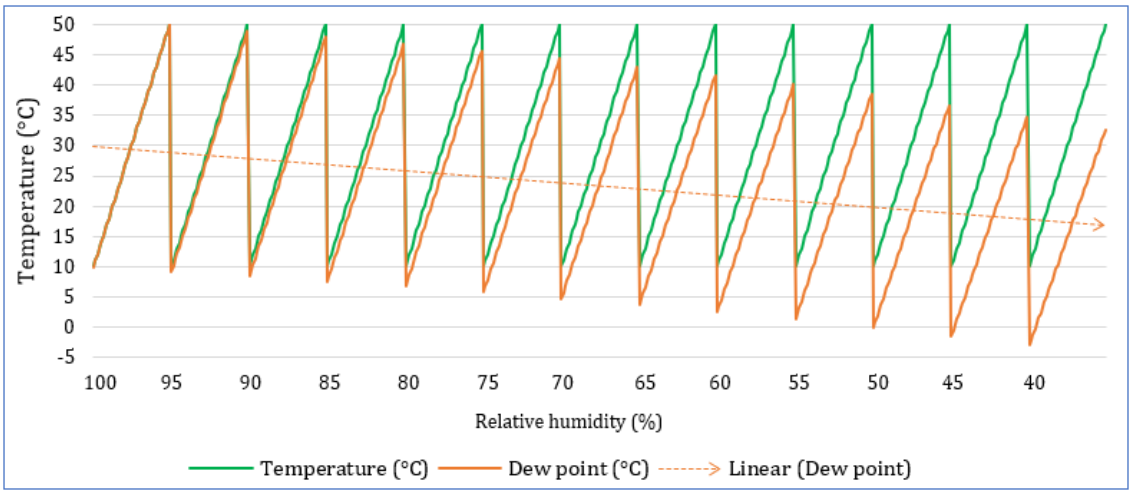


Fig. 9.8. Corresponding swing in dew point with continuous varying temperature and relative humidity

Table 9.3. Dewpoints and respective comfort levels as per Kolkata's weather

Dew point (°C)	Air quality
Below 5	Very dry
5 ~ 10	Dry
10 ~ 13	Ideal
13 ~ 16	Comfortable
16 ~ 19	Little humid but still comfortable
19 ~ 21	Humid
21 ~ 24	Very humid
24 ~ 26	Oppressive
26 and more	Health hazards

d) **Heat index:** This measurement indicates "how does the temperature feel", hence, also known as apparent temperature. Typically, in humid conditions, the air feels much hotter because it restrains the evaporation of perspiration from the skin. Even in the same environment (e.g., in the same room), different persons may experience different heat indices, though the variation may be very small. Although this measurement sounds subjective, using complex regressions with a combination of temperature and humidity, an indicative heat index can be calculated [757]. The heat index calculation is often associated with outdoor conditions, but as it is not intended to address direct sunlight rather is used to measure the air temperature in the shade, it can well also be an ideal metric to measure the indoor comfort level, especially in the tropical regions [758]. A higher heat index may be a severe health hazard and might cause heat cramps, heat exhaustion, heat stroke, or even death in extreme cases. Fig. 9.9 depicts a sample representation of heat index (°C) with respect

to temperature and relative humidity.

Temperature (°C)	Relative humidity (%)														
	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
50	68	74	81	88	96	105	114	124	134	144	156	167	180	193	206
49	65	71	77	84	91	99	108	117	126	137	147	158	170	182	195
48	62	67	73	80	87	94	102	110	119	129	139	149	160	172	184
47	59	64	70	76	82	89	96	104	113	121	131	141	151	162	173
46	57	61	66	72	78	84	91	98	106	114	123	132	142	152	163
45	54	58	63	68	73	79	86	92	100	107	116	124	133	143	153
44	52	55	60	64	69	75	81	87	94	101	108	116	125	134	143
43	49	53	57	61	65	70	76	82	88	94	101	109	117	125	134
42	47	50	54	57	62	66	71	77	82	88	95	102	109	117	125
41	45	48	51	54	58	62	67	72	77	83	89	95	102	109	116
40	43	46	48	51	55	59	63	67	72	77	83	88	95	101	108
39	41	43	46	49	52	55	59	63	67	72	77	82	88	94	100
38	39	41	43	46	49	52	55	59	63	67	71	76	81	87	92
37	38	39	41	43	46	48	51	55	58	62	66	70	75	80	85
36	36	38	39	41	43	46	48	51	54	58	61	65	69	74	78
35	35	36	37	39	41	43	45	48	50	53	57	60	64	68	72
34	33	34	35	37	38	40	42	44	47	49	52	55	58	62	66
33	32	33	34	35	36	38	40	41	43	46	48	51	54	57	60
32	31	31	32	33	34	36	37	39	40	42	44	47	49	51	54
31	30	30	31	32	33	34	35	36	38	39	41	43	45	47	49
30	29	29	30	30	31	32	33	34	35	36	38	39	41	43	44
29	28	28	29	29	30	30	31	32	33	34	35	36	37	39	40
28	27	27	28	28	28	29	29	30	31	31	32	33	34	35	36
27	26	27	27	27	27	28	28	28	29	29	30	30	31	32	33
26	25	26	26	26	26	26	27	27	27	27	28	28	28	28	28
25	24	24	25	25	25	25	25	25	25	26	26	26	26	26	25

Comfort	Caution	Extreme caution	Danger	Extremely danger	Death risk

Fig. 9.9. Temperature, relative humidity, and heat index chart

9.3.1 Dew Point Calculation and Consideration

To calculate the dew point, we used the Magnus-Tetens formula, defined by Eq. 9.1 and Eq. 9.2, considering two inputs – a) room temperature (T_C) and b) the relative humidity (RH).

$$dew\ point\ (D_C) = \frac{b \times \gamma(T_C,\ RH)}{a - \gamma(T_C,\ RH)} \tag{9.1}$$

where,

$$\gamma(T_C, RH) = \frac{a \times T_C}{b + T_C} + ln\frac{RH}{100} \tag{9.2}$$

with a = 17.27 and b = 237.7 (°C) [Different set of values of a and b are proposed. We considered the values used by [759] based on [760].]

The above calculation is correct to within ±0.4 °C and valid for the following ranges:

a) 0 °C < T_C < 60 °C

b) 1% < RH < 100%

c) 0 °C < D_C < 50 °C

Fig. 9.10 represents the relative humidity levels in Kolkata measured through the period between 1st February 2019 and 31st January 2020. It can be observed that the RH levels vary between 42% and 98%. Therefore, for our calculation, we considered the minimum RH as 40% and the maximum as 100%.

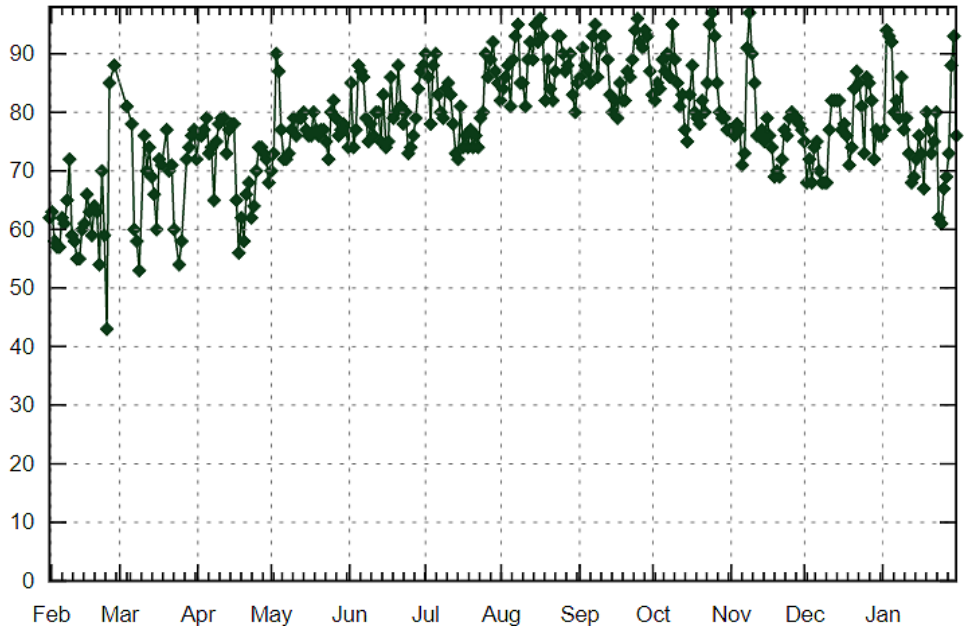


Fig. 9.10. Relative humidity (%) of Kolkata from 1.2.2019 to 31.1.2020 (retrieved from weather-online.in[44])

As per Table 9.3, we set to maintain the dew point within a range of 10°C – 13°C for any percentage of relative humidity. Table 9.4 tabulates the desired room temperatures for min and max dew points of 10°C and 13°C, respectively, for the humidity levels of 40 – 100%. Algorithm 9.1 imparts the procedure of adjusting the AC temperature based on Table 9.4 data.

Let us understand the working of Algorithm 9.1 with an example. Assume, at t_1

[44] https://www.weatheronline.in/weather/maps/city#

time, humidity is 75 and room temperature is 30. Using Eq. 9.1 and Eq. 9.2, the calculated dew point is 25. To bring down the dew point to 13, the AC temperature will be set to 16. In effect, the humidity drops and reaches 70 at t_2 time when the dew point is 10.5, which is within the ideal range, so there will be no change in temperature. 16°C might seem to be too cold, but it is to be noted that this is for a humidity level of 75, which is considerably higher. As the AC continues to run at this low temperature, the humidity will gradually decrease, which will lead to a decrease in dew point, considering the typical phases to decrease the humidity, as shown in Fig. 9.11. After a while, when again the room temperature and humidity are sensed, the dew point will be different. Let's say, at t_3 time, humidity is 50. So, at 16°C, the dew point will be 5.6. Therefore, the temperature now should be set to 22. This way, gradually, the dew point would settle somewhere between 10 to 13, and the AC temperature would also settle around 24. Table 9.5 shows the various comfort levels against different humidity concentrations and dew points.

Table 9.4. A representative calculation for desirable AC temperature

Relative humidity of the room's air (%)	Desirable room temperature for dew point 10 (°C)	Desirable room temperature for dew point 13 (°C)	Average desirable room temperature (rounded off)	AC temperature to be set
100	10	13	12	16
95	10.8	13.8	12	16
90	11.6	14.6	13	16
85	12.5	15.5	14	16
80	13.4	16.5	15	16
75	14.4	17.5	16	16
70	15.4	18.6	17	17
65	16.6	19.8	18	18
60	17.9	21.1	20	20
55	19.3	22.5	21	21
50	20.8	24.1	22	22
45	22.5	25.9	24	24
40	24.5	27.9	26	26

Usually, $RH \propto \frac{1}{T}$, i.e., relative humidity decreases as temperature increases if no moisture is added to the air because as the air becomes hotter, its water vapor holding capacity also increases, which means the saturation point increases. But, in a closed room with AC on, as the temperature decreases, its water vapor holding capacity also decreases; as a result, the relative humidity should increase, but most of the modern ACs have an evaporator coil that condenses water vapor in the air

of the room. This actually reduces the amount of water in the air resulting in decreased relative humidity. A perfectly working AC installed in a compatible-sized room can bring down the relative humidity as below as 30 at a temperature of 22°C - 24°C.

Algorithm 9.1: AC Temperature Adjustment Based on Dew Point

Input: Current room temperature (°C), Relative humidity (%)

Output: AC temperature adjustment required

```
AC_temp_adjst_DP(CurrRoomTemp, RH)
{
    if (RH >= 40 && RH <= 100)
    {
      Temp10 = calculateDesiredTemp(10, RH)  //calculate desired room temperature using Eq. 9.1
 and Eq. 9.2 for dew point 10 and relative humidity RH
      Temp13 = calculateDesiredTemp(13, RH) //calculate desired room temperature Eq. 9.1 and
 Eq. 9.2 for dew point 13 and relative humidity RH
      TempAvg = (Temp10+Temp13)/2
      if (TempAvg ≤ 16)
        DesiredTemp = 16   //set AC temperature to 16
      else
        DesiredTemp = TempAvg  //set AC temperature to average desirable temperature
      diff = DesiredTemp - CurrRoomTemp
    }
    return (diff)
}
```

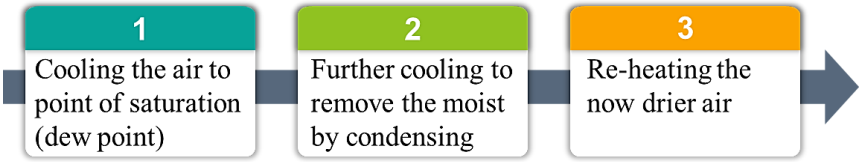


Fig. 9.11. Typical and necessary steps for reducing humidity from room air

Table 9.5. Relative humidity, dew point, and respective comfort level at a temperature of 24°C

Relative humidity at 24°C	Dew point (°C)	Comfort level
1 ~ 5	-35.4 ~ -18	Hazardous
5 ~ 10	-18 ~ 9.5	Unhealthy
10 ~ 20	-9.5 ~ -0.4	Very dry
20 ~ 30	-0.4 ~ 5.4	Dry
30 ~ 40	5.4 ~ 9.6	Comfortable
40 ~ 50	10.5 ~ 12.9	Ideal
50 ~ 60	12.9 ~ 15.8	Comfortable
60 ~ 70	15.8 ~ 18.2	Uncomfortable
70 ~ 80	18.2 ~ 20.3	Very uncomfortable
80 ~ 90	20.3 ~ 22.3	Extremely muggy
90 ~ 95	22.3 ~ 23.1	Oppressive
95 ~ 100	23.1 ~ 24	Hazardous

9.3.2 Heat Index Calculation and Consideration

There are several approaches and formulas for calculating heat index [761]. We

adopted the standard described by the National Weather Service[45] and was proposed by Lans P. Rothfusz. Eq. 9.3 shows the calculation of heat index, based on two inputs – a) Temperature (in Fahrenheit) and b) relative humidity (0 - 100%).

$$HI = -42.379 + 2.04901523 \times T_F + 10.14333127 \times RH - 0.22475541 \times T_F \times RH - 0.00683783 \times {T_F}^2 - 0.05481717 \times RH^2 + 0.00122874 \times {T_F}^2 \times RH + 0.00085282 \times T_F \times RH^2 - 0.00000199 \times {T_F}^2 \times RH^2 \quad (9.3)$$

However, Eq. 9.3 needs to be corrected a little bit for two certain situations, as mentioned below:

If $80 \leq T_F \leq 112$ and $RH < 13$, $Correction1$ is calculated using Eq. 9.4 and subtracted from Eq. 9.3.

$$Correction1 = (3.25 - 0.25 \times RH) \times \sqrt{1 - \frac{|T_F - 95|}{17}} \quad (9.4)$$

If $80 \leq T_F \leq 87$ and $RH > 85$, $Correction2$ is calculated using Eq. 9.5 and added to Eq. 9.3.

$$Correction2 = (0.1 \times RH - 8.5) \times (17.4 - 0.2 \times T_F) \quad (9.5)$$

The heat index, calculated using Eq. 9.3, may not be correct when $HI < 80$°F. In that case, Steadman's formula, as given in Eq. 9.6, is used to calculate the heat index. Actually, the normal convention is to first calculate heat index using Eq. 9.6, if $HI \geq 80$°F, then Eq. 9.3, along with suitable corrections, is used. The flowchart for calculating the heat index in our experiment is shown in Fig. 9.12. Since both Rothfusz and Steadman's formula are Fahrenheit-based calculations, we converted the Celsius reading into Fahrenheit, and for decision making, the calculated heat index is again converted to Celsius.

$$HI = 0.5 \times \{T_F + 61.0 + (T_F - 68.0) \times 1.2 + 0.094 \times RH\} \quad (9.6)$$

The heat index calculated for a person using Eq. 9.3 assumes that s/he is of average stature (170 cm, 67 kg), has a normal body temperature of 98.6 °F (37.0 °C), has minimal physical activity, is wearing a usual long trouser and a short-sleeved shirt,

[45] https://www.wpc.ncep.noaa.gov/html/heatindex_equation.shtml

and not sweating. Deviating from these may result in a slight change in the heat index of that particular person.

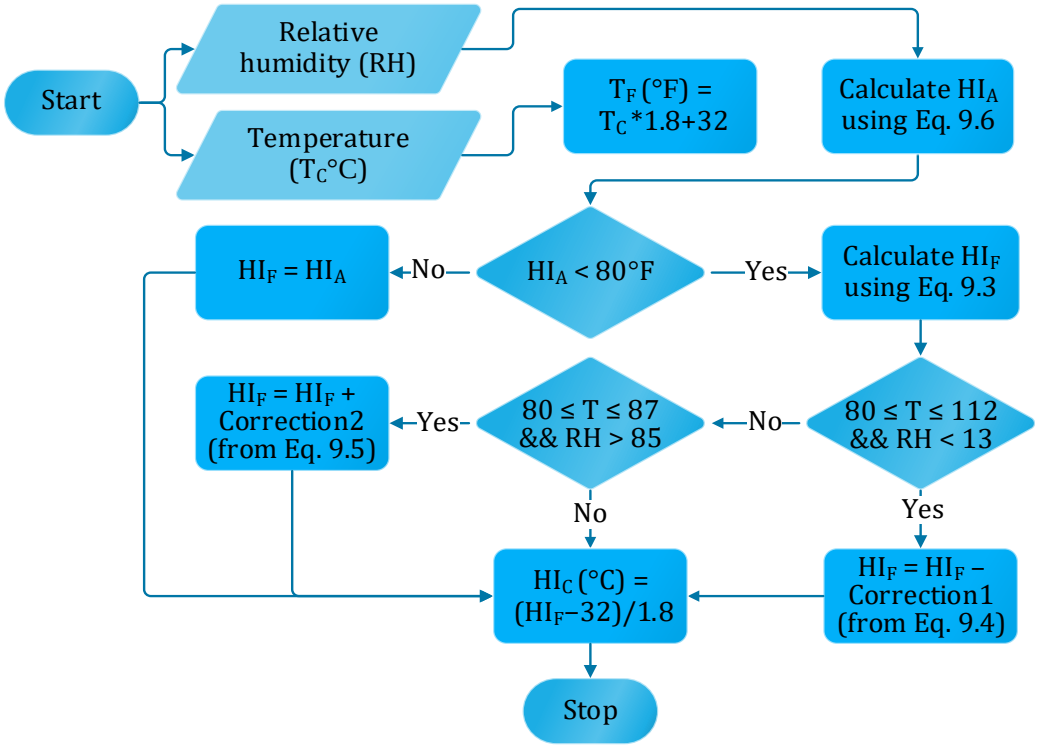


Fig. 9.12. Flowchart for calculating heat index

As mentioned earlier, though generally heat index is used for outdoor conditions, we emphasised to consider in our experiment bearing in mind that often in the peak summer, the sealed-off rooms (without windows opened) become extremely hot. Especially the rooms that get direct sunlight (e.g., the top-floor rooms) usually become like hot chambers without AC on. Therefore, it is necessary to check for the heat index of such rooms and set the AC accordingly so that the room gets cooled earliest. Table 9.6 shows comfort levels and the degree of health hazards for different levels of heat indices. To make it straightforward, we set the AC temperature to 16 if the heat index is found to be higher than 27°C. Algorithm 9.2 presents the scheme of AC temperature adjustment based on the heat index.

9.4 System Architecture and Implementation

This section exhaustively elaborates the complete architecture and implementational details of the proposed MCC-edge enabled smart HVAC system.

Table 9.6. Heat indices and respective comfort levels[46]

Heat index (°C)	Ambiance	Concern	Health hazard due to continuing exposure and activity
Below 27	Little to no discomfort	Safe	No risk of heat hazard.
27 ~ 32	Very warm	Caution	Fatigue and heat cramps.
32 ~ 39	Hot	Extreme caution	Fatigue, heat cramps, heat exhaustion, convulsions, and heat stroke.
39 ~ 51	Very hot	Danger	Severe fatigue, heat cramps, heat exhaustion, convulsions, and heat stroke.
Above 51	Extremely hot	Extreme danger	Heat stroke is imminent.

Algorithm 9.2: AC Temperature Adjustment Based on Heat Index

Input: Current room temperature (°C), Relative humidity (%)

Output: Boolean value for considerable heat index

```
AC_temp_adjst_HI(CurrRoomTemp, RH)
{
    if (RH >= 40 && RH <= 100)   //consider minimum humidity 40% and maximum humidity 100%
    {
      Hx = calculateHeatIndex(CurrRoomTemp, RH) //calculate heat index using Eq. 9.3, Eq. 9.4, Eq. 9.5 and Eq. 9.6
      if (Hx ≥ 27)
        HeatFlag = True
      else
        HeatFlag = False
    }
   return (HeatFlag)
}
```

9.4.1 The MCC-Edge System Model

In this section, we specifically discuss the details of the proposed MCC-edge system. This includes the components, architecture and the working of the system.

9.4.1.1 Major Components

A typical MCC-edge system consists of the following major components:

SMDs: The computing tasks are carried out on the SMDs that agree to be a crowdworker. A crowdworker is an SMD that is willing to take part in MCC and allow its computing resources for executing external jobs.

Local coordinator: In an MCC environment, the SMDs are coordinated by the local coordinators (LCs) which are in direct contact with the SMDs. An LC can be an SBC, an SMD, an AP, a computer, and so on. In this experimental setup, we

[46] https://www.weather.gov/safety/heat-index

considered SBCs as LCs. The responsibilities of an LC are listed in Fig. 9.13.

Middleware: The middleware is a part of LC and is responsible for performing most of the jobs of LC. The jobs include SMD selection, job formation, job distribution, job scheduling and allocation, fault handling, result collection (from the SMDs), etc.

Client program: The client part of the MCC application is installed on the crowdworkers. It is responsible for sending the SMD's information to the LC. It is also responsible for executing the allocated task in the SMD, opportunistically, in a work-stealing fashion, i.e., without interrupting the other applications running in the device. Before installing the client application, it is to be checked if the SMD satisfies the minimum resource requirement. If it fulfils, then only the client application is installed.

MCC coordinator: Supervises the overall MCC, including fault tolerance and aggregated resource management. It keeps track of all the LCs and their resource requirement and SMD availability. If a particular LC does not have the minimum resources, the MCC coordinator (MC) makes arrangements to allow the LC to use SMDs from other neighbouring LCs. In case of no sufficient SMDs are available in the MCC, the MC coordinates with the cloud to get the job done.

Cloud: Cloud services are used as the backup to MCC when sufficient crowdworkers are not available. Furthermore, the required information are stored in the cloud for long-term access. This would allow the application of different analytical methods for pattern and knowledge discovery, which helps in predictive maintenance.

Communication network: Different communication methods are used to connect the components of MCC. The sensors are connected to the LC through Wi-Fi. The LCs and the SMDs are connected to the Wi-Fi APs. All the APs are connected through a LAN. MCs are connected to the LCs through Wi-Fi and to the cloud through an external network. A high-level communication architecture of MCC is shown in Fig. 9.14.

Acquiring the sensor data	•Colects the data produced by the sensors connected to it.
Store aquired data	•Stores the collected sensor data temporarily.
Track SMD availability	•Keeps tracks when an SMD enters within its range. Also notify the MC if there is no or sufficient number of SMDs available.
SMD selection	•Collects the dynamic statistics (e.g. RAM, battery, Wi-Fi, etc.) of the connected SMDs to decide their suitability.
Job schedule and dispatch	•Dispatch job threads to the selected SMDs for processing.
Result collection	•After processing SMDs send the result to the LC (via AP).
Store result	•The results are temporarily stored for immediate analysis and event notification.
Analyse result	•Results are analysed, based on which actions are taken.
Error notification	•Triggers alerts in case of abnormalities in AC operations.
Support neighbours	•Takes the responsibility of neighbouring LC's jobs if such request received from MC.

Fig. 9.13. Responsibilities of the local coordinator

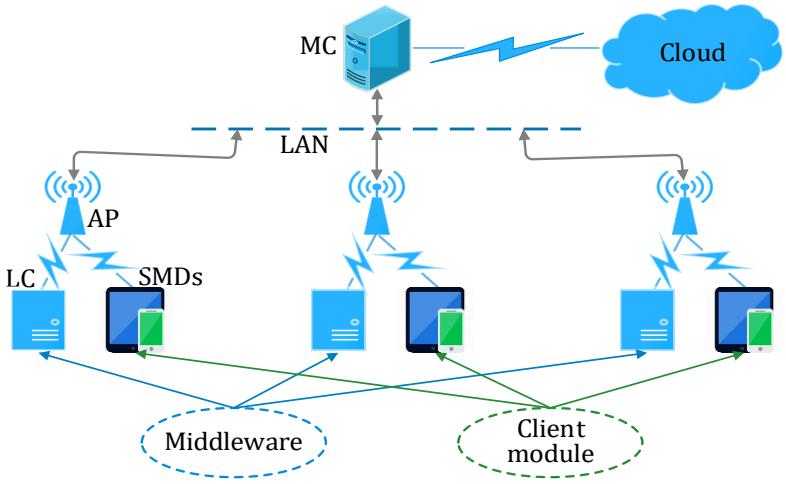


Fig. 9.14. High-level communication architecture of MCC

9.4.1.2 System Architecture

The proposed system can be perceived through different architectural views, as discussed below.

9.4.1.2.1 Layered Architecture

An abstract layered architecture of the proposed system is portrayed in Fig. 9.15. The functionalities of the layers are described below.

Cyber-physical layer: This is the base layer that comprises hardware components like sensors, controllers, and actuators. The sensor transmits the sensed data to the upper layer, while the controller gets instructions/commands from the upper layer to control the heating/cooling and air flow systems.

LC layer: LC layer directly communicates with the hardware layer. This layer manages the job processing, which includes coordinating the available SMDs, job distribution and dispatch, and result collection. The acquired sensor data are sent to the job dispatch module, which packages the data as a processing job and assigns it to the SMDs for processing. The resource control & coordinate module keeps resource status and availability information of the SMDs. If sufficient resources are not available at its disposal, it coordinates with the resource control and coordinate module of the MC layer. SMD select module selects the best crowdworkers among the available crowdworkers and sends the list of selected crowdworker to the job dispatch module, which dispatches the job to the selected crowdworkers. The results obtained from the SMDs are stored in a database through the result collection module and are analysed by the result analysis module to generate instruction for hardware control and event notification. Subsequently, the hardware control module sends commands to the corresponding controller, and the event notification generates user notification.

Data processing layer: This layer contains SMDs as data processing units that take jobs as inputs, process them, and send the output to the corresponding LC. Each SMD communicates with the LC layer through the communication interface. A buffer is used to synchronise the incoming and the outgoing rates of the jobs and the output, respectively.

MC layer: MC layer is responsible for the overall supervising and managing of MCC. It, on the top of the architecture, observes for available SMDs under each LC, and based on the resource availability, it coordinates the load distribution from one LC to another. The MC helps an LC to find resources in the neighbourhood if it does not have sufficient SMDs under its management. The resource coordination is done through the communication between the respective resource control & coordinate modules of the MC and LC layers. MC layer also interfaces MCC with the

cloud. In case of unavailability of SMDs or below threshold availability for all LCs, the data processing jobs are forwarded to the cloud. The job dispatch module obtains the job and dispatches it to the cloud through the cloud interface. The obtained results are sent to the database of the respective LCs.

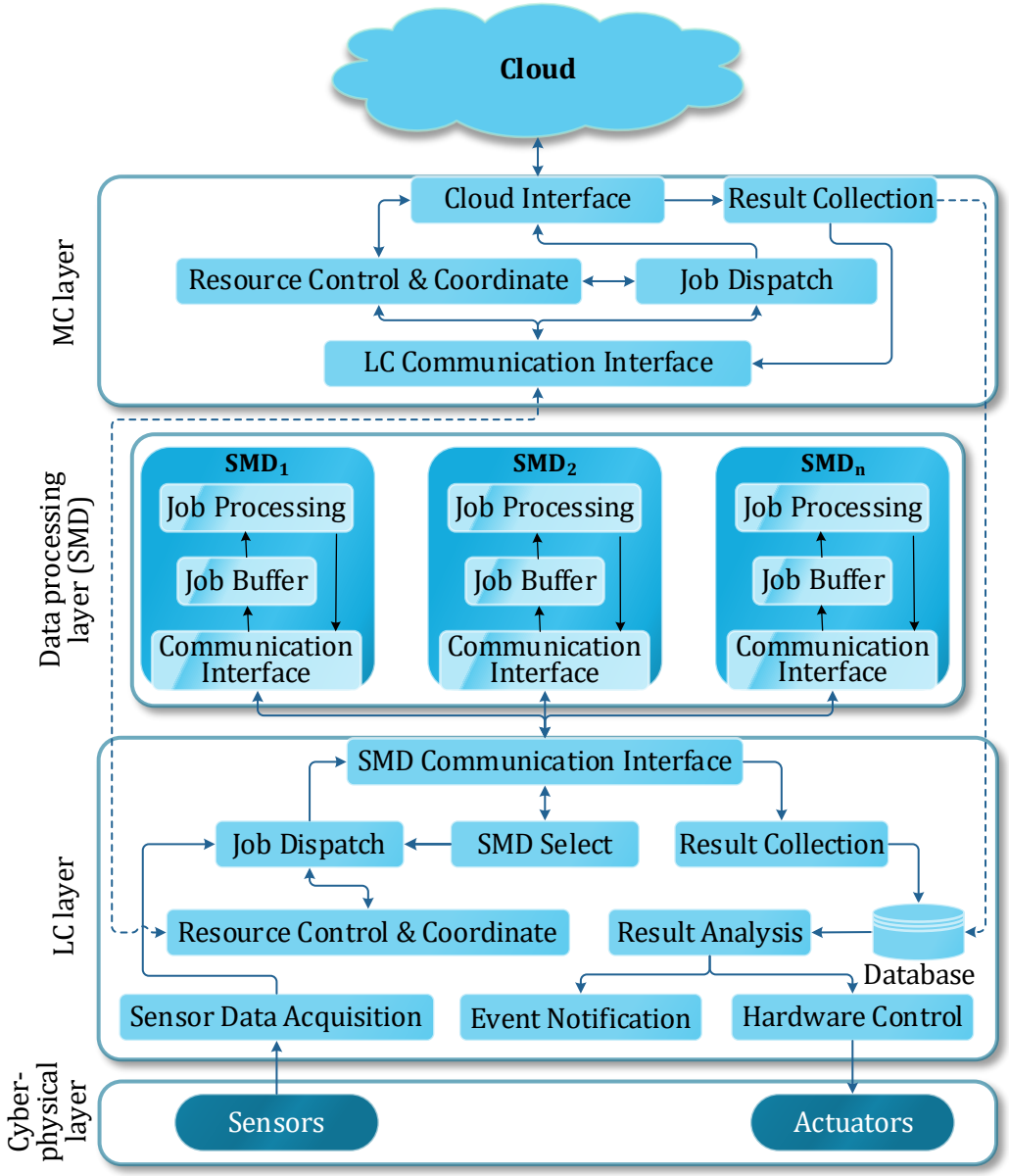


Fig. 9.15. The layered architecture of the proposed MCC-edge system

9.4.1.2.2 Hierarchical Architecture

The proposed system is perceived as a hierarchical client/server model, as shown in Fig. 9.16. Each SMD runs the client module of the MCC-edge application. The client module is responsible for carrying out the assigned MCC task opportunistically or in a CPU-stealing fashion. The LC controls the SMDs under it and performs the responsibilities of a typical server, such as resource management, job

management, and error reporting. The MC takes care of fault tolerance globally and error notification, as discussed in Section 9.4.1.2.1 and Section 9.4.3.5.

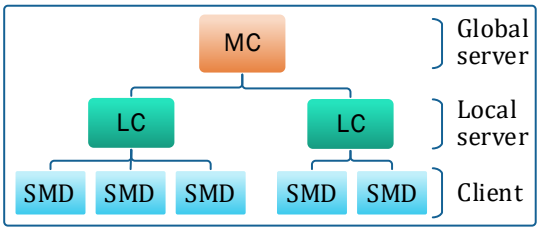


Fig. 9.16. Hierarchical architecture of MCC-edge

9.4.1.3 System Process

The sequence of the key processes through which the functioning of the proposed system proceeds is shown in Fig. 9.17, while the complete process flow is shown in Fig. 9.18. The data generated by the sensors are sent to the LC, which creates independent jobs, selects suitable crowdworker among the available SMDs, and dispatch the jobs to the selected SMDs. The SMD executes the assigned job and returns the result to the LC, which analyses the result and forwards the actionable instructions to the AC controller. As shown in Fig. 9.18, if no sufficient crowdworkers are available under an LC, it forwards its jobs to the peer LCs with the guidance of the MC. If no crowdworker is available at all, the jobs are sent to the cloud. In case of any issue, such as the AC is not able to maintain the desired temperature for a long time, an alert message is sent to the maintenance people. The complete system cycle is complemented by several data silos. The key data management components in a typical MCC system are shown in Table 9.7.

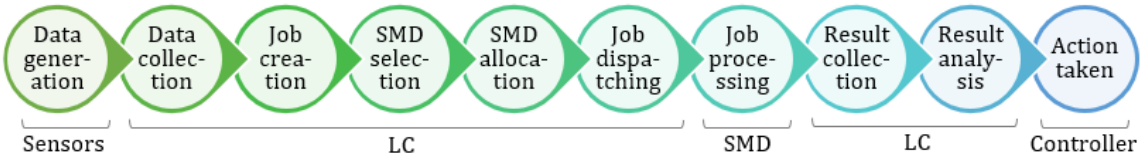


Fig. 9.17. The system process sequence of a typical MCC

9.4.2 System Set Up

To design the MCC-edge system, various hardware and software were used. In the section, we give the details of each component along with the hardware and networking set up.

9.4.2.1 Hardware and Software

The key hardware and software used in the experiment are discussed in the

following. In Table 9.8, all the necessary hardware and software elements, in general, along with their purpose and other details are listed, while

Table 9.9 lists the details of the hardware and software, in particular, used in the experiment.

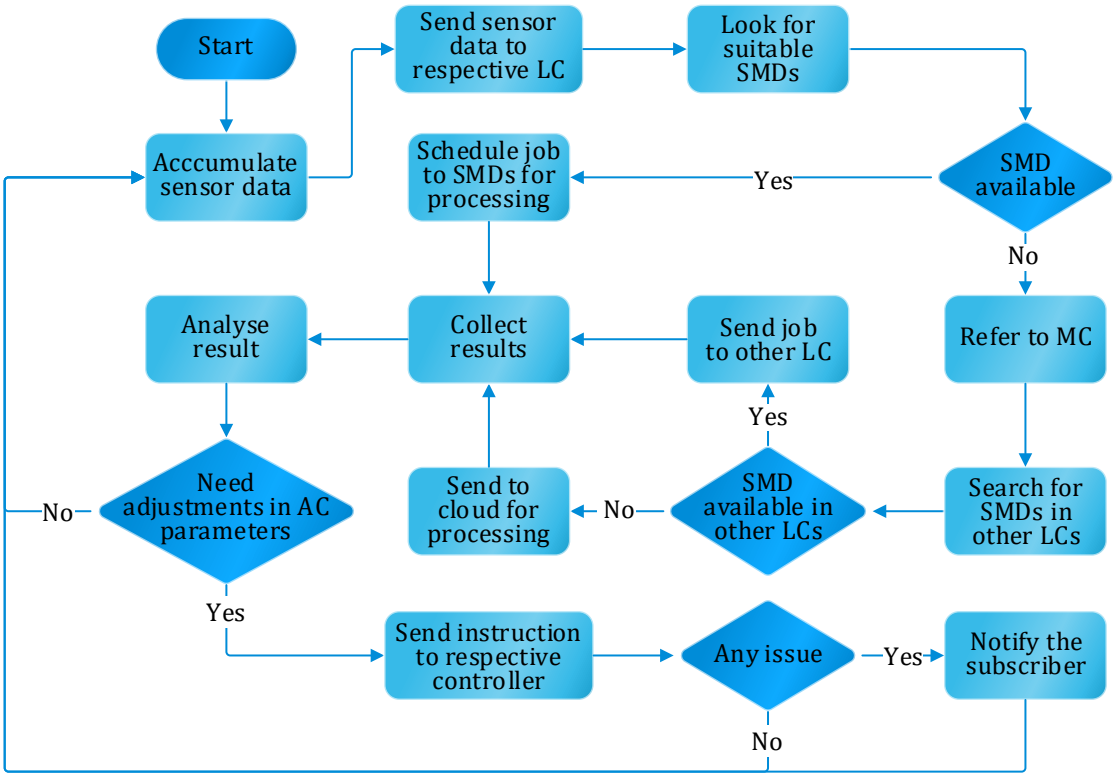


Fig. 9.18. System process flow

Table 9.7. Key data management components in a typical MCC-enabled HVAC system

Process	Purpose	Concerned entity	Employed site
Data generation	Assess the ambient environment	Sensors	Each individual room
Data transport	Transmission of data from source to processing unit and to sink	Wi-Fi, LAN	Throughout the building
Data collection	Accumulate sensor data	SBC	Each LC
Data computations	Process sensor data	SMD	SMDs under a certain LC
Data storage	Store processed data for a limited period	Light-wight database	Each LC
Data analysis	Detecting anomaly and reporting	SBC	Each LC
Data access/ uses	Receiving real-time alert and periodical report	Facility staff, managers	Devices of facility staff and managers

NodeMCU: Node Micro Controller Unit (NodeMCU[47]) is an open-source IoT

[47] https://www.nodemcu.com/index_en.html

platform that runs on an ESP8266 Wi-Fi chip. The firmware of this micro-controller communicates through Wi-Fi using TCP/IP. It comes onboard with analogue and digital pins and acquires data from the different types of sensors with the support of serial communication protocols like UART, SPI, I2C, etc. The detailed technical specification NodeMCU v3 can be found at [762]. In our experiment, we used it to connect the temperature and occupancy sensors and also to control the AC through infrared signals.

Raspberry Pi: It is a small SBC that supports small databases (user-created), is loaded with Linux operating system, and provides an environment for Java and Python programming. The Raspberry Pi 3 Model B[48], used in our experiment, has a 1.2 GHz processor, 1 GB RAM, and an SDD secondary storage. It supports Wi-Fi and Ethernet communication networks and has four USB ports, one Ethernet port, and one HDMI port. In our model, the Raspberry Pi is used as the coordinator (LC) for collecting data from the connected sensors and coordinating the job processing in the SMDs.

Lua script: Lua[49] is a lightweight, embedded scripting language that supports object-oriented, procedural, and data-driven programming approaches. It works across multiple platforms ranging from servers to mobile devices. Lua provides a programmable interface to microcontrollers like NodeMCU, enabling them to control the equipment's operability. Lua script is being flashed on the NodeMCU firmware, which is loaded into RAM for processing the required operation. It uses a library package named AdaFruit to read the sensor values. We wrote the NodeMCU controlling program using the Lua script.

MariaDB: MariaDB[50] is one of the most popular lightweight open-source relational database servers and is fully compatible with MYSQL. MariaDB is the default choice of Debian-based operating systems such as Raspberry Pi. Installation of the MariaDB server in Raspberry Pi is very straightforward; also, it works very fast in

[48] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[49] https://www.lua.org/about.html
[50] https://mariadb.org/

Raspberry Pi, even in the older versions of hardware. We used MariaDB version 10.5 at the LC module for storing data temporarily.

DHT22: For acquiring temperature and humidity data, DHT22, a low-cost digital humidity and temperature sensor, is used. The sensor consists of an in-built humidity sensor and temperature measuring thermistor. The sensor reads the surrounding air and produces the temperature and relative humidity measurement as a digital output. We used a DHT22 from SparkFun[51], the technical specification of which is summarised in Table 9.10, and the details can be found at [763]. The details of the measurement of a DHT22 are shown in Table 9.11. We used this sensor to assess the room temperature and the relative humidity.

PIR sensor: A PIR (Passive Infra-Red) sensor perceives the movement or presence of people in a room by detecting infrared heat radiation of moving people (up to 20 feet or 6 meters with 110° x 70° detection range). The sensor produces high voltage if motion is detected, otherwise low. We used this sensor to detect if there is any person in a room.

9.4.2.2 Circuit Design

The DHT22 and PIR sensors are accessed and controlled by a program-enabled micro controller NodeMCU. DHT22 and PIR sensors are connected to NodeMCU through a single-wire interface at a digital input pin, as shown in Fig. 9.19. Each sensor is connected with a Raspberry Pi (LC) through a NodeMCU, as shown in Fig. 9.20. A control program written using Lua script is flashed into the ESP8266 board, which gathers the digital signals from DHT22 and PIR and decodes suitably. The Lua script sends the acquired data (temperature and RH for DHT22 and occupancy for PIR), annotated with the sensor id, to the LC using a socket connection.

9.4.2.3 Networking and Communication

In our implementational model, each LC controls a small-scale local MCC, typically covering a portion of a certain floor of the building. Generally, in a crowded commercial building, several Wi-Fi APs are installed on the same floor because the Wi-

[51] https://www.sparkfun.com/products/10167

Fi speed gets deteriorated imperceptibly as more devices get connected. All the LCs are connected to a centralised MC through the backbone network, and the MC communicates with the cloud.

Table 9.8. Hardware and software requirements for a typical MCC setup for an HVAC system

Category	Element	Purpose	Provider/ developer	Working/installation site
Hardware setup	Sensors	Assessing the ambient conditions	Sensor vendor	Installed in each room
	Wi-Fi APs	Through which sensors and SMDs are connected to the LC	Network vendor	Installed for a floor
	SBCs	Gathers sensor data and connects to AC controller	SBC vendor	Installed in each room
	Controllers	Regulates the AC temperature	Controller vendor	Installed with each ACs in room
	Gateway	Connects the MC to the cloud	Network vendor	Installed along the backbone network, one unit installed per building
	SMDs	Process the job received from LC and receive notifications	Mobile device vendor	For processing, the SMDs (crowdworkers) available within the premises, and for notifications, the SMDs of the maintenance people
Software support	Client application	Receive the MCC job from LC, get it executed on the SMD, and return the result to the LC	Designed and developed by the MCC developer	Voluntarily installed on the SMDs
	Server program	Performs and supervises the core functions of MCC	Design and developed by MCC developer	Installed at LC and MC
	APIs	Connects/interfaces different software application programs	Library package provided by the system software vendor	LC, MC, and SMD
	GUIs	Access HVAC system operations report	Design and developed by MCC developer	Devices of administrators and facility staffs
	Local database	Stores sensor data and processed information	Open-source	Installed at each LC
	Controller program	Accesses sensors and controls AC	Library package provided by system software vendors	Loaded on NodeMCU and SBC

Each local MCC under each LC functions independently, unless in certain

scenarios like unavailability of the required number of crowdworkers. The SMDs are connected to the LC through the AP that is connected to the backbone network via a switch. The typical layout of our network representation for the experiment is shown in Fig. 9.21.

Table 9.9. Implementational environment specifications for the proposed system

Program/ application	Runs on	Language/platform	Purpose
Sensor module	NodeMCU	Lua script and associated library	Accumulating and transmitting sensor data
LC module	Raspberry Pi	Java	Hosting the functionalities of the LC
Database	Raspberry Pi	MariaDB	Storing system data generated from various components
SMD module	Android mobile devices	Java, compatible with Android version 4.4 (KitKat) and above	Execute the MCC job opportunistically
Controller module	NodeMCU	Lua script and associated library	Sending control signal to regulate AC
MC module	Computer	Java	Overall supervising and ensuring fault tolerance

Table 9.10. Specifications of DHT22

Specification	Value
Sampling rate	0.5Hz (one reading per two seconds)
Size	15.1x25x7.7 mm
Operating voltage	3-5V
Max current during measuring	2.5mA
Repeatability	±1%
Durability	±0.5% per year
Manufacturer	MaxDetect Technology Co. Ltd.

Table 9.11. DHT22 measurement details

Measurement	Range	Accuracy
Temperature	-40°C to +125°C	±0.5°C
Relative humidity	0 to 100%	±2%RH

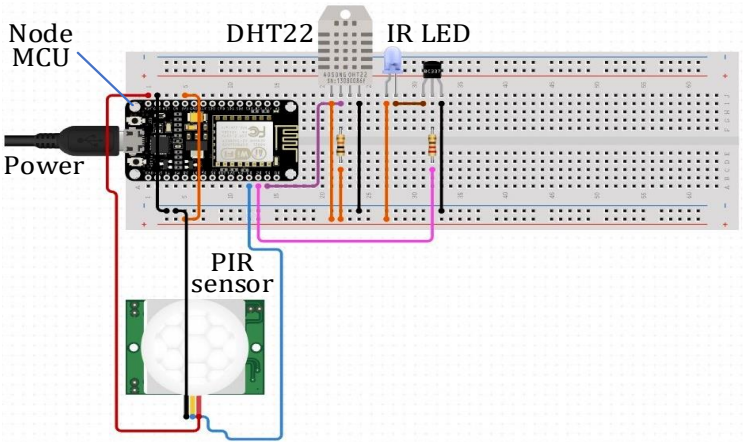


Fig. 9.19. Circuit connection between NodeMCU and DHT22, PIR sensor, and IR LED

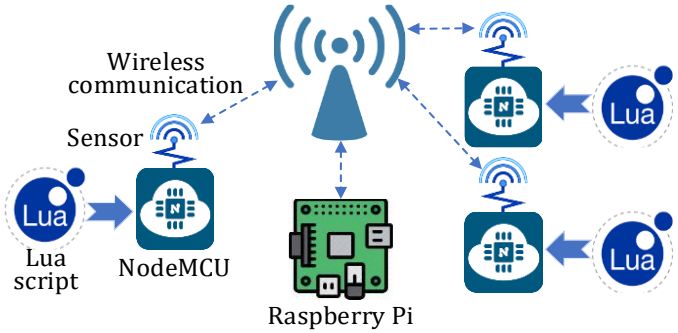


Fig. 9.20. Connection between NodeMCU and Raspberry Pi

The LC and the SMDs connected to it are in the same address space. The LC always has a fixed address in this address space so that an SMD can easily locate the LC when it connects to the associated AP and initiate communicating with the LC. We planned to make this address common to every LC in the building, wrapped as a subnet mask. This would allow the client module of a mobile user to connect different LCs within the building seamlessly.

At the application layer, socket connections are used, the details of which are shown in Table 9.12. The socket connection helps in data communication irrespective of the interfacing programs at the end of the communication channel. Physically the NodeMCU is connected with the LC through Wi-Fi, while it controls the AC through an infra-red signal. At the data communication level, the NodeMCU connects to LC through a socket.

Table 9.12. Socket connections used in the experiment

Entity	Port no.	Incoming/ outgoing	Connects to	Purpose
LC	5000	Both	SMD	Sending job packets and receiving results
	6000	Incoming	NodeMCU	Receiving sensor data
	7000	Outgoing	NodeMCU	Sending AC control instructions

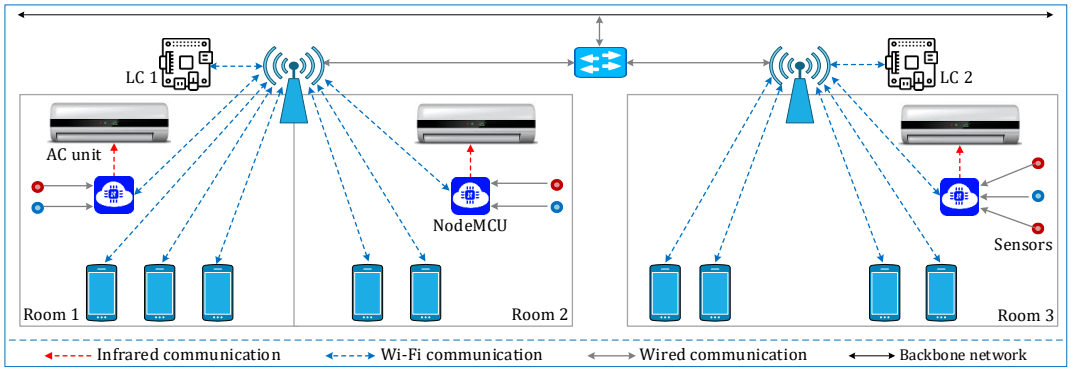


Fig. 9.21. Network layout of a local MCC

9.4.2.4 System Layout

Fig. 9.22 demonstrates a representative layout of the MCC-powered HVAC system for a four-storey office building. Each room houses decentralised AC units. The exhaust fans are placed at the roof top. Each temperature sensor is associated to each AC. The number of sensors installed according to the room size. For example, in a room of size less than 12 x 12 feet, one DHT22 is sufficient, while for a bigger room, depending on the size, two or more could be used. Similarly, in a large hall, depending on the room size, two or more PIR sensors are installed. All the sensors are placed in such a way so that the whole room is covered uniformly. It should be noted that though for designing the system we took into account the whole building, for experimental purpose we considered only three adjacent rooms on the same floor. However, we believe this should not deter the working of the proof-of-concept.

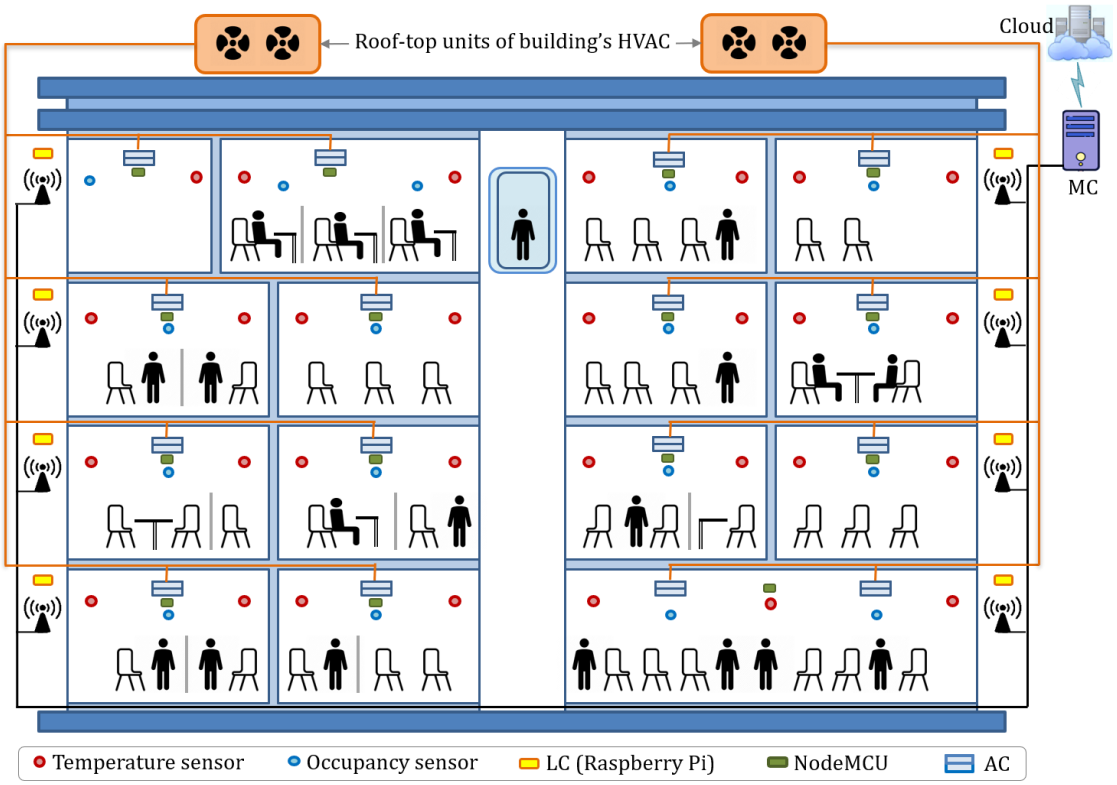


Fig. 9.22. System model layout for a four-storey office building

9.4.3 System Design and Implementation

The design of the HVAC-MCC system is presented in Fig. 9.23. The design consists of five modules which are discussed in this section.

9.4.3.1 Sensor Module

The sensor module, embedded on NodeMCU, is configured to capture the ambiance information through sensors and transmit them to the LC. The module performs the following functions:

a) Data acquisition, filtering, and annotation: With the help of the AdaFruit library, the Lua Script, running on the NodeMCU, accesses the digital sensors, acquires the digital signal, and converts it into data. Due to faulty sensors and other reasons, it may return an erroneous value. The program script filters out the erroneous data (anomaly) based on the threshold value set for each type of sensor reading. The filtered data is being tagged with the sensor id. The Lua script program gathers the PIR sensor and DHT22 sensor data in a period of thirty seconds and five seconds, respectively. The pseudocode for the data acquisition process is shown in Procedure 9.1.

b) Establishing data communication: Using the Lua script, the client socket connection is created in NodeMCU, which connects the server socket at LC. Using the established socket connection, temperature, humidity, and occupancy data are transmitted from the sensor module to the LC module.

9.4.3.2 SMD Module

SMD module is designed to provide computing services for the jobs dispersed by the LC module. The functionality of this module is described in Procedure 9.2. When a crowdworker enters an LC network, the LC retrieves the status of its resource parameters. If it is selected as the resource provider, it receives the job from the LC and stores it in the form of a sequential list structure in the buffer. A rule set is defined in the client application according to which the jobs are processed in a CPU-stealing fashion, i.e., when the SMD has a low processing load or is in an idle state. On processor availability, the job from the buffer is fetched in first-in-first-out (FIFO) manner for processing. The process result is sent back to the LC module. We implemented the SMD module as an Android application that was developed in Java to run on an Android device.

The client module receives the sensor data (temperature, humidity, and

occupancy) from LC. Using the temperature and humidity, the dew point and heat index are calculated. With these two parameters, the ideal room temperature is decided. The client module checks the difference between the ideal temperature and the current temperature using Algorithm 9.1 and Algorithm 9.2. If they are different, the difference value is sent to the LC based on which the AC temperature would be adjusted. The temperature difference (∇T_a) is calculated using Eq. 9.7. For example, let us assume the present room temperature is 30°C and the desired temperature is 20°C. Therefore, the SMD module would send '-10' to LC.

$$\nabla T_a = T_d - T_c \tag{9.7}$$

where, T_c is the current temperature and T_d is the desired temperature.

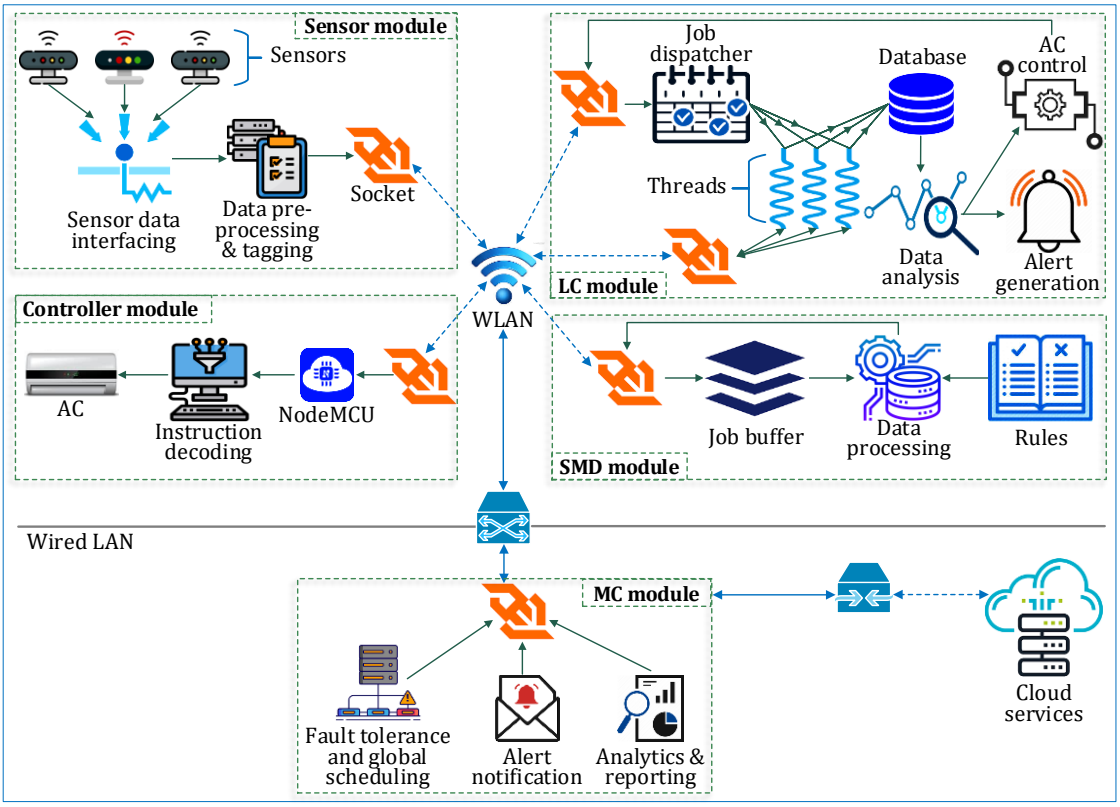


Fig. 9.23. Various modules of the proposed HVAC-MCC system

9.4.3.3 LC Module

The LC module is designed to carry out the responsibilities listed in Fig. 9.13. To communicate (for job dispatching and result collection) with the SMDs, the LC initiates individual threads for each selected SMD, i.e., each SMD communicates to the LC through different interfaces. Each thread maintains a live connection until the communication is terminated. In the LC, a job pool is maintained where

the jobs to be processed are pooled temporarily before dispatching to the designated SMD. When an SMD leaves the network, the connection is gracefully terminated, the buffer space in the pool is released, and the SMD is deallocated so that there is no further job dispatch to that particular SMD. The entire LC module is implemented through seven functional activities, discussed in the following subsections.

Procedure 9.1: Sensor Data Acquisition

```
clientSocket = createSocket()  //a client socket is created
clientSocket.connect(LC_IP, 6000)  //connect the LC at port 6000 at LC_IP for data transmission
tempSenseThread = TempSense.createThread()  //thread created for continuous room temperature sensing
pirThread = OccupSense.createThread()  //thread created for occupancy sensing
tempSenseThread.start()  //a thread for temperature and humidity data acquisition and transmission is started
pirThread.start()  //a thread for occupancy data acquisition and transmission is started

Thread TempSense  //defining the thread for temperature and humidity data
{
   start()
   {
      while (True)
      {
         wait(5)  //read temperature and humidity data every 5 seconds
         sensor = accessSensor(DHT22)  //access the DHT22 sensor
         tempData = sensor.getTemperature()  //read the temperature data from DHT22
         RHData = sensor.getRelativeHumidity()  //read the humidity data from DHT22

         if (tempData ≥ 5 && tempData ≤ 55)  //we considered temperature range between 5°C to 55°C
         {
            data = sensorId  + "#" + tempData + "#" + RHData  //'#' is used as temperature and humidity data delimiter
            clientSocket.send(data)  //send the temperature and humidity data to LC
         }
      }
   }
}

Thread OccupSense  //defining the thread for occupancy data
{
   start()
   {
      while(True)
      {
         wait(30) //read occupancy data every 30 seconds
         occupData = accessSensor(PIR)  //access the PIR sensor
         if (occupData == 1 || occupData == 0)  //occupancy value 0 denotes no body in the room, and 1 denotes one or more persons are there in the room
         {
            data = sensorId  + "@" + occupData  //'@' is used as occupancy data delimiter
            clientSocket.send(data)  //send the occupancy data to LC
         }
      }
   }
}
```

Procedure 9.2: MCC Task Execution in SMD

```
clientSocket = createSocket()
clientSocket.connect(LC_IP, 5000)  //connect the LC at port 5000 at LC_IP for data communication

while(True)
{
  data = clientSocket.receive()  //the client application in SMD module receives data from LC
  t1 = t2 = getTime()
  while (t2-t1 ≤ 30)  //the life of a received task is 30 seconds; if it is not executed by this time, it is aborted
  {
    if (getCPULoad() ≤ 70)  //the task will be executed if CPU load is less than equal to 70%
    {
      sensorId, tempData, RHData, t_date, t_time = data.split()
      if (AC_temp_adjst_HI(tempData, RHData))  //check for heat index using Algorithm 9.2
        ta =16 - tempData
      else
        ta = AC_temp_adjst_DP(tempData, RHData)  //calculate temperature to be adjusted using Algorithm 9.1

      result = sensorId + "#" + t_date + "#" + t_time + "#" + tempData + "#" + RHData + "#" + ta
//combine temperature to be adjusted with other information
      clientSocket.send(result)  //send result to LC
      break  //present task is executed
    }
    else
      t2=getTime()  //get the present time to check the timer for 30 seconds
  } //if resource is not available even after 30 seconds, the task is aborted
}
```

9.4.3.3.1 Sensor Data Collection and Job Creation

A server socket is created at the server program of LC to connect the client socket of the sensor module. The sensor data are continually collected in the form of data streams. The received data are stored as processable job units in a list-structured job pool. The jobs are comprised of sensor readings and required metadata (e.g., sensor id, date & time). Each job has uniform computation requirements, input and output data size, and format. It is to be noted that only the temperature and humidity data were used as processable jobs and sent to the SMDs. The occupancy data are directly checked for a binary outcome (occupancy yes or no) at the LC. The pseudocode for the process of data collection and job creation is shown in Procedure 9.3.

9.4.3.3.2 SMD Resource Acquiring

When a crowdworker connects to an LC, its resource parameter statistics are acquired by the LC. The LC only acquires some variable parameters (e.g., available RAM, battery remaining, and Wi-Fi signal strength) and checks their present status. It is assumed that all the SMDs that have the client application installed possess the minimum strength of the key hardware resource parameters (e.g., CPU

frequency, CPU cores, GPU, total RAM, etc.) that are sufficient to process HVAC data, and hence, are eligible to be crowdworker. Each SMDs are uniquely identified by their IMEI numbers. The pseudocode for the SMDs' resource acquiring function is shown in Procedure 9.4.

Procedure 9.3: Sensor Data Collection and Job Creation

```
db = databaseConnect(Database)
serverSocket = createSocket(6000)  //open port for receiving data from sensor module
jobPool as List   //define a job pool as list data structure

while (True)
{
   data = serverSocket.recieve()   //reading data from the sensors
   if (isTemperatureData(data))   //check if the data are coming from DHT22. Temperature and humidity data and occupancy data are distinguished by some internal demarcation
   {
      data = data + "#" + getDate() + "#" + getTime()   //received sensor data are annotated with timestamp
      jobPool.add(data)  //add sensed data to the job pool
    }
    elseif (isOccupancyData(data))   //check if the data are coming from PIR
    {
       sensorId, occupData = data.split()   //received data are split into two fields for storing in the database
       db.executeUpdate(insert into Occu_data(O_sen_id, o_data, o_time, occupancy) values (sensorId, getDate(), getTime(), occupData))   //store the occupancy data in the database
    }
}
```

Procedure 9.4: Acquiring SMD Resource Values

```
getResPar(mobileDevice mobile, Socket serverSocket)   //acquire the value of the resource parameters from the SMD
{
   serverSocket.send("IMEI")  //request made to SMD for sending its IMEI no.
   mobile.IMEI = serverSocket.read()   //get the IMEI no. of the SMD as unique id
   serverSocket.send("Battery")  //request made to SMD for sending its available battery
   mobile.Battery = serverSocket.read()   //get the remaining battery of the SMD
   serverSocket.send("Signal")  //request made to SMD for sending its Wi-Fi signal strength
   mobile.Signal = serverSocket.read()   //get the Wi-Fi signal strength of the SMD
   serverSocket.send("freeRAM")  //request made to SMD for sending its available RAM
   mobile.freeRAM = serverSocket.read()   //get the available memory of the SMD
   mobile.Allocated = False   //the newly entrant SMDs are not allocated initially
   mobile.serverSocket = serverSocket
}
```

9.4.3.3.3 SMD Selection

As mentioned in the previous subsection that the LC checks only three dynamic resource parameters. It is assumed that all the SMDs which have the client application installed fulfil the minimum requirement of the fixed hardware resource parameters such as CPU and GPU capacity. But, as mentioned before, there are some changeable SMD properties that are crucial in a crowdworker. Since these properties are variable and change dynamically as per device usage and user behaviour they need to be checked for their instantaneous status before task

submission. An SMD would be deemed fit and selected for job submission if these dynamic parameters fulfil the threshold criteria. Table 9.13 lists the considered dynamic parameters with their threshold values and justification. The pseudocode for the SMD selection process is presented in Procedure 9.5.

Table 9.13. Threshold criteria for dynamic parameters for SMD selection

Resource parameters	Threshold values	Comment
Available RAM	≥ 100 MB	To run the client module, a minimum amount of memory is needed. The RAM requirement varies depending on the MCC application. In the HVAC application, the requirement is nominal.
Battery	≥ 60%	SMD users are always worried of the battery. If the remaining battery is low, they will not participate in MCC. Therefore, we put a check on the remaining battery level of the SMD for considering it as crowdworker.
Wi-Fi signal	≥ 2	A satisfactory degree of wireless signal strength is required for data transmission. We also assume a lower-strength signal suggests the user might be moving away from the AP, which might lead to connection loss.

Procedure 9.5: SMD Selection

```
socket = createSocket(5000) //open port for communicating with the SMD
mobileDevice   //define SMD's resource parameters
{
    IMEI as String
    Battery as Integer
    Signal as Integer
    freeRAM as Float
    Buffer as List
    Allocated as Boolean
    serverSocket as Socket
}

selectedDevice as <mobileDevice> List

while(True)
{
   serverSocket = socket.connect()   //connect SMD through port 5000 for data receiving
   mobile as mobileDevice
   getResPar(mobile, serverSocket)  //get the SMD resource parameters
   if (mobile.freeRAM ≥ 100 && mobile.Battery ≥ 60 && mobile.Signal ≥ 2)   //check for threshold criteria
      selectedDevice.add(mobile)   //maintain a list of the selected SMDs
}
```

9.4.3.3.4 SMD Allocation

As per requirement, a number of SMDs are allocated for job processing from the pool of the selected SMDs. The allocated SMDs are registered using threads. The LC runs multiple thread processes for job dispersion and result accumulation. Each thread is dedicated to and interfaced with each registered SMD through a socket connection between LC and the SMD. Since the MC requires maintaining the records of all the available crowdworkers (as discussed in Section 9.4.1.2.1 and Section

9.4.3.5), the information of the number of available and allocated crowdworkers are sent to MC. The process of SMD allocation is presented in Procedure 9.6.

Procedure 9.6: SMD Allocation

```
AllocatedDevice as <Thread>List   //create a list of threads for the allocated SMDs

SMDAllocation(int req)   //SMD allocation as per requirement
{
   count = 0   //set a counter for number of allocated SMDs
   for each mobile in selectedDevice
   {
      if (mobile.Allocated == False)   //if the SMD is free, i.e., not yet allocated
      {
         mobile.Allocated = True   //the SMD is allocated
         mobileCommnThread = MobileCommn.createThread (mobile, mobile.serverSocket)   //cre-
ate a separate thread for communication with the allocated SMD
         AllocatedDevice.add(mobileCommnThread)   //append the created thread to the list of allo-
cated devices
         mobileCommnThread.start()   //start the thread
         count++
      }
      if (count == req)  //check if number of required SMDs are allocated
         break;   //if no SMDs are required further, stop allocation
   }
   if (count < req)   //if sufficient SMDs are not available
   {
      SMDDef = req – count   // calculate the deficiency in the number of required crowdowrkers
      resourceRequestMC(LCId, SMDDef)  //send resource request to MC
   }
   resStatusMC(LCId, selectedDevice.length, AllocatedDevice.length)   //send information of availa-
ble and allocated SMDs to MC
}
```

9.4.3.3.5 Job Schedule and Dispatch

Jobs are allocated and dispatched sequentially to the respective registered SMDs for processing. Here, each SMD is represented as an independent computing resource thread. Each thread maintains a job buffer into which the scheduled jobs are queued in a round-robin fashion, as shown in Fig. 9.24. The jobs are dispatched serially from each thread to the corresponding SMDs. The data are timestamped before pushing to the job buffer. We noted this time for analysing the time-variant records of the temperature sensor data. The pseudocodes for job scheduling and dispatching are shown in Procedure 9.7 and Procedure 9.8, respectively.

9.4.3.3.6 Result Collection

The processed results are sent back from the SMDs to the respective LCs and are stored in a database. Here, the result includes the difference between the desired temperature and the actual temperature, i.e., the measure of AC temperature needs to be adjusted (increased or decreased). The database is used for performing

analysis on the information received from the SMD module to make decisions for AC controlling. Also, based on the analysis, appropriate alerts are generated. The schema of the LC database is shown in Fig. 9.25. The following five tables are defined to store the sensor information and processed results, and the fields of these tables are described in Report generation: The results are stored in the LC database for one month, which are analysed, and a monthly report is generated. However, we limited our experiment only to analysing the monthly AC faults. We aimed to identify the ACs that are more fault-prone. This report is forwarded to MC for further analysis and decision-making.

Table 9.14.

- Temp_sensor: Stores the temperature and humidity sensors' ids along with the AC to which they are associated and the room where they are placed.
- Temp_data: Stores the temperature and humidity data, and the temperature magnitude needs to be adjusted.
- Occu_sensor: Stores the PIR sensors' ids along with the AC to which they are associated and the room where they are placed.
- Occu_data: Stores the occupancy (yes or no) of a room.
- AC_fault_rp: Logs the AC fault occurrences to generate alerts.

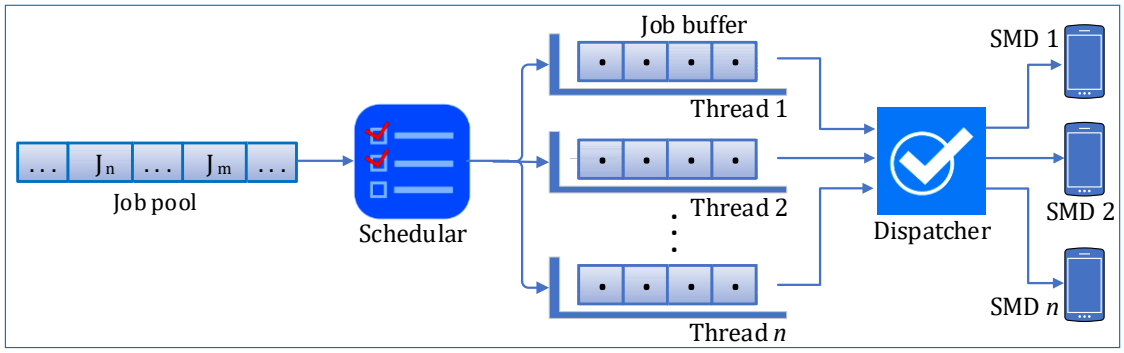


Fig. 9.24. Job scheduling and dispatching

Procedure 9.7: Job Scheduling

```
jobSchedule()  //schedule the jobs into the individual threads
{
   while(True)
   {
      for each mobileCommnThread in AllocatedDevice  //select each thread representing an SMD from the allocated list
      {
         job = read(jobPool)  //pick the jobs from the job pool
         mobileCommnThread.Buffer.add(job)  //add the jobs to the job threads
      }
   }
}
```

9.4.3.3.7 Result Analysis and Action Steps

The stored results are assessed at regular intervals, and based on the outcome, appropriate action is taken by the LC. Periodically after every one minute, an SQL script is run over the database to analyse the results. The complete processes of result analysis and actions taken based on temperature and humidity data and occupancy data are shown in Procedure 9.9 and Procedure 9.10, respectively. Based on the result analysis, the following two types of actions are taken:

i. Instruct controller module to initiate AC control mechanism: We considered the following three automated AC adjustments:
   a. Adjust AC temperature: As mentioned earlier, we intend to maintain the ideal comfort level of the room occupants by adjusting the AC temperature. The result indicates the adjustment (+ or -) needed in the current temperature to achieve the desired temperature. To make this decision, we took the average of the results of the stretch of one-minute duration for each sensor, and based on that, the final AC adjustment value (how much temperature is to be decreased or increased) is calculated by the LC and sent to the AC controller.
   b. Adjust AC fan speed: The AC fan speed is regulated as per the requirement of temperature adjustment. For example, if the value of temperature to be adjusted (-) is considerably high to cool the room as quickly as possible, the fan speed should be set to a higher measure.
   c. Switch off the AC: Based on the occupancy analysis, the AC is automatedly switched off if there is no one in the room for a certain duration. The occupancy is checked after every 5 minutes. We checked for continuous ten readings of the occupancy data. If no one is in the room for this duration, the occupancy value would be 0. In this case, the LC module sends an instruction to the controller of the respective AC to be switched off.
ii. Generate alerts and forwards to MC: Based on our calculations (Algorithm 9.1 and Algorithm 9.2), the room temperature should reach the comfort level sooner or later, depending on the difference between the current room temperature (and humidity) and the desired room temperature. However, if the room temperature does not reach close to the ideal temperature even after a

long time, we assume that the AC is not working properly. For the AC fault-checking measure, we set a flag if the temperature is less than 16°C or more than 30°C. We observed the status of this flag for continuous 10 minutes. For every abnormal temperature reading, the flag value is incremented by one. If the flag value reaches 10, it is assumed that the AC is not working properly. In that case, an alert is generated and sent to MC to be forwarded to the facility staff.

Procedure 9.8: Job Dispatching and Result Collection

```
Thread MobileCommn  //define a thread for SMD communication
{
  mobile as mobileDevice  //create an SMD instance representing the SMD
  serverSocket as Socket  //create a socket variable

  MobileCommn(mb, ss)  //initialise the SMD instance
  {
    mobile = mb
    serverSocket = ss
  }

  start()  //define the thread functionalities
  {
    while(True)
    {
      job = mobile.read(Buffer)  //pick the jobs from the SMD buffer in FIFO manner
      serverSocket.send(job)  //send the picked job to the designated SMD
      result = serverSocket.recieve()  //receive the results from the SMD after processing
      sensorId, t_date, t_time, tempData, RHData, ta = result.split()  //split the results to store in the database

      db = databaseConnect(Database)  //connect to the database
      db.executeUpdate(insert into Temp_data values(sensorId, t_date, t_time, tempData, RHData, ta))  //store the values into respective fields in the designated table
    }
  }
}
```

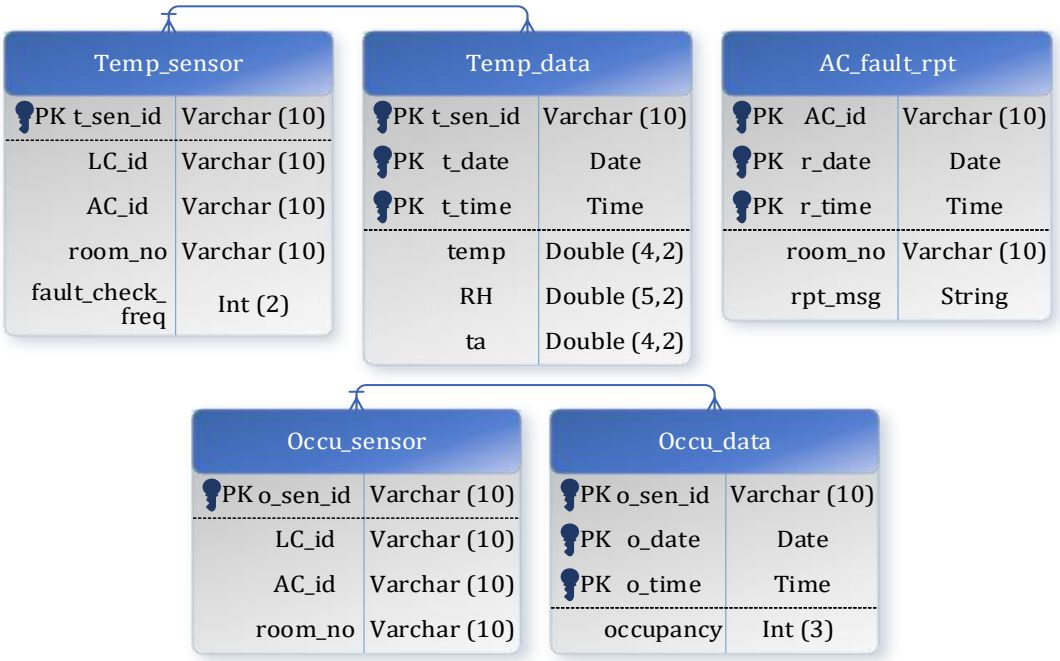


Fig. 9.25. Database schema for LC

iii. Report generation: The results are stored in the LC database for one month, which are analysed, and a monthly report is generated. However, we limited our experiment only to analysing the monthly AC faults. We aimed to identify the ACs that are more fault-prone. This report is forwarded to MC for further analysis and decision-making.

Table 9.14. Fields used in the result database

Field	Description
room_no	The room number where the particular sensor and AC are placed
AC_id	Identifies each AC uniquely
LC_id	Identifies each LC uniquely
t_sen_id	Identifies each DHT22 sensor
t_date	The date of temperature and humidity acquisition
t_time	The time of temperature and humidity acquisition
temp	Sensed temperature data
RH	Sensed humidity data
ta	Temperature to be adjusted
fault_check_freq	Counting for fault checking period
o_sen_id	Identifies each PIR sensor
o_date	The date of occupancy acquisition
o_time	The time of occupancy acquisition
occupancy	Sensed occupancy data
rpt_msg	Alert message
r_date	The date of alert generation
r_time	The time of alert generation

Procedure 9.9: Result Analysis and Action Taken by LC Module Based on Temperature and Humidity Data

```
db = databaseConnect(Database)
Tsensors[] = db.executeQuery(select t_sen_id from Temp_sensor) //consider all the temperature
```

```
and humidity sensors under the considered LC
serverSocket = createSocket(7000) //open port to send temperature-based control instruction to AC
controller

while (True)
{
   wait(60)  //wait for one minute for average analysis
   for each Tsensor in Tsensors[]   //consider each temperature and humidity sensor in the room
  {
     T1 = getTime()   //get the current time
     T2 = T1 – 60  //get the start off clock time of the immediate last one minute

     avgTa = db.executeQuery(select avg(ta) from Temp_data where t_sen_id = Tsensor and t_time
between T1 and T2)   //calculate the average of adjustable temperature (ta) for a period of one mi-
nute for a particular sensor
     ACId = db.executeQuery(select AC_id from Temp_sensor where t_sen_id = Tsensor)   //get the
AC id associated to the particular sensor
     RoomId = db.executeQuery(select room_no from Temp_sensor where t_sen_id = Tsensor) //get
the room no. where the particular sensor is installed

    if (|avgTa| ≥10)   //check the deviation between the current and desirable room temperature to
set the AC fan speed
       fanSpeed = High   //if the deviation is more set the fan speed to high
    elseif (9 ≥ |avgTa| ≥ 6)
       fanSpeed = Medium   //if the deviation is moderate set the fan speed to medium
    else
      fanSpeed = Low   //if the deviation is less set the fan speed to low

    serverSocket.send(ACId, avgTa, fanSpeed) //send the AC temperature and fan speed to be ad-
justed to the AC controller

     /* check for abnormalities of AC functioning */
     avgTmp = db.executeQuery(select avg(temp) from Temp_data where t_sen_id = Tsensor and
t_time between T1 and T2)   //check the average room temperature returned by a particular sensor
for last one minute
     fault_check_freq = db.executeQuery(select fault_check_freq from Temp_sensor where t_sen_id
= Tsensor)   //get the frequency of AC faults connected to the particular sensor
     if (AC == "ON")   //check whether the AC is running
     {
        if (avgTmp > 30 || avgTmp ≤ 16)  //check for abnormality of the room temperature
           fault_check_freq = fault_check_freq + 1   //if the room temperature is abnormal AC fault
frequency in incremented by one
        else
           fault_check_freq = 0   //if the room temperature is normal set AC fault frequency to zero
     }
     db.executeUpdate(update Temp_sensor set fault_check_f = fault_check_freq where t_sen_id =
Tsensor)   //update the AC fault frequency in the database

     if (fault_check_freq ≥ 10)  //check for fault for continuous 10 minutes
     {
        msg = "AC is not working properly. Room temperature is" + avgTmp + "degree for last 10
minutes"  //build the alert message
        Time = getTime()   //time of alert
        Date = getDate()   //date of alert
        SendMssgCenter(msg, ACId, RoomId, Date, Time)  //forward the alert message to MC
       db.executeUpdate(insert into AC_fault_rpt values (ACId, RoomId, msg, Date, Time))   //log for
long-term AC fault report generation
     }
   }
}
```

Procedure 9.10: Result Analysis and Action Taken by LC Module Based on Occupancy Data

```
db = databaseConnect(Database)

Osensors[] = db.executeQuery("select o_sen_id from Occu_sensor")   //consider all the occupancy
```

```
sensors under the considered LC

serverSocket = createSocket(7000)  //open port to send occupancy-based control instruction to AC controller

while (True)

{

    wait (300)  //wait for 5 minutes

    for each Osensor in Osensors[]  //consider each occupancy sensor in the room
    {
       T1 = getTime()  //get the current time
       T2 = T1 – 300  //get the start off clock time of the immediate last five minutes
       ACId = db.executeQuery(select AC_id  from Occu_sensor where  o_sen_id = Osensor)  //get the AC id associated to the particular occupancy sensor
       os_sum = db.executeQuery(select sum(occupancy) from Occu_data where o_sen_id = Osensor and o_time between T1 and T2)  //get the cumulative value of occupancy reading for five minutes

       if (os_sum == 0)  //check if cumulative occupancy value is zero

         serverSocket.send (ACId, "OFF")  //if cumulative occupancy value is zero turn off the AC
       else

         serverSocket.send (ACId, "ON")  //if cumulative occupancy value is not zero turn on the AC
    }
}
```

9.4.3.4 *Controller Module*

The AC in each room is controlled by the same NodeMCU (ESP8266 Wi-Fi chip), which is used for housing the DHT22 and PIR sensors. The NodeMCU fitted with infrared LED acts as a remote that mimics the infrared signals of an AC remote controller for operations like a) temperature adjustment, b) fan speed regulating, and c) AC switch on/off. The process for AC control by NodeMCU is given in Procedure 9.11.

Procedure 9.11: Controlling AC

```
clientSocket = createSocket()
clientSocket.connect(LC_IP, 7000)  //connect the LC at port 7000 for data receiving
ACIds[] = getConnectedACId()  //get the ACs connected to the controller

while (True)
{
    msg = clientSocket.receive()  //receive AC control instruction from LC
    ACId, data, fanSpeed = msg.split()  //extract the control instruction for the AC to take action
    for each id in ACIds  //consider each AC in connected to the controller in the room
    {
        if (id == ACId)  //identify the particular AC for which the instruction is meant
        {
            if (isNumeric(data))  //check for the instruction data type
            {
                adjustTemp(ACId, data)  //if the value of data is numeric, the AC temperature would be adjusted by data
                adjustFanSpeed(ACId, fanSpeed)  //AC fan speed is regulated
            }
            elseif (data == "OFF")  //check if the instruction is equal to Off
                offAC(ACId)  //turn off the AC
            elseif (data == "ON")  //check if the instruction is equal to Off
                onAC(ACId)  //turn on the AC
```

```
        }
    }
}
```

9.4.3.5 MC Module

The MC is a centralised coordinator of the whole MCC system deployed in the building. It is responsible for the overall management of the MCC that includes the LCs and the SMDs. The primary responsibilities of the MC are:

- **Fault tolerance:** The MC ensures the fault tolerance due to the non-availability of the SMDs. The MC keeps track of the number of available and allocated crowdworkers for each LC. If an LC does not have the required number of crowdworkers, it sends an SOS to the MC, which, in turn, suggests a suitable LC in the neighbourhood of the low-resourced LC to which it can send the tasks temporarily and get them executed.
- **Communicating with the cloud:** If there are no sufficient crowdworkers available in the building, the MC sends the sensor data forwarded by the LCs to the cloud. It retrieves the results from the cloud and redirects them to the respective LCs.
- **Alert notification:** Each LC generates event notifications in case of abnormalities in AC functioning. These notifications are forwarded to the MC, which in turn send them to the appropriate authorities (e.g., facility staff). A sample error report is shown in Fig. 9.26. This error report was generated using dummy data because, while experimenting, we did not experience any faulty condition.
- **Analytics:** The MC receives the periodical (e.g., monthly) reports from each LC. It performs data analysis on these reports to find patterns and actionable information.

Procedure 9.12 and Procedure 9.13 present the processes of resource management and alert notifications submodules of the MC module.

Procedure 9.12: MC Module – Resource Management

```
socket = createSocket(8000)  //MC opens a socket at port 8000 for communicating with LC
while(True)
{
   serverSocket = socket.connect()  //connects to an individual LC on its request
   CommnThread = faultTolerance.createThread (serverSocket)  //create a communication thread
for the LC
```

```
    CommnThread.start()
}
Thread faultTolerance
{
   serverSocket as Socket   //create a socket variable

   faultTolerance(ss)   //initialise the server socket
   {
      serverSocket = ss
   }
  Start()
 {
     LCId, SMDReq = serverSocket.read()   //receive resource request from a LC in case of SMD unavailability

     LC_IP = searchSuitableLC(LCId, SMDReq)   //look for other nearby LCs with adequate SMDs

      if (LC_IP == Null)   //if no such LC found
      {
         cloudInterface = connectCloud()   //connect to the cloud
         while(job = serverSocket.read())   //task received from LC
         {
            cloudInterface.send(job)   //send the task to the cloud
            result = cloudInterface.read()   //receive the results from the cloud
            serverSocket.send(result)   //send the results to the concerned LC
         }
       }
       else
          serverSocket.send(LC_IP)   //send the address of the LC to whom the requesting LC would send the task
     }
}
```

Procedure 9.13: MC Module – Alert Notification

```
socket = createSocket(9000)   //open a socket at port 9000 for communicating with LC
while(True)
{
    serverSocket = socket.connect()   //connects to an individual LC on its request
    msg = serverSocket.read()   //get the alert message from the LC
    saveNotice(msg)   //log the alert for future analysis
    alertNotification(msg)   //send the alert notification immediately to the facility staff
}
```

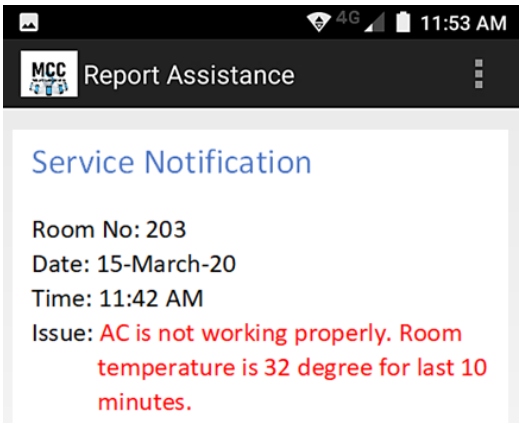


Fig. 9.26. A sample error report generated by LC/MC

9.5 Comparing MCC-Edge with Commercial Edge and Cloud Computing

To validate the advantages of MCC we compared it with other computing

architectures such as cloud and edge computing in terms of cost, latency and energy consumption in HVAC system scenario. To compare these three architectures, we made the following considerations:

- We excluded the devices (e.g., sensors, Node MCU, actuators, switches, routers) that are generally commonly employed by all the three architectures for implementing an automated HVAC control system.
- We considered only those devices that are associated with the specific architecture.
- We considered a four-storey building wholly covered by the HAVC system.

We used a Raspberry Pi as the MCC coordinator; hence it is included in the comparison. Similarly, for implementing HVAC controls system through proprietary edge computing, edge devices like a single unit of edge server and edge gateway are required, in addition to other usual hardware and networking components. Likewise, to implement HVAC control system through cloud, a local gateway (computer) is required for accumulating data acquired from different sensors and forwarding them to cloud. Since, the data centres play a primary role in cloud services, the cost incurs for cloud services and the energy consumption of data centre are also included in the comparative study.

The three architectures follow different technologies and frameworks for operation. Moreover, edge and cloud computing can be implemented in various ways, resulting in multiple options for comparison. Thus, for a justified comparison among the three technologies many considerations are made, and are discussed wherever applicable. We considered only those products of which the comparable data were available openly.

9.5.1 Cost

Infrastructural cost is a great factor for adopting and implementing technologies in largescale. Specifically, for computing infrastructure, not only its usefulness but also the cost (both fixed and variable) involved need to be estimated for employing in organisational levels. Keeping that in mind, we compared the three computing architectures in terms of infrastructural and service cost for smart HVAC

implementation. For comparison, we considered only the cost of MCC devices, some commercial edge computing solutions and cloud services that are applicable in respective architectures to process the HVAC data streams.

We considered Dell Edge Server and Dell Edge Gateways, Lenovo Edge Server and Cisco Edge Series Switch as sample edge devices. On average, these devices cost approximately $3,045. Likewise for cloud, a local gateway computer, Dell EMC PowerEdge R340 was considered as a sample product. Further for processing data in cloud, we considered Amazon EC2, Microsoft Azure, Google cloud, and Heroku as sample cloud services. We considered a cloud service of 16 GB RAM and octa-core virtual CPU and the cost was calculated on hourly basis which resulted in an average yearly cost of approximately $4,809. In case of MCC-edge, the public-owned SMDs are used for computational tasks, hence cost of these was not taken into consideration. The LC is implemented on Raspberry Pi, per unit cost of which is approximately $35[52]. It is the only expense in MCC architecture, not considering the cost of the Wi-Fi APs and associated networking, assuming they are commonly available in a modern office or residential building. An additional cost of the MC should be taken into account depending on the MC deployment device. We used a Raspberry Pi, hence, another $35 would be added. Cost comparisons are given in the Table 9.15. It is very obvious that among the three approaches, the MCC-edge involves the least cost.

Table 9.15. Comparing MCC with cloud and edge computing in terms of cost

Proprietary edge		**Cloud services**		**MCC-edge**	
Product	*Cost*	*Service*	*Cost*	*Device*	*Cost*
Dell Edge server PowerEdge T340[53] server	$1,029.00	Amazon EC2 (vCPUs: 8, RAM: 32 GB)	$0.384/hour, $3363/ year (24*7) [764]	Raspberry Pi (1 per floor i.e., total 4 units)	LC: $35*4 + MC: $35
Dell Edge Gateway 5100[54]	$1,928.74	Microsoft Azure	$0.437/hour, $3828/ year (24*7)[55] [764]		
Lenovo Think System SE350[56] edge server	$2,589.00	Google cloud	$0.268/hour, $2347/ year		

[52] https://www.tomshardware.com/how-to/raspberry-pi-buying-guide
[53] https://i.dell.com/sites/csdocuments/Product_Docs/en/dell-emc-poweredge-t340-spec-sheet.pdf
[54] https://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Spec_Sheet_Dell_Edge_ Gateway_5000_Series.pdf
[55] https://www.parkmycloud.com/cloud-pricing-comparison/
[56] https://lenovopress.lenovo.com/datasheet/ds0088-thinksystem-se350

			(24*7)[57] [764]		
Cisco Edge 300[58] series switch	$545.95	Heroku cloud	$3000/year[59]		
		Average cloud service cost (yearly)	$3134/year		
		Dell EMC PowerEdge R340 (local gateway computer)	$1675.00		
Approximate total (server and gateway)	$3,045.00	**Approximate total**	$4,809.00	**Approximate total**	$175.00

9.5.2 Latency

Latency is one of the most important metrics to be considered for real-time applications. Here in the HVAC case scenario, we considered latency as the time taken to send HVAC data to processing unit, time for processing the data and return back the result for control decision. We have compared the latency among the three considered computing architectures i.e., edge, cloud and MCC-edge. The average latencies for each case are given in Table 9.16.

To estimate the latency in cloud, we deployed the SMD module in cloud. There are many commercial cloud services available. Among them, we considered the three most popular cloud services, namely AWS, Azure and Heroku cloud as sample for latency test case study. The latency time found varying from 49 ms to 3017 ms. For most of the cases the latency remains in the time frame of 1000 ms or more. The big reason for higher latency is due to network constraint and far distance of the data centres. Similarly, to estimate the latency in edge, the SMD module was deployed on an edge server. Due to infrastructural constraints, instead of using a proprietary edge server, we deployed a high-end server as an edge server. The latency for the edge was found in the range of 20 ms to 35 ms. While, in case of MCC-edge, the latency was approximately 75-100 ms, which includes the transmission and job distribution time.

As expected, the latency for MCC is much lesser than cloud computing, but higher

[57] https://www.parkmycloud.com/cloud-pricing-comparison/
[58] https://www.connection.com/product/cisco-edge-300-series-switch/cs-e300-k9/15017294
[59] https://www.heroku.com/pricing

than traditional edge systems. This is due to the fact that the edge devices are dedicated and powerful computing nodes whereas in case of MCC, the SMDs are less powerful than the edge devices and also, the task are executed only when the SMD CPUs is free. Furthermore, distributing the jobs to different SMDs also involves some extra latency. Relatively, the cloud computing involves higher latency due to transmission delay and network traffic. The data need to travel a number of intermediate nodes before they get processed at a data centre. In contrast, both in edge computing and MCC-edge, the data need to travel maximum of two to three nodes before it gets processed and hence saves time resulting in lower latency than cloud.

Table 9.16. Comparing MCC with cloud and edge computing in terms of latency

Proprietary edge	Cloud services		MCC-edge
20-35 ms	AWS	49-3017 ms[60]	75-100 ms
	Azure	70-316 ms	
	Heroku	2000 ms	

9.5.3 Energy Consumption

In the recent years, power consumption has become an important factor for computing resources. Energy efficient frameworks not only save money but also are good for the planet. They help in reducing air and water pollution caused by energy consumption and generation and minimizes the harmful impacts on the ecosystem.

For the comparison, we considered the energy consumption of different devices used each of the approach, as shown in Table 9.17. The energy consumption was calculated in watt hour unit. In cloud architecture, a data centre consumes massive energy than edge computing and MCC. Further in cloud, the data need to traverse hundreds of miles from the point of data generation before to be processed. Whereas in MCC and edge computing, the applications run close to the data generation and consumption; hence data traversing is reduced, which in turn minimises energy consumption [35]. The energy consumptions of an edge computing server and an edge gateway are very less. While in MCC setup, the total energy

[60] https://ping.psa.fun/

consumption of Raspberry Pi and an average SMD is negligible compared to cloud and edge devices. So, it can be concluded that MCC is the most energy efficient approach compared to edge or cloud computing.

Table 9.17. Comparing MCC with cloud and edge computing in terms of energy consumption

Proprietary edge		**Cloud services**		**MCC-edge**	
Device	*Energy consumption*	*Cloud component*	*Energy consumption*	*Device*	*Energy consumption*
Edge server PowerEdge T340	2.232 kWh[61]	Data center [765] [766]	13.9 mWh	Raspberry Pi	0.0576 kWh[62]
				5 Raspberry Pis	0.288 kWh
Dell Edge Gateway 5100	1.56 kWh[63]	Dell EMC PowerEdge R340	8.4 kWh[64]	SMD	0.015 kWh
				10 SMDs	0.15 kWh
Total energy consumption	3.792 kWh	**Total energy consumption**	(8.4 + τ) kWh	**Total energy consumption**	< 0.438 kWh
τ is the energy consumption for processing a client application in a data centre					

9.5.4 Environmental Hazards

In Section 1.2, we mentioned the environmental threats of computing devices and the need to minimise them. We also discussed in Section 1.6, considering the environmental benefits, how MCC can be a suitable option for this. We also briefly discussed the environmental benefits of MCC-edge in Section 9.1.4.3. We understand that the SMDs will be used by the users anyway and if these existing devices are utilised effectively there will be significant reduction in additional manufacturing materials as required for dedicated computing devices as in case of edge and cloud. Also, due to the much smaller size, SMDs have much less negative environmental impact and e-waste than larger computers. Moreover, non-requirement of dedicated power backups and cooling systems results in significantly reduced carbon emission. The environmental advantages of MCC-edge have been corroborated by a statistical comparison with traditional edge and cloud, as presented in Table 9.18. In the table, the operational CO_2 emission for SMDs and cloud are for general uses. The actual figures accounting to processing MCC tasks only would be

[61] https://www.servethehome.com/dell-emc-poweredge-t340-review-a-high-end-low-cost-server/4/
[62] https://www.pidramble.com/wiki/benchmarks/power-consumption
[63] https://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Spec_Sheet_Dell_Edge_ Gateway_5000_Series.pdf
[64] https://i.dell.com/sites/csdocuments/Product_Docs/en/poweredge-r340-spec-sheet.pdf

much lower.

Table 9.18. Environmental hazards comparison of MCC-edge with cloud data centres and edge infrastructure

Environmental hazard aspects	Cloud data centre	Edge devices	MCC-edge (SMD)
Operational CO_2 emission	45.12 trillion kg (overall) [767].	0.3 - 47.41 kg (per device/year) [768].	Raspberry Pi: 8.322 kg (per unit/year) [767]. Smartphone: 63 kg (for 1hour usage/year) [769].
CO_2 emission in the manufacturing process	171,630 kg CO_2 [42]. Around 0.3% of overall carbon emissions [5].	22 to 153 metric tons per year for all edge devices globally [768].	16 kg [45] to 55 kg [46] [770] CO_2 equivalent for an average smartphone.
Other environmental hazards related to manufacturing	The hazardous materials used in manufacturing different components used in data centre includes PVC, PCBs, Hg, Pb, Cd, Be, Cr(VI), Sb, BFRs. Besides, other harmful elements used in data centres include ethylene/propylene glycol for cooling systems, diesel fuel for backup generators, lead-acid in UPS batteries, and compressed gases for fire suppression [771].	The hazardous materials used in manufacturing of different edge devices includes PVC, PCBs, Pb, Cd, Be, Cr(VI), Sb, BFRs, etc. [771].	Various hazardous metals such as Cd, Pb, Li, Hg, BFRs, Cl, Cr(VI) and plastics are used in mobile manufacturing [45].
E-waste generation	32360 metric tons e-waste in 2018.	4.7 million metric ton e-waste by small IT equipment in 2019 [772].	More than 5 million metric tons in 2019 [773] [774].
E-waste decomposition	The e-waste generated from data centres include computers, network communication devices, and cables and other power systems. The generator, UPS and other electrical equipment have long lifecycle (5-10 years) and are repurposed. While the computers, communication device, cables are redeployed, remarketed, recovered and recycled. The recycle companies are mainly involve in collection of e-waste from data centers.	The e-waste generated from edge device include switch, routers, edge gateway, cables, and UPS. The edge devices have long lifecycle (10 years or more) and are often repurposed and remarketed. Discarded edge devices are recycled to obtain steel, plastic, and PCB. PCB are source of precious metals like Au, Ag, Cu, Ta, Pd. PCB are recycled according to standard practice like reusing ICs, cable circuit, large metal	Mobile devices have an average life span of approximately four years. As a reason, huge mobile devices are discarded as e-waste. 80% mobile devices are recycled using standard recycling policy. Plastic and glass are recycled for reuse. PCBs are recycled to obtain precious metals such as Au, Ag, Cu, Pd, Ta and other mobile device parts like microphone, speaker, connector, metal fittings are recycled to get steel, Cu and other

	The material which are recovered in huge quantity includes plastic, Al, and steel. The other material which are recovered in small quantity are Au, Ta, Pb, Cu. While the hazardous wastes are decomposed as per the e-waste decomposition policy [775] [776].	component, plastics are sent for further treatment, glass are sent for refining while battery and Hg are land filled [777] [778].	metals. The batteries that cannot be recycled are landfilled. Though the quantity of e-waste by a single mobile device is small, the total amount of e-waste can be substantial [779] [778].

9.5.5 Evaluation

From the comparative study based on cost, latency, energy consumption and e-waste generation it can be concluded that MCC architecture is sustainable and environment friendly as compared to proprietary edge and cloud infrastructure for small and medium computing requirement such as smart HVAC.

This cost factor is an important aspect for an organization when the question of sustainability arises. It is very much evident from the cost comparison that the MCC architecture bears negligible cost for implementation and is thus lucrative to organization as compared to other two architectures. It was observed that the latency of MCC is significantly lower than cloud but slightly higher than edge. However, this smaller latency is bearable for the soft real-time applications such as HVAC. Further in MCC, the extra energy consumption accounting to the computation is negligible which is not only eco-friendly (less carbon emission) but also economical for the organizations in long run, compared to the edge and cloud. Moreover, as in MCC the devices are utilised which are already being used by the users there will be no need for exclusive or additional device production. This not only minimises the environmental hazards arises due the manufacturing process but also ensures no extra e-waste is contributed to the environment unlike the dedicated computing and allied devices as in case of edge and cloud computing architectures.

All these advantages make MCC-edge a low cost, greener, and environment friendly computing solution and a feasible alternative to the traditional edge and cloud computing.

9.6 Discussion

In this section, we cover a couple of points in which, we assume, the readers might be inquisitive of. Below, we try to shed light on a couple of such apparent reservations.

How would the system materialise if no SMDs were available? In MCC, since the computing devices are mobile and not dedicated, their availability is uncertain. However, the primary requirement of MCC is the availability of sufficient SMDs. For a global MCC like BOINC, the availability of the SMDs is not that bothersome because they are connected through the internet, and the physical location of an SMD does not count for being available. But in the case of a local or campus-based MCC, the SMDs need to be physically available since they are connected to the coordinator through a short-distance local network. In this case, users' presence and mobility are of big concern. In an office environment, it is assumed that the users remain at their designated places most of the time during office hours. Hence the movement and chance of unavailability of the SMDs will be minimal during office hours. Even if an LC does not have sufficient SMDs under it, the MC will handle it, as discussed in Section 9.4.3.5. Nevertheless, the concern is for the non-office hours and the holidays, when no SMD would be available in the building. It can be well assumed that there would be no requirement of real-time controlling of the HVAC system; hence it is not required real-time processing. In this case, the HVAC operations data need to be processed only for long-term monitoring and analysis purposes. This can be done on the MC itself if the data streaming and hence processing interval is set to a longer period. Otherwise, a cloud service can also be availed.

Why not to use a cluster of Raspberry Pi instead of MCC? It can be argued that why not use a cluster of multiple Raspberry Pi [780] [781] instead of a cluster of SMDs. It seems a valid argument considering the uncertainty of SMDs and the increasing capacity of every updated version of Raspberry Pi. But there are some fundamental issues with Raspberry Pi, as mentioned in the following, due to which it is not a wise option to implement the whole edge computing solution on it:

- With a small number of cores, multitasking is limited in Raspberry Pi.
- Raspberry Pi has a limited amount of usable memory. For heavy tasks, it uses an SD card, and frequent swapping between RAM and SD card causes thrashing and affects the throughput.
- Raspberry Pi is not suitable for computing-intensive applications.
- Data transmission is slower. Raspberry Pi has only a 100Mbps Ethernet port and a single controller for USB and Ethernet, which divides the speed.
- Raspberry Pi is prone to heat as there is no heatsink. Hence, continuous compute-intensive processing makes the RP hot and early wear out and even damage SSD.
- Raspberry Pi is constrained in terms of flexibility and scalability.
- Raspberry Pi has reliability issues. It has no fuse protection. So, there is a high chance of the board getting damaged.

9.7 Limitations and Further Scopes

Since in this work, our aim was to present a working prototype of the proposed system, we tried to keep the things uncomplicated as far as possible. In this section, we submit the aspects that we intentionally or insistently overlooked. These unaddressed facets can be incorporated or implemented in the future enhancement of the system. We also list out some interesting points which can augment the MCC-edge enable HVAC or the MCC-edge in general further. For a better construal, we divided the deficiencies of this prototype and the scopes for further enhancements into two categories, as discussed in the following subsections.

9.7.1 MCC-Edge

Below, we list down the lacking aspects and the further development scopes concerned to the proposed MCC-edge in general:

- The minimum resource capacity of each SMD requisite for the job to-be-scheduled is required to be determined. Also, for wholesome job allocation, the number of SMDs required under each LC individually needs to be estimated, depending on the size of the room and the number of sensors.
- We did put aside designing and implementing the analytics part in the MC

module as future work.

- Although in the core MCC-HVAC model, we suggested interfacing between MC with the cloud, in the proof-of-concept, it was not implemented.
- Instead of Wi-Fi, connecting the sensors with the LC using some energy-aware low-range protocols like ZigBee, BLE, etc., might be more energy-efficient [433] [782].
- We assumed that the computing resources (hardware and software) do not influence the SMD selection if all pass the threshold criteria. But this is a very naïve assumption. In practice, SMDs vary vastly as per their resources, and accordingly, the throughput would differ [480]. This varying capacity should be considered for crowdworker selection.
- Though the scenario of crowdworker unavailability is considered in the design, due to complexity, it was not implemented in the MC module. The fault tolerance aspect of MCC should be explored and implemented in the LC and MC levels.
- For simplicity, we did not set any task execution deadline at the crowdworker's end. However, in a hard real-time system, this needs to be enforced and administered.
- The computation time of different MCC modules can be further reduced by fine tuning the programs. Ideally the turnaround time should be very close to the computation time on a dedicated edge device.
- We did not include any fault-checking mechanism, i.e., what to be done if the crowdworker fails to send the results to LC. Furthermore, the LC itself might be unavailable (due to network failure) or out of order (due to the reason mentioned in Section 9.6). However, for critical applications, it is crucial to implement fault handling mechanisms.
- As mentioned in Section 9.6 that Raspberry Pis may not be reliable; therefore, implementing the LC on a Raspberry Pi would make the MCC system unstable. Instead of relying only on the Raspberry Pi, as a backup, the LC module can also be implemented on one of the crowdworkers among the SMDs connected to the same AP. The SMD should be selected or elected as a leader, satisfying some predesignated criteria [783].

- The presented MCC-edge prototype is not general purpose. It is neither application nor job neutral. The tasks (instructions) are predefined and hardcoded in the client module. However, a generalised MCC system should be open and flexible, which is capable of carrying different types of jobs. This requires sending the tasks that include the data and instructions to the crowdworkers instantaneously by the coordinator. The client program should abstract the heterogeneity in the type, format, and size of the tasks and the associated data.

9.7.2 MCC-Edge Enabled HVAC

In the following, we identify the features that could be incorporated specifically to the MCC-edge enabled HVAC, leading to further extensive development of the demonstrated prototype:

- In our prototype experiment, only the front-end of the HVAC, i.e., AC and air vents, are considered. The backend part of the HVAC that includes the operations of the boiler, chiller, cooling tower, heat exchangers, etc., is not considered.
- Being in the demands, the sensors are manufactured by different companies based on different locations. Hence, the sensors are designed and tested under dissimilar conditions. Using them in other conditions might cause a slight deviation from the actual readings. Likewise, the DHT22 sensor also might exhibit a bit different characteristic in the tropical region. To avoid this error, the sensors should be calibrated by comparing the temperature and humidity value obtained using some authentic manual measurement such as ASTM-117C [784] and ASTM E337-84 [785] methods. However, in this experiment, we did not perform any such calibration.
- The PIR sensor that is used to detect occupancy has a major disadvantage when the occupant remains stationary for a certain period of time. Since the PIR sensors are designed to detect deviations of positions (or movement) if the occupant remains stationary at a location, the sensor will not perceive any change in the movement, leading to the failure in occupancy observation. Instead of a motion sensor, CCTV video processing can be used to detect real-time occupancy [397].

- Another problem with the PIR sensor is that it cannot recognize the increase in occupancy. We intended to auto adjust the AC fan-speed based on occupancy level that would allow better humidity controlling, but could not due to this limitation. CCTV-based occupancy detection will be helpful in this case also.
- We only considered the occupancy presence. However, there are several factors that could be considered as discussed in the following [752]:
  - Occupant's position: The exact position of the occupant(s) in the room (to determine the distance from the AC vent based on which the fan speed should be regulated). For this, radio frequency identification (RFID) can be used [786].
  - Occupancy density: Maintaining the same AC temperature and fan speed might not offer the equal comfort when the number of the people varies in the room significantly. Therefore, occupancy density based HVAC controlling would be a better choice for superior comfort. The PIR sensors are not sufficient for this. Some Wi-Fi devices and camera sensors could be useful to assess the number of occupants in the room.
  - Occupants' activity: Similar to knowing the number of people in the room, the comfort level can be set ideally based on what they are doing [787]. Different activities would have different perceived heat levels [788].
  - Personalised comfort: Each person's perceived comfort level varies, though slightly. It can be possible to set comfort levels if each occupant can be identified with their predicted comfort levels. However, this is not applicable for multioccupancy.
  - Movement trajectory: Tracking one's movement trajectory within the building would allow identifying moving-from and moving-to locations. This information can be used proactively to set the comfort level at the destination room for a particular occupant, especially in a residential building where the scope and room areas of the occupant's activities are limited or do not change much in comparison to a commercial building [789]. However, this is also not applicable for multiple persons unless they move together.
- The occupancy sensors could be more effectively utilised by exploiting

machine-learning techniques and dynamic analysis for predicting the occupancy patterns of the rooms, based on which the HVAC can be controlled proactively. This would result in better occupancy comfort and considerable energy savings. Vision-based techniques can be used for detecting occupant numbers, locations, and activities [790]. To predict occupancy presence and occupants' activities, machine learning models that use camera data can be helpful [790] [791] [792]. Towards this, various radiofrequency-based sensor networks such as RFID, Wi-Fi, WLAN, Bluetooth, and Zigbee can be utilised [793] [794].

- The proposed system can be integrated with real-time local weather information, accordingly which the thermal comfort thresholds would be adjusted automatically and proactively.
- We generated the basic report from the LC for identifying the ACs that more frequently falters. However, suitable data logging would enable the LCs to generate more useful reports such as:
  - Running duration of each AC, room-wise, based on which the power consumption is estimated.
  - Which rooms are more crowded and at which time and for how long?
  - Working status of the ACs.
- The responsibilities of the MC can further be extended by storing every HVAC operational detail for longer periods and analysing them to assess the HVAC operations and understand the requirements better.
- In standard HVAC systems, humidifier/dehumidifiers are used to maintain the ideal humidity in the room. A humidifier/dehumidifier unit can work in conjunction with the AC but can be operated independently, offering more control in maintaining an optimized humidity level. If the room temperature is lowered, the moisture holding capacity of air is reduced, resulting in condensation of the excess water vapour which is carried away outside by the dehumidifier unit of a standard HVAC. This lessens the humidity in the room. But in our experiment, we were restricted and did not have a humidifier/dehumidifier. Hence, we aimed to maintain the ideal dew point in the room by adjusting the room temperature. This might have compromised the optimal comfort level

slightly.

- The comfortable RH level varies with the season. For instance, in summer, cooling by evaporation of body moisture is necessary in order to expel the heat; hence, it is suggested to keep the RH below 60%. On the other hand, in winter, a higher level of humidity can be tolerated; hence the RH level can be up to 80%. However, we did not consider this fact in our calculation. Inclusion of this adjustment in the calculation of dew points and heat index would offer more accurate operations of the HVAC.
- We straightforwardly set the AC temperature to the lowest i.e., 16 if the heat index is above 27°C. Here, a regressive method could be adopted to set the AC temperature as per different heat index levels mentioned in Table 9.6. This would allow to shun imposing unnecessary load on the AC and hence, saving energy.
- The HVAC operations can be made more energy-efficient by considering the enthalpy change [795] [796]. Enthalpy is a function of temperature and humidity and signifies the total change in heat content. In the context of a HVAC system, a change in enthalpy is deemed equivalent to the heat absorbed or released. For cooling, the heat is absorbed from the airstream and transferred to somewhere, often outdoors, while the cool air is distributed to the room. The mechanical cooling process uses electricity for transferring the heat from room air to the coolant. Whereas another energy-efficient method known as free cooling makes the most of favourable conditions outdoors to introduce air with greater capacity for heat absorption instead of the recycled air [797] [798].

9.8 Summary

In this chapter, we presented a proof-of-concept of MCC demonstrating the feasibility of its practical applications We proposed to use MCC as a viable and sustainable alternative to proprietary edge computing. For this, we considered the HVAC system of an office building as a use case scenario.

HVACs are attributed to the foremost energy consumers in commercial and residential buildings. The energy consumption can be reduced to some extent while offering optimal comfort to the occupants if the building HVACs are controlled

smartly. However, the smart HVACs are equipped with several sensors which generate a huge amount of data. For real-time response, these data are required to be processed immediately for which, edge computing is the favourable option. But proprietary and dedicated edge computing solutions incur financial and energy costs. In view of that, we considered utilising the SMDs available in the building premises to build a dynamic computing environment and employed it as a local edge computing setup.

We attempted to maintain the ideal comfort level for the occupants in a room by automatedly controlling the AC settings. In the setup, we used temperature and humidity sensors and occupancy sensors. To measure the ambient comfort level precisely, we calculated the dew point and heat index using the room temperature and humidity. Based on the occupancy data, the fan speed of the AC was regulated, and in case of no occupancy, the AC was automatedly turned off.

We used DHT22 as temperature and humidity sensors and PIR sensors to assess the occupancy in a room. All the required computations were conducted on the SMDs. We used a Raspberry Pi based single-board computer (SBC) to implement the local coordinator (LC), which is responsible for collecting the sensor-generated data, composing independent jobs, selecting suitable SMDs, dispatching the jobs to the SMDs, collecting the results from the SMDs, analysing the results, send instructions to the controller, and generate error alerts in case of abnormality. The LC module was programmed using Java. The controller was implemented using a NodeMCU microcontroller, which was programmed using LuaScript. The overall MCC is monitored by the centralised MCC coordinator (MC). MC monitors the resource availability under each LC. In case of unavailability of SMDs under an LCs, MC refers the job to other LCs or to the cloud.

To establish the advantage of MCC as edge computing, we compared it with the commercial edge computing solutions and also with cloud services in terms of cost, energy consumption, latency and environmental impacts. In all the parameters, MCC has shown to have significant advantages over the other two. MCC is most energy efficient, environment friendly and economical. Only the latency of MCC is slightly higher than edge computing. But this difference is not significant enough

to discourage using MCC for real-time applications. In conclusion, MCC is definitely a lucrative and feasible sustainable computing solution that can be used as edge infrastructure to cater the needs of data- and computing-intensive real-time applications.

We presented the technical approach to bootstrapping the operations of each module with a bit of extra emphasise on the LC module, as it acts as the nucleus of the MCC. We depicted the minute details of designing, developing, and implementing the proposed prototype. To aid the interested researchers and programmers in recreating and deploying the MCC-edge of their own, the procedures of each designed module (sensor, SMD, LC, controller and MC) are presented in sufficient specifications and descriptions. All the modules are built from a curated set of easily available open-source and commodity tools and libraries that are being popularly adopted and well supported by the development communities. This might be an absolute boon for researchers for effortless and hassle-free designing and deploying MCC-edge at small-scale or individual levels.

10

Conclusions and Future Vision

"Life is the art of drawing sufficient conclusions from insufficient premises." --- Samuel Butler

Global warming has become the most dreadful reality for the inhabitants of the Earth. This has resulted in erratic climate change, causing to increasing severe weather events from powerful storms to devastating floods and from deadly heat-waves to extreme snowfalls and notwithstanding to the rising sea levels due to the melting of the polar ice plates. Industrial developments are the major reasons for deteriorating environmental conditions. Environmental effect is the most common negative externality for any industry. The damage is done through various forms such as excessive energy consumption, carbon and greenhouse gas emission, heat generation, use of toxic materials in production, non-degradable waste generation, etc. To sustain the Earth's environment, we need to focus on sustainable developments. Sustainable computing is an important armament for that. In line with the data volume increase, the need for computing resource has increased tremendously. Along with the supercomputers, the traditional HPCs, recently cloud computing has been popularized as a cheap and on-demand computing resource. But both of them have problems in terms of environmental externalities. Grid computing intends to use the existing resources (mainly desktops) for catering to the need for HPC. But desktops are losing popularity; in fact, same for laptops.

SMDs are gaining huge acceptance, and with the computing power they offer thanks to the power-packed hardware, they can be considered our new computer. Actually, they already have become our new computer for daily computing chores. The philosophy of combining computation power of numerous distributed SMDs to escalate the computation power leads to mobile crowd computing (MCC). The cumulative computing power achieved by such grids of SMDs can well be compared to other HPC systems and can tail off the dependency on the data centres and low-end supercomputers as well. MCC is established on the policy of sharing

the CPU cycles of multiple SMDs in a distributed manner, voluntary or involuntary basis, to resolve high computation tasks. The easy availability of SMDs and the distributed nature makes MCC flexible highly scalable, which could be set up in an ad-hoc or on-demand basis.

With setting the ground and motivation in Chapter 1, in this thesis, we proposed to exercise MCC to attain sustainable computing. The use of SMDs would lessen the externalities because the manufacturing process and device operation would be minimized significantly. Due to smaller in size, less raw materials would be used in production which means less exposure to harmful elements and less pollution due to e-waste. Furthermore, if organizations can avoid buying computing resources can save a significant IT investment and operational cost.

MCC is not a totally new concept. It is standing on the strong basis of many existing and established computing paradigms such as distributed systems, parallel computing, distributed computing, grid computing, volunteer computing, opportunistic computing, crowdsourced computing, to name a few. In Chapter 2, we briefly discussed these technologies. In this chapter, we investigated the theoretical and empirical research works which are closely related to the problems addressed in this thesis. We confirmed the research scope for each case that led us to nurture the problem to find an acceptable solution. We found that not much work has been done specifically on MCC. This motivated us to take a deep dive into the realm of MCC.

In Chapter 3, we established the concept of MCC and presented its various aspects in detail. Since its inception, mobile devices have evolved a long way. The latest SMD CPUs, accompanied by their apposite counterpart GPUs, enable us to accomplish all the tasks for which we currently need a computer using an SMD. Today's SMDs are powerful enough to compete with the supercomputers of a few years back. The revolution of SMDs, regarding capability and usage, has changed our perception of what computing is today. Also, the popularity and adoption of SMDs have vanquished all estimations. Presently, nearly 6.648 billion people (83.37% of the world's population) use SMDs, not counting tablets. And like any other electronic consumer, SMDs also remain unused most of the day. This yields a vast pool

of unutilised resources, considering the global SMD user base. Along with, several other reinforcing factors such as increasing Wi-Fi zones, low-cost and highspeed mobile data, and highspeed and energy-efficient short-range communication technologies gave us the confidence to consider MCC as a feasible HPC option.

Besides presenting an unambiguous definition of MCC we made clear the confusions between MCC and other alike computing systems. We also saw that MCC can have two fundamental architecture types – centralised and P2P and based on these, can be of three types – global, local, and ad-hoc MCC. For proper and expected functioning of MCC it should be designed suitably. For this several criteria such as abstraction, generalisation, adaptability, reliability, fault-tolerance, QoS, scalability, elasticity, user friendliness, non-intrusiveness, energy efficiency, and mitigating SLA, liabilities and legalities should be met. Also, several other factors such as determining appropriate architecture as per MCC applications and available infrastructure, crowdworker management (including discovery, profiling, selection, ensuring availability, and monitoring), task farming, task scheduling, utilising resource scavenging and opportunistic computing, workflow management, and result verification and aggregation should be considered carefully. We discussed the advantages that MCC brings on that includes ubiquitous and sustainable computing.

No system does come without issues and challenges. MCC is also no exception. Some crucial issues like battery constraint and quick heating of the SMDs, and unreliable wireless network connectivity and bandwidth might hamper the employment of MCC. For successful application of MCC, several other challenges such as ensuring security, privacy and trust, motivating people in taking part in MCC, and, for that, framing sustainable economic models need to be addressed. We mentioned several potential applications of MCC ranging from meeting conventional HPC demands, businesses, military to smart cities, and many more.

MCC indeed has immense computational potential. But the success of MCC entirely depends on the availability of usable SMDs. It will be an outright failure if a sufficient number of SMDs are not available in crucial times of execution. That is why it is not advised to rely completely on MCC to perform critical jobs unless

there is an absolute guarantee of availability. In that case, we suggest to have an alternate computing system (e.g., cloud service) as a backup option. Nonetheless, barring this kind of unfortunate conditions, MCC is a brilliant scheme to have inexpensive HPC.

Regarding empirical study, in this thesis, we mainly addressed different aspects of local MCC where the public-owned SMDs are connected to the organisational MCC through a local wireless network (e.g., WLAN), and most of the SMDs are recurrently and regularly connected. In Chapter 4, we designed an operational framework for profiling various information of the connected SMDs which are required for assessing the compatibility and capability of the SMD for being assigned an MCC task. We also conceived a customised benchmarking scheme to assess the computing competency of the SMDs.

After profiling the resource capacity of the SMDs the next challenging job was to select the right SMDs among the available ones. This selection process is not trivial due to the fact that the resource parameters which define an SMD's suitability as a computing resource provider are vastly heterogeneous and dynamically variable. This problem can be perceived as an MCDM problem where the selection would be made by optimally balancing all the parameters. There are several MCDM methods in the literature and in practice. To find out the most suitable MCDM method for resource selection in MCC, in Chapter 5, we performed a comparative analysis of different MCDM methods. Here, we considered five distinct MCDM methods (EDAS, ARAS, MABAC, MARCOS, and COPRAS) that follow different solution approaches. Among them, we found COPRAS being the most favourable MCDM method for the given problem.

In Chapter 6, we adopted heuristic and metaheuristic optimisation techniques for task scheduling in MCC. We proposed a) a resource-aware heuristic scheduling algorithm considering makespan, resource utilisation and load balance and b) a PSO-based scheduling algorithm considering energy consumption and load balance. Both of these scheduling criteria are crucial for a better performance of MCC and its successful implementation by maintaining the required level of user satisfaction. The first one ensures minimising overall execution time while utilising all

the available SMDs evenly without overburdening only a few particular SMDs. The second on ensures minimising overall energy consumption while maintaining a fair load balance among the SMDs. Both of our proposed algorithms performed better in different simulation set ups (different combinations of varied task size and number of SMDs) compared to other existing heuristic and metaheuristic algorithms.

In Chapter 7, we designed a model for predicting mobile devices' availability for a local MCC where people join the MCC regularly and stay connected for varying periods. Before submitting the job to a crowdworker, the probability of the considered crowdworker being available until the job execution is finished is evaluated. If the job execution time is greater than the predicted availability duration, an alternative crowdworker is considered. This would improve the QoS of MCC by minimizing the job reassignment. We presented a novel dynamic feature extraction method where the features are unknown. Based on the extracted and selected features, we experimented with a convolutional prediction model applied on LSTM and GRU. Utilizing the real user mobility traces for a Wi-Fi AP deployed in a research lab, a convolutional LSTM (CLSTM) and a convolutional GRU (CGRU) based prediction methods were applied to predict the out-time of the user for each in-time. The prediction model was implemented on two datasets of different volumes. Our experiment showed that introducing the proposed convolutional feature extractor to the LSTM and GRU prediction models exhibit significant improvement in the prediction accuracy compared to the traditional LSTM and GRU-based prediction models. The proposed model competes favourably against another models, viz. ARIMA, popularly used in time-series prediction. The proposed CLSTM and CGRU models give satisfactory performance accuracy not only when the dataset is large enough but also for the small-sized dataset. It is observed that with an increase in dataset size, the performance of the model improves significantly. Also, with the increase in the dataset, the error estimation of the model gets better. However, compared to the CGRU, CLSTM model exhibited better performance. Thus, we can conclude that the proposed CGRU model can be a feasible resource availability prediction model in MCC both with small- and large-scale

user mobility data.

In Chapter 8, we presented the concept of a P2P MCC. Though P2P MCC is a brilliant opportunity for sharing computing services in a flexible and ad-hoc manner, the failure in addressing the mobility factor of the participating mobile devices will lead to an unsuccessful implementation. We presented solutions to the problem of finding a stable pairs of service provider and consumer for both single and multi-cluster PMCCs. In a single-cluster PMCC, to avoid frequent job loss and job handover, it is desired that the SMDs which are relatively static to the service requesting SMD for a longer period should be chosen as the service providers. To find the mobility patterns of a group of mobile users using the proposed algorithm, we conducted two studies on a dataset of real traces of mobile users connected to different Wi-Fi APs. The first study applied the algorithm to calculate relative stability for 20 sample users from the trace, choosing time spans of 2-hour duration. These values gave the instantaneous behaviour of the nodes. In the second study, the average relative stability of a user was calculated taking its neighbourhood information during the entire trace period of 78 days. A logistic regression analysis of the average stability of all users from the trace was performed.

In a multi-cluster PMCC, the mismatch in the contact time between the service consumer, provider, and the carrier (that acts as a mediator to exchange the service between the provider and the consumer) prohibits in exchanging the service. Hence, it is very important to know when they would be in contact with each other so that the service could be exchanged. In a mobile setup, it is not trivial to know. But if the mobility patterns of the devices are tracked, the location of the devices and the duration for which they remain in a network can be predicted. We applied a mobility prediction algorithm on the same dataset to extract different mobility-related information of the users. These information were utilised in selecting the service carrier and the provider so that the availability (consumer waits for the service indefinitely) and liveness (the carrier returns with the service but cannot find the consumer) problems could be averted.

To validate the utility of MCC, in Chapter 9, we presented a proof-of-concept. We demonstrated the feasibility of MCC as a case of crowdsourced edge computing.

We proposed to use MCC as a viable and sustainable alternative to proprietary edge computing. For this, we considered a smart HVAC system of a building as a use case scenario where the sensor data are processed in real time using MCC. We considered dew temperature and heat index to auto adjust the room temperature at an ideal level. Data generated by the temperature and humidity sensors and occupancy sensors were collected by a local coordinator which then curated them into parallelly executable subtasks and dispatched to the SMDs. As per the pre-s*et al*gorithms and depending on the results returned by the SMDs, the coordinator sent the required adjustment values to the AC controllers.

To make it straightforwardly reproduceable, we illustrated the exhaustive details of designing, developing, and implementing of each module of the prototype with sufficient specifications and descriptions. All the modules were built using open-source and commodity tools and libraries that are being popularly accepted and well supported by the development communities.

To establish the advantage of MCC as edge computing, we compared it with the commercial edge computing solutions and also with cloud services in terms of cost, energy consumption, and latency. MCC had been emerged as the most sustainable edge computing solution thanks to its crucial advantages such as energy efficient, environment friendly, and economical.

We considered the HVAC system just to demonstrate the feasibility of using MCC as an edge computing solution. However, this can be extended for the other aspects of a smart building, such as security, fire system, etc. In our view, the MCC-edge system should be of general purpose and its utilisation would not be limited to a building; rather, it can be utilised to set up a semi-infrastructure based or infrastructure-less ad-hoc and dynamic edge computing system ubiquitously, supporting diverse applications. Actually, that would upshot the realization of the true value of MCC. Therefore, we encourage the research community to explore the application of MCC in the diverse domains while addressing the challenging issues.

Though building prototypes and proof-of-concept is a necessary measure to evaluate and validate ideas, it is not sufficient for the full-fledged implementation and,

most importantly, to excite the commercial and general userbase. Due to practical limitations, our study has been restricted to a few aspects of MCC only. However, there are several important aspects and crucial issues yet to be addressed, which we wish to investigate in the future.

To make the designed prototype simple, we did not include the aspects presented in chapters 4 to 8. Here our focus was to establish a working MCC system that can be used to process sensor data distributedly on a set of SMDs. However, for real-world adoption of the projected proof-of-concept, all the components should be integrated into a fully operational system so that people are convinced of the system processes and their effectiveness, which is essential for any sustainable proposition. We believe the demonstrated proof-of-concept is a modest but definitive step towards sustainable computing. We advocate making the proposed system evolve and much worthful by gaining experience through its live deployments, which would lead to filling the obvious gaps with innovative solutions from the allied research community.

We envisage MCC to be flourished as a dependable and widely-used computing paradigm. We expect that the computation and storage capacity of the future SMDs will most likely continue to increase according to Moore's law. However, it will be interesting to see to what extent the wireless data transmission speed and battery capacity will match that. Because the battery still remains the major hurdle and will be for at least a couple of years with the present battery technology. Also, the wireless technology has failed to reach the users' satisfactory level till now with a continuous increase in data traffic due to a massive upsurge in sophisticated data-intensive applications. Nevertheless, the continuous growth in the number of SMD users gives us confidence in alleviating the availability issue. There will be a sufficient number of devices in the vicinity that can be harnessed. The SMD user base will continue to grow as people of all levels would be forced to have SMDs with the introduction of more digitised services (both government and private), which might be essential to lead daily life for each individual. Furthermore, in the coming years, more people will shift towards the cities, and as the density of cities increases, the availability of the required number of SMDs at any place will be almost

certain. This would be advantageous to integrate MCC with smart city applications such as traffic monitoring and management, public security and policing, etc. On account of the flexibility and ad-hoc nature of MCC, it will play a crucial role in satisfying the need for ubiquitous computing. Especially in the M2M, IoT, and IoE scenario, MCC can play a vital role in providing an impromptu HPC milieu for real-time data processing and analysis. We believe proper utilisation of MCC can make it an integral part of the IoE ecosystem and industry 5.0. Well-thought designing and deployment would allow us to leverage the services of MCC for varieties of trending and futuristic technologies such as UAVs, autonomous vehicles, context-aware servicing, etc.

We further suppose that MCC will realise its success when its services are not only limited to the organisation directed to solve certain specific problems but also equally accessible to the common people in a generalised way. The SMD users will not only be service providers (crowdworkers) but also can avail of computing services when needed. We are optimistic about realising MCC-as-a-Service (MaaS) is very much like IaaS in cloud computing. The MaaS recipients should only submit their jobs with a wrapper without worrying much about the underlying complexities. The MCC system would handle the required actions such as task farming, task distribution, result aggregation and verification, and returning the final results to the service recipient. Unlike cloud computing, the cost of availing of MaaS will be either zero or very nominal. A barter system can be followed, i.e., a crowdworker can avail of free services in return for a certain level of contribution to the MCC from its end. This will build a largescale sustainable MCC ecosystem.

At long last, stating our conviction, we are expectant that the research work reported in this thesis is able to add a small stone towards building a sustainable computing paradigm. Our effort will be rewarding if it is successful in drawing attentions of the prospective researchers and take along the mission. It will be great if this study encourages the organisations and other stakeholders to give a serious thought on the negative impacts of computing devices and consider adopting MCC as their primary computing infrastructure.

References

[1] Climatenexus, "Top climate events of 2017," 2018. [Online]. Available: https://climatenexus.org/2017-top-climate-events/. [Accessed 19 August 2018].

[2] P. K. D. Pramanik, S. Pal and P. Choudhury, "Smartphone crowd computing: a rational solution towards minimising the environmental externalities of the growing computing demands," in *Emerging Trends in Disruptive Technology Management*, R. Das, M. Banerjee and S. De, Eds., New York, Chapman and Hall/CRC, 2019, pp. 45-80.

[3] R. Monroe, "Comment on recent record-breaking CO2 concentrations," 20 April 2016. [Online]. Available: https://scripps.ucsd.edu/programs/keelingcurve/2016/04/20/comment-on-recent-record-breaking-co2-concentrations/#more-1406. [Accessed 26 August 2018].

[4] WWF-Australia, "Causes of global warming," WWF-Australia, 2018. [Online]. Available: https://www.wwf.org.au/what-we-do/climate/causes-of-global-warming. [Accessed 28 August 2018].

[5] N. Jones, "How to stop data centres from gobbling up the world's electricity," 12 September 2018. [Online]. Available: https://www.nature.com/articles/d41586-018-06610-y. [Accessed 24 February 2019].

[6] E. Gelenbe and Y. Caseau, "The impact of information technology on energy consumption and carbon emissions," *Ubiquity*, vol. 2015, no. June, pp. 1-15, 2015.

[7] United Nations University Newsletter, "Study tallies environmental cost of computer boom," 2004. [Online]. Available: http://archive.unu.edu/update/archive/issue31_5.htm. [Accessed 18 February 2019].

[8] P. H. Gleick and M. J. Cohen, "Water content of things: the biennial report on freshwater resources," in *The World's Water 2008–2009*, Island Press, 2009, pp. 335-338.

[9] K. D. Decker, "The monster footprint of digital technology," Low-tech Magazine, 16 June 2009. [Online]. Available: https://www.lowtechmagazine.com/2009/06/embodied-energy-of-digital-technology.html. [Accessed 13 May 2022].

[10] "To manufacture the computer in which you read this, 1,500 liters of water were consumed," EL PAÍS, 7 March 2007. [Online]. Available: https://elpais.com/tecnologia/2007/03/07/actualidad/1173259681_850215.html. [Accessed 13 May 2022].

[11] A. Wang, "TrendForce reports notebook shipments totaled 164.4 million units in 2015 with Apple gaining greater market share annually," TrendForce, 16 February 2016. [Online]. Available: https://www.trendforce.com/presscenter/news/20160216-9238.html. [Accessed 13 May 2022].

[12] P. K. D. Pramanik, B. Mukherjee, S. Pal, T. Pal and S. P. Singh, "Green smart building: requisites, architecture, challenges, and use cases," in *Green Building Management and Smart Automation*, A. Solanki and A. Nayyar, Eds., IGI Global, 2019, pp. 1-50.

[13] R. Monroe, "Potent greenhouse gas more prevalent," 23 October 2008. [Online]. Available: https://ucsdnews.ucsd.edu/archive/newsrel/science/10-08GreenhouseGas.asp. [Accessed 31 July 2018].

[14] Research and Markets, "Global e-waste recycling & reuse rervices market size, market share, application analysis, regional outlook, growth trends, key players, competitive strategies and forecasts, 2018 to 2026," January 2019. [Online]. Available: https://www.researchandmarkets.com/research/lwvm28/ewaste_recycling?w=4. [Accessed 18 February 2019].

[15] S. Honda, D. S. Khetriwal and R. Kuehr, "Regional e-waste monitor: East and Southeast Asia," United Nations University, Bonn, Germany, 2016.

[16] R. Leblanc, "E-Waste Recycling Facts and Figures," 31 December 2018. [Online]. Available: https://www.thebalancesmb.com/e-waste-recycling-facts-and-figures-2878189. [Accessed 21 February 2019].

[17] C. Gillespie, "Negative effects of pollution," 11 June 2018. [Online]. Available: https://sciencing.com/negative-effects-pollution-5268664.html. [Accessed 26 August 2018].

[18] S. Prakash, R. Liu, K. Schischke and L. Stobbe, "Timely replacement of a notebook under consideration of environmental aspects," Federal Environment Agency (Umweltbundesamt), Germany, 2012.

[19] "The environment and electronic devices," Alboan, 2022. [Online]. Available: https://www.tecnologialibredeconflicto.org/en/environment/. [Accessed 13 May 2022].

[20] E. Gillespie, "'Spaceship earth': design for the challenge of sustainability in the 21st century," 2004. [Online]. Available: https://www.sda-uk.org/Ed%20Gillespie.ppt. [Accessed 14 May 2022].

[21] ProKerala, “Responsibility of the government on e-waste management,” 21 July 2012. [Online]. Available: https://www.prokerala.com/going-green/e-waste-management-and-role-of-government-and-indistries.htm. [Accessed 26 February 2019].

[22] Silicon Mechanics, “Sustainable computing at silicon mechanics,” 2018. [Online]. Available: https://www.siliconmechanics.com/i16642/sustainable-computing.php. [Accessed 21 February 2019].

[23] M. Maksimovic, “Greening the future: green Internet of Things (G-IoT) as a key technological enabler of sustainable development,” in *Internet of Things and Big Data Analytics Toward Next-Generation Intelligence*, Springer, 2017, pp. 283-313.

[24] W. Chedid and C. Yu, “Survey on power management techniques for energy efficient computer systems,” Cleveland State University, Cleveland, 2002.

[25] Y. Liu and H. Zhu, “A survey of the research on power management techniques for high performance systems,” *Software-Practice & Experience*, vol. 40, no. 11, pp. 943-964, 2010.

[26] S. Mittal, “Power management techniques for data centers: a survey,” Oak Ridge National Laboratory, USA, 2014.

[27] K. Goyal, “Power management in mobile devices by various protocols,” *International Journal of Computer Science and Communication*, vol. 2, no. 2, pp. 505-508, 2011.

[28] C. Shah, S. Chaudhary and P. Agrawal, “Performance analysis of efficient power management controls in Android device,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 6, no. 3, pp. 83-89, 2017.

[29] A. Abdelmotalib and Z. Wu, “Power management techniques in smartphones operating systems,” *International Journal of Computer Science Issues*, vol. 9, no. 3, p. 157–160, 2012.

[30] P. K. D. Pramanik, S. Pal, A. Brahmachari and P. Choudhury, “Processing IoT data: from cloud to fog. It’s time to be down-to-earth,” in *Applications of Security, Mobile, Analytic and Cloud (SMAC) Technologies for Effective Information Processing and Management*, P. Karthikeyan and M. Thangavel, Eds., IGI Global, 2018, pp. 124-148.

[31] E. Knorr, “What serverless computing really means,” 11 July 2016. [Online]. Available: https://www.infoworld.com/article/3093508/what-serverless-computing-really-means.html. [Accessed 28 March 2019].

[32] Cloudflare, “How are serverless computing and platform-as-a-service different? | PaaS vs. serverless,” 2019. [Online]. Available: https://www.cloudflare.com/learning/serverless/glossary/serverless-vs-paas/. [Accessed 28 March 2019].

[33] J. G. Koomey, “Growth in data center electricity use 2005 to 2010,” 1 August 2011. [Online]. Available: https://www.missioncriticalmagazine.com/ext/resources/MC/Home/Files/PDFs/Koomey_Data_Center.pdf. [Accessed 1 September 2018].

[34] P. K. D. Pramanik, S. Pal, G. Pareek, S. Dutta and P. Choudhury, “Crowd computing: the computing revolution,” in *Crowdsourcing and Knowledge Management in Contemporary Business Environments*, R. Lenart-Gansiniec, Ed., IGI Global, 2018, pp. 166-198.

[35] M. Y. Arslan, I. Singh, S. Singh, H. V. Madhyastha, K. Sundaresan and S. V. Krishnamurthy, “CWC: a distributed computing infrastructure using smartphones,” *IEEE Transactions on Mobile Computing*, vol. 14, no. 8, pp. 1587-1600, 2015.

[36] P. K. D. Pramanik, N. Sinhababu, K.-S. Kwak and P. Choudhury, “Deep learning based resource availability prediction for local mobile crowd computing,” *IEEE Access*, vol. 9, pp. 116647-116671, 2021.

[37] P. Prakash, “Environmental impact of internet searches and data centers,” 10 July 2017. [Online]. Available: https://www.linkedin.com/pulse/environmental-impact-internet-searches-data-centers-pranav-prakash/. [Accessed 18 August 2018].

[38] T. Bawden, “Global warming: data centres to consume three times as much energy in next decade, experts warn,” 23 January 2016. [Online]. Available: https://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html. [Accessed 23 February 2019].

[39] A. S. Andrae, “Total consumer power consumption forecast,” in *Nordic Digital Business Summit*, Helsinki, Finland, 2017.

[40] D. M. Shila, W. Shen, Y. Cheng, X. Tian and X. S. Shen, “AMCloud: toward a secure autonomic mobile ad hoc cloud computing system,” *IEEE Wireless Communications*, vol. 24, no. 2, pp. 74-81, 2017.

[41] Energuide, “How much power does a computer use? And how much CO2 does that represent?,” Sibelga, 2019. [Online]. Available: https://www.energuide.be/en/questions-answers/how-much-power-does-a-computer-use-and-how-much-co2-does-that-represent/54/. [Accessed 22 February 2019].

[42] IEA, “CO2 emissions from fuel combustion highlights,” 2013.

[43] Fujitsu, “Life cycle assessment and product carbon footprint,” Fujitsu, 2010.

[44] Arizona State University, “Factory is where our computers eat up most energy,” 14 April 2011. [Online]. Available: https://phys.org/news/2011-04-factory-energy.html. [Accessed 22 February 2019].

[45] Lovefone Blog, “How much CO2 does it take to make a smartphone?,” 28 March 2018. [Online]. Available: https://www.lovefone.co.uk/blogs/news/how-much-co2-does-it-take-to-make-a-smartphone. [Accessed 22 February 2019].

[46] Restart, “The global footprint of mobiles,” 2018. [Online]. Available: https://therestartproject.org/the-global-footprint-of-mobiles/. [Accessed 24 February 2019].

[47] Research Triangle Institute, “Hazard sssessment of the electronic component manufacturing industry,” U.S. Department of Health and Human Services, Ohio, 1985.

[48] S. Needhidasan, M. Samuel and R. Chidambaram, “Electronic waste – an emerging threat to the environment of urban India,” *Journal of Environmental Health Science & Engineering,* vol. 12, no. 36, 2014.

[49] ERIdirect, “How long does it take electronic waste to decompose?,” 3 November 2015. [Online]. Available: https://eridirect.com/blog/2015/11/how-long-does-it-take-electronic-waste-to-decompose/. [Accessed 22 February 2019].

[50] C. Conger, “Can my computer poison me?,” 22 August 2009. [Online]. Available: https://computer.howstuffworks.com/computer-poison1.htm. [Accessed 22 February 2019].

[51] A. Zeenat, “A study and development of an efficient e-waste management system for minimizing the risks of environmental pollution,” Savitribai Phule Pune University, Pune, 2016.

[52] R. Mihindukulasuriya, “Your mobile phone is a major contributor to toxic e-waste in the country,” 28 October 2018. [Online]. Available: https://theprint.in/science/your-mobile-phone-is-a-major-contributor-to-toxic-e-waste-in-the-country/141430/. [Accessed 25 February 2019].

[53] S. Liu, “Analysis of electronic waste recycling in the United States and potential application in China,” Columbia University, New York City, 2014.

[54] MCMC, “Mobile e-waste: old phone, new life,” 2015. [Online]. Available: https://mobileewaste.mcmc.gov.my/en-my/about-mobile-e-waste. [Accessed 25 February 2019].

[55] P. K. D. Pramanik, P. Choudhury and A. Saha, “Economical supercomputing thru smartphone crowd computing: an assessment of opportunities, benefits, deterrents, and applications from India’s perspective,” in *4th International Conference on Advanced Computing and Communication Systems (ICACCS-2017)*, Coimbatore, India, 2017.

[56] S. Srivastava, “Global smartphone shipments reached record 1.55 billion units in CY 2017,” 2 February 2018. [Online]. Available: https://www.counterpointresearch.com/global-smartphone-shipments-reached-record-1-55-billion-units-cy-2017/. [Accessed 17 August 2018].

[57] A. Scarsella, “Worldwide smartphone forecast update, 2021–2025: December 2021,” IDC, December 2021. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=US48451621. [Accessed 17 March 2022].

[58] S. O'Dea, “Smartphone users worldwide 2016-2023,” 31 March 2021. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Accessed 9 April 2021].

[59] S. O'Dea, “Number of smartphone subscriptions worldwide from 2016 to 2027,” Statista, 23 February 2022. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Accessed 17 March 2022].

[60] Statista Research Department, “Smartphone users in India 2015-2025,” 17 February 2021. [Online]. Available: https://www.statista.com/statistics/467163/forecast-of-smartphone-users-in-india/#:~:text=Smartphone%20users%20in%20India%202015%2D2025&text=The%20number%20of%20smartphone%20users,3.8%20billion%20users%20in%202021.. [Accessed 9 April 2021].

[61] A. Al-Heeti, “Android is on over 2.5 billion active devices,” 7 May 2019. [Online]. Available: https://www.cnet.com/tech/mobile/android-is-on-over-2-5-billion-active-devices/. [Accessed 21 April 2022].

[62] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan and D. Estrin, “Diversity in smartphone usage,” in *MobiSys’10*, San Francisco, USA, 2010.

[63] D. T. Wagner, A. Rice and A. R. Beresford, “Device analyzer: understanding smartphone usage,” in *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, vol. 131, Springer International Publishing, 2014, pp. 195-208.

[64] K. Abualsaud, T. M. Elfouly, T. Khattab, E. Yaacoub, L. S. Ismail, M. H. Ahmed and M. Guizani, “A survey on mobile crowd-sensing and its applications in the IoT era,” *IEEE Access*, vol. 7, pp. 3855-3881, 2018.

[65] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich and P. Bouvry, “A survey on mobile crowdsensing systems: challenges, solutions, and opportunities,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2419-2465, 2019.

[66] A. I. Chittilappilly, L. Chen and S. Amer-Yahia, “A survey of general-purpose crowdsourcing techniques,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 9, pp. 2246-2266, 2016.

[67] W. Soliman and V. K. Tuunainen, “Understanding continued use of crowdsourcing systems: an interpretive study,” *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 10, no. 1, pp. 1-18, 2015.

[68] M. Hosseini, C. M. Angelopoulos, W. K. Chai and S. Kundig, “Crowdcloud: a crowdsourced system for cloud infrastructure,” *Cluster Computing*, vol. 22, p. 455–470, 2019.

[69] Y. Wang, X. Jia, Q. Jin and J. Ma, “Mobile crowdsourcing: framework, challenges, and solutions,” *Concurrency and Computation: Practice and Experience*, vol. 29, no. 3, p. e3789, 2017.

[70] W. Li, W.-j. Wu, H.-m. Wang, X.-q. Cheng, H.-j. Chen, Z.-h. Zhou and R. Ding, “Crowd intelligence in AI 2.0 era,” *Frontiers of Information Technology & Electronic Engineering*, vol. 18, pp. 15-43, 2017.

[71] S. C. Shah, “Recent advances in mobile grid and cloud computing,” *Intelligent Automation & Soft Computing*, 2017.

[72] E. Miluzzo, R. Cáceres and Y.-F. Chen, “Vision: mClouds – computing on clouds of mobile devices,” in *3rd ACM workshop on Mobile cloud computing and services (MCS’12)*, Low Wood Bay, UK, 2012.

[73] S. W. Loke, Crowd-powered mobile computing and smart things, Cham, Switzerland: Springer, 2017.

[74] M. Satyanarayanan, “The emergence of edge computing,” *Computer*, vol. 50, no. 1, pp. 30-39, 2017.

[75] E. E. Marinelli, “Hyrax: cloud computing on mobile devices using MapReduce,” Masters Thesis, Carnegie Mellon University, Pittsburgh, 2009.

[76] M. Hirsch, C. Mateos and A. Zunino, “Augmenting computing capabilities at the edge by jointly exploiting mobile devices: a survey,” *Future Generation Computer Systems*, vol. 88, no. November, pp. 644-662, 2018.

[77] J. Kozlowicz, “8 ways data center environmental impact goes beyond emissions,” 11 November 2015. [Online]. Available: https://www.greenhousedata.com/blog/data-center-environmental-impact-goes-beyond-emissions. [Accessed 18 August 2018].

[78] A. S. Tanenbaum and M. v. Steen, Distributed systems: principles and paradigms, 2nd ed., New Jersy, USA: Pearson, 2007.

[79] M. J. Quinn, Parallel computing: theory and practice, India: McGraw-Hill Education, 1994.

[80] M. Baker and R. Buyya, “Cluster computing at a glance,” in *High Performance Cluster Computing - Architectures and Systems*, New Jersey, USA, Prentice Hall PTR, 1999, pp. 3-47.

[81] M. Baker and R. Buyya, “Cluster computing: the commodity supercomputer,” *Journal of Software: Practice and Experience*, vol. 29, no. 6, pp. 551-576, 1999.

[82] T. M. Mengistu and D. Che, “Survey and taxonomy of volunteer computing,” *ACM Computing Surveys*, vol. 52, no. 3, pp. 1-35, 2020.

[83] M. N. Durrani and J. A. Shamsi, “Volunteer computing: requirements, challenges, and solutions,” *Journal of Network and Computer Applications*, vol. 39, pp. 369-380, 2014.

[84] E. J. Korpela, "SETI@home, BOINC, and volunteer distributed computing," *Annual Review of Earth and Planetary Sciences,* vol. 40, pp. 69-87, 2012.

[85] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins and Z. Xu, "Peer-to-peer computing," HP Laboratories Palo Alto, 2003.

[86] D. Barkai, "An introduction to peer-to-peer computing," *Intel Developer Update Magazine,* pp. 1-7, February 2000.

[87] D. Xu, Y. Li, X. Chen, J. Li, P. Hui, S. Chen and J. Crowcroft, "A survey of opportunistic offloading," *IEEE Communications Surveys & Tutorials,* vol. 20, no. 3, pp. 2198-2236, 2018.

[88] M. Conti and M. Kumar, "Opportunities in opportunistic computing," *Computer,* vol. 43, no. 1, pp. 42-50, January 2010.

[89] M. D. Kristensen, "Scavenger: transparent development of efficient cyber foraging applications," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Mannheim, Germany, 2010.

[90] M. R. H. Ahangar, M. R. E. Taba and A. Ghafouri, "On a novel grid computing-based distributed brute-force attack scheme (GCDBF) by exploiting botnets," *International Journal of Computer Network and Information Security,* vol. 6, pp. 21-29, 2017.

[91] J. W. Strickland, V. W. Freeh, X. Ma and S. S. Vazhkudai, "Governor: autonomic throttling for aggressive idle resource scavenging," in *2nd International Conference on Autonomic Computing (ICAC'05)*, Seattle, USA, 2005.

[92] E. Rosales, G. Sotelo, A. Vega, C. O. Díaz, C. E. Gómez and H. Castro, "Harvesting idle CPU resources for desktop grid computing while limiting the slowdown generated to end-users," *Cluster Computing,* vol. 18, no. 4, pp. 1331-1350, 2015.

[93] F. Büsching, S. Schildt and L. Wolf, "DroidCluster: towards smartphone cluster computing - the streets are paved with potential computer clusters," in *32nd International Conference on Distributed Computing Systems Workshops*, Macau, China, 2012.

[94] D. P. Anderson, "iSGTW Opinion - Volunteer computing: grid or not grid?," 4 July 2007. [Online]. Available: https://sciencenode.org/feature/isgtw-opinion-volunteer-computing-grid-or-not-grid.php. [Accessed 6 August 2022].

[95] G. Massari, M. Zanella and W. Fornaciari, "Towards distributed mobile computing," in *Mobile System Technologies Workshop (MST)*, Milan, Italy, 2016.

[96] D. Datla, X. Chen, T. Tsou, S. Raghunandan, S. M. Hasan, J. Reed, B. Fette, C. B. Dietrich, J.-H. Kim and T. Bose, "Wireless distributed computing: a survey of research challenges," *IEEE Communications Magazine,* vol. 50, no. 1, 2012.

[97] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikainen and V. H. Tuulos, "Misco: a MapReduce framework for mobile systems," in *3rd International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '10)*, Samos Greece, 2010.

[98] T. Kakantousis, I. Boutsis, V. Kalogeraki, D. Gunopulos, G. Gasparis and A. Dou, "Misco: a system for data analysis applications on networks of smartphones using MapReduce," in *IEEE 13th International Conference on Mobile Data Management (MDM)*, Bengaluru, India, July 2012.

[99] S. Lee, K. Grover and A. Lim, "Enabling actionable analytics for mobile devices: performance issues of distributed analytics on Hadoop mobile clusters," *Journal of Cloud Computing: Advances, Systems and Applications,* vol. 2 (Article number: 15), 2013.

[100] E. Arnold, *AVRF: a framework to enable distributed computing using volunteered mobile resources,* vol. Paper 127, University of Puget Sound, 2011.

[101] Z. Dong, L. Kong, P. Cheng, L. He, Y. Gu, L. Fang, T. Zhu and C. Liu, "REPC: reliable and efficient participatory computing for mobile devices," in *Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Singapore, 2014.

[102] C. Dumont, F. Mourlin and L. Nel, "A mobile distributed system for remote resource access," in *14th International Conference on Advances in Mobile Computing and Multi Media (MoMM '16)*, Singapore, 2016.

[103] H. m. Salem, "Distributed computing system on a smartphones-based Network," in *Software Technology: Methods and Tools (TOOLS 2019). Lecture Notes in Computer Science*, vol. 11771, M. Mazzara, J. M. Bruel, B. Meyer and A. Petrenko, Eds., Springer, Cham, 2019, pp. 313-325.

[104] P. Sanches, J. A. Silva, A. Teófilo and H. Paulino, "Data-centric distributed computing on networks of mobile devices," in *Parallel Processing (Euro-Par 2020). Lecture Notes in Computer Science*, vol. 12247, M. Malawski and K. Rzadca, Eds., Springer, Cham, 2020, p. 296–311.

[105] D. E. Attia, A. M. ElKorany and A. S. Moussa, "High performance computing over parallel mobile systems," *International Journal of Advanced Computer Science and Applications,* vol. 7, no. 9, pp. 99-103, 2016.

[106] M. Conti, S. Giordano, M. May and A. Passarella, "From opportunistic networks to opportunistic computing," *IEEE Communications Magazine,* vol. 48, no. 9, September 2010.

[107] D. G. Murray, E. Yoneki, J. Crowcroft and S. Hand, "The case for crowd computing," in *2nd ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds (MobiHeld '10)*, New Delhi, India, 2010.

[108] C. Shi, V. Lakafosis, M. H. Ammar and E. W. Zegura, "Serendipity: enabling remote computing among intermittently connected mobile devices," in *13th ACM international symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '12)*, South Carolina, USA, 2012.

[109] A. Mtibaa, K. A. Harras, K. Habak, M. Ammar and E. W. Zegura, "Towards mobile opportunistic computing," in *IEEE 8th International Conference on Cloud Computing*, New York, USA, 2015.

[110] C. Tapparello, C. Funai, S. Hijazi, A. Aquino, B. Karaoglu, H. Ba, J. Shi and W. Heinzelman, "Volunteer computing on mobile devices: state of the art and future research directions," in *Enabling Real-Time Mobile Cloud Computing through Emerging Technologies*, IGI Global, 2015, pp. 153-181.

[111] E. Lavoie, L. Hendren, F. Desprez and M. P. Correia, "Pando: personal volunteer computing in browsers," in *20th International Middleware Conference (Middleware '19)*, California, United States, 2019.

[112] P. Jenviriyakul, G. Chalumporn, T. Achalakul, F. Costa and K. Akkarajitsakul, "ALICE Connex: a volunteer computing platform for the time-of-flight calibration of the ALICE experiment. An opportunistic use of CPU cycles on Android devices," *Future Generation Computer Systems,* vol. 94, pp. 510-523, 2019.

[113] D. P. Anderson, "BOINC: a system for public-resource computing and storage," in *Fifth IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, 2004.

[114] M. Y. Arslan, I. Singh, S. Singh, H. V. Madhyastha, K. Sundaresan and S. V. Krishnamurthy, "Computing while charging: Building a distributed computing infrastructure using smartphones," in *8th international conference on Emerging networking experiments and technologies (CoNEXT '12)*, France, 2012.

[115] S. Schildt, F. Busching, E. Jorns and L. Wolf, "CANDIS: heterogeneous mobile cloud framework and energy cost-aware scheduling," in *IEEE GreenCom iThings/CPSCom*, Beijing, 2013.

[116] T. Phan, L. Huang and C. Dulan, "Integrating mobile wireless devices into the computational grid," in *8th Annual International Conference on Mobile Computing and Networking (MobiCom '02)*, Atlanta, USA, 2002.

[117] T. Phan, L. Huang and C. Dulan, "Challenge: integrating mobile wireless devices into the computational grid," in *8th Annual International Conference on Mobile Computing and Networking (MobiCom '02)*, New York, USA, 2002.

[118] F. Gonzalez-Castano, J. Vales-Alonso and M. Livny, "Condor grid computing from mobile handheld devices," *Mobile Computing and Communications Review,* vol. 6, no. 2, pp. 117-126, April 2002.

[119] B. P. Clarke and M. Humphrey, "Beyond the 'device as portal': meeting the requirements of wireless and mobile devices in the legion grid computing system," in *16th International Parallel and Distributed Processing Symposium (IPDPS 2002)*, Fort Lauderdale, FL, USA, 2002.

[120] D. C. Chu and M. Humphrey, "Mobile OGSI.NET: grid computing on mobile devices," in *5th IEEE/ACM International Workshop on Grid Computing (associated with Supercomputing 2005)*, Pittsburgh, PA, 2004.

[121] U. Farooq and W. Khalil, "A generic mobility model for resource prediction in mobile grids," in *International Symposium on Collaborative Technologies and Systems*, Las Vegas, USA, 2006.

[122] S. Kurkovsky and Bhagyavati, "Wireless grid enables ubiquitous computing," in *16th International Conference on Parallel and Distributed Computing Systems (PDCS-2003)*, Reno, NV, 2003.

[123] S. Kurkovsky, Bhagyavati and A. Ray, "A collaborative problem-solving framework for mobile devices," in *42nd annual Southeast regional conference (ACM-SE 42)*, New York, USA, 2004.

[124] K. Katsaros and G. C. Polyzos, "Optimizing operation of a hierarchical campus-wide mobile grid for intermittent wireless connectivity," in *15th IEEE Workshop on Local & Metropolitan Area Networks*, Princeton, USA, 2007.

[125] M. Black and W. Edgar, "Exploring mobile devices as grid resources: using an x86 virtual machine to run BOINC on an iPhone," in *10th IEEE/ACM International Conference on Grid Computing*, Melbourne, Australia, 2009.

[126] H. Viswanathan, E. K. Lee, I. Rodero and D. Pompili, "Uncertainty-aware autonomic resource provisioning for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 26, no. 8, pp. 2363-2372, 2015.

[127] R. Sriraman K., "Grid computing on mobile devices: a point of view," Altimetrik Insights, April 2014.

[128] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, M. Imran and S. Guizani, "Mobile ad hoc cloud: a survey," *Wireless Communications and Mobile Computing,* vol. 16, no. 16, pp. 2572-2589, 2016.

[129] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (MCS '10)*, San Francisco, California, 2010.

[130] A. Khalifa, R. Hassan and M. Eltoweissy, "Towards ubiquitous computing clouds," in *3rd International Conference on Future Computational Technologies and Applications*, Rome, Italy, 2011.

[131] A. Khalifa and M. Eltoweissy, "A global resource positioning system for ubiquitous clouds," in *International Conference on Innovations in Information Technology (IIT)*, Abu Dhabi, UAE, 2012.

[132] A. Khalifa and M. Eltoweissy, "Collaborative autonomic resource management system for mobile cloud computing," in *The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization*, Valencia, Spain, 2013.

[133] A. Khalifa and M. Eltoweissy, "MobiCloud: A reliable collaborative mobilecloud management system," in *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Austin, USA, 2013.

[134] A. Khalifa, M. Azab and M. Eltoweissy, "Resilient hybrid mobile ad-hoc cloud over collaborating heterogeneous nodes," in *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Miami, USA, 2014.

[135] T. Nishio, R. Shinkuma, T. Takahashi and N. B. Mandayam, "Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud," in *First international workshop on Mobile cloud computing & networking*, Bangalore, India, 2013.

[136] C. Funai, C. Tapparello, H. Ba, B. Karaoglu and W. Heinzelman, "Extending volunteer computing through mobile ad hoc networking," in *IEEE Global Communications Conference*, Austin, USA, 2014.

[137] D. Remédios, A. Teófilo, H. Paulino and J. Lourenço, "Mobile device-to-device distributed computing using data sets," in *12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MOBIQUITOUS)*, Coimbra, Portugal, 2015.

[138] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar and M. Imran, "Heterogeneity-aware task allocation in mobile ad hoc cloud," *IEEE Access,* vol. 5, p. 1779–1795, 2017.

[139] V. Balasubramanian and A. Karmouch, "An infrastructure as a service for mobile ad-hoc cloud," in *IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, USA, 2017.

[140] K. Habak, M. Ammar, K. A. Harras and E. Zegura, "FemtoClouds: leveraging mobile devices to provide cloud service at the edge," in *8th International Conference on Cloud Computing*, New York, USA, 2015.

[141] S. W. Loke, K. Napier, A. Alali, N. Fernando and W. & Rahayu, "Mobile computations with surrounding devices: proximity sensing and multiLayered work stealing," *ACM Transactions on Embedded Computing Systems,* vol. 14, no. 2, 2015.

[142] N. Fernando, S. W. Loke and W. Rahayu, "Honeybee: a programming framework for mobile crowd computing," in *Mobile and Ubiquitous Systems: Computing, Networking, and Services (MobiQuitous 2012). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 120, K. Zheng, M. Li and H. Jiang, Eds., Berlin, Heidelberg, Springer, 2013, pp. 224-236.

[143] N. Fernando, S. W. Loke and W. Rahayu, "Computing with nearby mobile devices: a work sharing algorithm for mobile edge-clouds," *IEEE Transactions on Cloud Computing,* vol. 7, no. 2, pp. 329-343, 2019.

[144] S. W. Loke, "Crowd+Cloud Machines," in *Crowd-powered mobile computing and smart things*, Cham, Springer, 2017, pp. 11-25.

[145] M. P. Kumar, R. R. Bhat, S. R. Alavandar and V. S. Ananthanarayana, "Distributed public computing and storage using mobile devices," in *IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, Mangalore, India, 2018.

[146] S. Kündig, C. M. Angelopoulos, S. R. Kuppannagari, J. Rolim and V. K. Prasanna, "Crowdsourced edge: a novel networking paradigm for the collaborative community," in *16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Marina del Rey, USA, 2020.

[147] University of California, "BOINC on Android," 2014. [Online]. Available: https:// boinc.berkeley.edu/trac/wiki/AndroidBoinc. [Accessed 18 August 2016].

[148] matszpk, "NativeBOINC," 13 September 2012. [Online]. Available: http://nativeboinc.org/site/ uncat/start. [Accessed 18 August 2016].

[149] J. Duda and W. Dłubacz, "Distributed evolutionary computing system capable to use mobile devices," in *Conference of Informatics and Management Sciences*, 2013.

[150] "DreamLab: app creates 'smartphone supercomputer' to help find cure for cancer," 9 November 2015. [Online]. Available: http://www.abc.net.au/news/2015-11-09/smartphone-app-dreamlab-helps-find-cure-for-cancer/6923452. [Accessed 11 August 2022].

[151] E. Lavoie and L. Hendren, "Personal volunteer computing," in *16th ACM International Conference on Computing Frontiers (CF '19)*, Alghero, Italy, 2019.

[152] A. Zavodovski, L. Corneo, A. Johnsson, N. Mohan, S. Bayhan, P. Zhou, W. Wong and J. Kangasharju, "Decentralizing computation with edge computing: potential and challenges," in *Interdisciplinary Workshop on (de) Centralization in the Internet (IWCI'21)*, Germany, 2021.

[153] A. Mardani, A. Jusoh, K. M. Nor, Z. Khalifah, N. Zakwan and A. Valipour, "Multiple criteria decision-making techniques and their applications – a review of the literature from 2000 to 2014," *Economic research,* vol. 28, no. 1, pp. 516-571, 2015.

[154] E. K. Zavadskas, Z. Turskis and S. Kildienė, "State of art surveys of overviews on MCDM/MADM methods," *Technological and Economic Development of Economy,* vol. 20, no. 1, pp. 165-179, 2014.

[155] E. K. Zavadskas, J. Antucheviciene, H. Adeli, Z. Turskis and H. Adeli, "Hybrid multiple criteria decision making methods: a review of applications in engineering," *Scientia Iranica,* vol. 23, no. 1, pp. 1-20, 2016.

[156] A. Zhou, S. Wang, J. Li, Q. Sun and F. Yang, "Optimal mobile device selection for mobile cloud service providing," *Journal of Supercomputer,* vol. 72, no. 8, pp. 3222-3235, 2016.

[157] B. Venkatraman, F. A. Zaman and A. Karmouch, "Optimization of device selection in a mobile ad-hoc cloud based on composition score," in *2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, Mumbai, India, 2017.

[158] H. Viswanathan, E. K. Lee and D. Pompili, "Mobile grid computing for data- and patient-centric ubiquitous healthcare," in *1st IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT)*, Seoul, Korea (South), 2012.

[159] M. Hosseinzadeh, H. K. Hama, M. Y. Ghafour, M. Masdari, O. H. Ahmed and H. Khezri, "Service selection using multi-criteria decision making: a comprehensive overview," *Journal of Network and Systems Management,* vol. 28, pp. 1639-1693, 2020.

[160] P. Bagga, A. Joshi and R. Hans, "QoS based web service selection and multi-criteria decision making methods," *International Journal of Interactive Multimedia & Artificial Intelligence,* vol. 5, no. 4, pp. 113-121, 2019.

[161] I. Grgurević and G. Kordić, "Multi-criteria decision-making in cloud service selection and adoption," in *5th International Virtual Research Conference In Technical Disciplines*, Žilina, Slovak Republic, 2017.

[162] H. Alabool, A. Kamil, N. Arshad and D. Alarabiat, "Cloud service evaluation method-based multi-criteria decision-making: a systematic literature review," *Journal of Systems and Software,* vol. 139, pp. 161-188, 2018.

[163] G. Büyüközkan, F. Göçer and O. Feyzioğlu, "Cloud computing technology selection based on interval-valued intuitionistic fuzzy MCDM methods," *Soft Computing,* vol. 22, pp. 5091-5114, 2018.

[164] A. E. Youssef, "An integrated MCDM approach for cloud service selection based on TOPSIS and BWM," *IEEE Access,* vol. 8, pp. 71851-71865, 2020.

[165] C. Singla, N. Mahajan, S. Kaushal, A. Verma and A. K. Sangaiah, "Modelling and analysis of multi-objective service selection scheme in IoT-cloud environment," in *Cognitive Computing for Big Data Systems Over IoT. Lecture Notes on Data Engineering and Communications Technologies*, vol. 14, A. Sangaiah, A. Thangavelu and V. Meenakshi Sundaram, Eds., Springer, Cham, 2018, pp. 63-77.

[166] H. Wu, "Multi-objective decision-making for mobile cloud offloading: a survey," *IEEE Access,* vol. 6, pp. 3962-3976, 2018.

[167] H. Bangui, M. Ge, B. Buhnova, S. Rakrak, S. Raghay and T. Pitner, "Multi-criteria decision analysis methods in the mobile cloud offloading paradigm," *Journal of Sensor and Actuator Networks,* vol. 6, no. 4, p. 25, 2017.

[168] A. Ravi and S. K. Peddoju, "Handoff strategy for improving energy efficiency and cloud service availability for mobile devices," *Wireless Personal Communications,* vol. 81, pp. 101-132, 2015.

[169] M. K. Mishra, N. K. Ray, A. R. Swain, G. B. Mund and B. S. P. Mishra, "An adaptive model for resource selection and allocation in fog computing environment," *Computers & Electrical Engineering,* vol. 77, pp. 217-229, 2019.

[170] A. A. Gad-ElRab and A. S. Alsharkawy, "Multiple criteria-based efficient schemes for participants selection in mobile crowd sensing," *International Journal of Communication Networks and Distributed Systems,* vol. 21, no. 3, pp. 384-417, 2018.

[171] S. Mohammadi, S. Homayoun and E. T. Zadeh, "Grid computing: strategic decision making in resource selection," *International Journal of Computer Science Engineering and Applications,* vol. 2, no. 6, pp. 1-12, 2012.

[172] A. M. Abdullah, H. A. Ali and A. Y. Haikal, "A reliable, TOPSIS-based multi-criteria, and hierarchical load balancing method for computational grid," *Cluster Computing,* vol. 22, pp. 1085-1106, 2019.

[173] M. Kaur and S. S. Kadam, "Discovery of resources using MADM approaches for parallel and distributed computing," *Engineering Science and Technology, an International Journal,* vol. 20, pp. 1013-1024, 2017.

[174] W. N. S. W. Nik, B. B. Zhou, J. H. Abawajy and A. Y. Zomaya, "Cost and performance-based resource selection scheme for asynchronous replicated system in utility-based computing environment," *International Journal on Advanced Science, Engineering and Information Technology,* vol. 7, no. 2, pp. 723-735, 2017.

[175] A. Yildiz and E. U. Ergül, "A two-phased multi-criteria decision-making approach for selecting the best smartphone," *The South African Journal of Industrial Engineering,* vol. 26, no. 3, 2015.

[176] G. Büyüközkan and S. Güleryüz, "Multi criteria group decision making approach for smart phone selection using intuitionistic fuzzy TOPSIS," *International Journal of Computational Intelligence Systems,* vol. 9, no. 4, pp. 709-725, 2016.

[177] S. S. Goswami and D. K. Behera, "Evaluation of the best smartphone model in the market by integrating fuzzy-AHP and PROMETHEE decision-making approach," *DECISION: Official Journal of the Indian Institute of Management Calcutta,* vol. 48, no. 1, pp. 71-96, 2021.

[178] S. Kumar, S. K. Singh, T. Ashwin Kumar and S. Agrawal, "Research methodology: prioritization of new smartphones using TOPSIS and MOORA," in *International Conference of Advance Research & Innovation (ICARI)*, Meerut, India, 2020.

[179] A. Aggarwal, C. Choudhary and D. Mehrotra, "Evaluation of smartphones in Indian market using EDAS," *Procedia Computer Science,* vol. 132, pp. 236-243, 2018.

[180] I. Irvanizam, M. Marzuki, I. Patria and R. Abubakar, "An application for smartphone preference using TODIM decision making method," in *International Conference on Electrical Engineering and Informatics (ICELTICs)*, Banda Aceh, Indonesia, 2018.

[181] A. Q. Abdulhadi, "Selection a new mobile phone by utilize the voting method, AHP and enhance TOPSIS," *International Journal Academic Research in Business and Social Sciences,* vol. 10, no. 8, pp. 717-732, 2020.

[182] E. Triantaphyllou, Multi-criteria decision making methods: a comparative study, Springer, Boston, MA, 2000.

[183] M. Velasquez and P. T. Hester, "An analysis of multi-criteria decision making methods," *International Journal of Operations Research,* vol. 10, no. 2, pp. 56-66, 2013.

[184] S. H. Zanakis, A. Solomon, N. Wisharta and S. Dublish, "Multi-attribute decision making: a simulation comparison of select methods," *European Journal of Operational Research,* vol. 107, no. 3, pp. 507-529, 1998.

[185] J. A. B. Ruby Annette and P. Subash Chandran, "Comparison of multi criteria decision making algorithms for ranking cloud renderfarm services," *Indian Journal of Science and Technology,* vol. 9, no. 31, 2016.

[186] A. Piegat and W. Sałabun, "Comparative analysis of MCDM methods for assessing the severity of chronic liver disease," in *Artificial Intelligence and Soft Computing (ICAISC 2015). Lecture Notes in Computer Science,* vol. 9119, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh and J. Zurada, Eds., Springer, Cham, 2015, pp. 228-238.

[187] M. Mathew and S. Sahu, "Comparison of new multi-criteria decision making methods for material handling equipment selection," *Management Science Letters,* vol. 8, pp. 139-150, 2018.

[188] I. Ghosh and S. Biswas, "A comparative analysis of multi-criteria decision models for ERP package selection for improving supply chain performance," *Asia-Pacific Journal of Management Research and Innovation,* vol. 12, no. 3-4, pp. 250-270, 2016.

[189] A. Nesticò and P. Somma, "Comparative analysis of multi-criteria methods for the enhancement of historical buildings," *Sustainability,* vol. 11, no. 17, p. 4526, 2019.

[190] M. Moradian, V. Modanloo and S. Aghaiee, "Comparative analysis of multi criteria decision making techniques for material selection of brake booster valve body," *Journal of Traffic and Transportation Engineering,* vol. 6, no. 5, pp. 526-534, 2019.

[191] A. M. Ghaleb, H. Kaid, A. Alsamhan, S. H. Mian and L. Hidri, "Assessment and comparison of various MCDM approaches in the selection of manufacturing process," *Advances in Materials Science and Engineering,* vol. 2020 (Article ID 4039253), 2020.

[192] B. Ceballos, M. T. Lamata and D. A. Pelta, "A comparative analysis of multi-criteria decision-making methods," *Progress in Artificial Intelligence,* vol. 5, no. 4, pp. 315-322, 2016.

[193] E. Mulliner, N. Malys and V. Maliene, "Comparative analysis of MCDM methods for the assessment of sustainable housing affordability," *Omega,* vol. 59, no. Part B, pp. 146-156, 2016.

[194] M. Vuković, S. Pivac and Z. Babić, "Comparative analysis of stock selection using a hybrid MCDM approach and modern portfolio theory," *Croatian Review of Economic, Business and Social Statistics,* vol. 6, no. 2, pp. 58-68, 2020.

[195] W. Sałabun and A. Piegat, "Comparative analysis of MCDM methods for the assessment of mortality in patients with acute coronary syndrome," *Artificial Intelligence Review,* vol. 48, pp. 557-571, 2017.

[196] A. Valipour, H. Sarvari and J. Tamošaitiene, "Risk assessment in PPP projects by applying different MCDM methods and comparative results analysis," *Administrative Sciences,* vol. 8, no. 4, p. 80, 2018.

[197] H.-C. Lee and C.-T. Chang, "Comparative analysis of MCDM methods for ranking renewable energy sources in Taiwan," *Renewable and Sustainable Energy Reviews,* vol. 92, pp. 883-896, 2018.

[198] P. Karande, E. K. Zavadskas and S. Chakraborty, "A study on the ranking performance of some MCDM methods for industrial robot selection problems," *International Journal of Industrial Engineering Computations,* vol. 7, no. 3, pp. 399-422, 2016.

[199] E. Harirchian, K. Jadhav, K. Mohammad, S. E. A. Hosseini and T. Lahmer, "A comparative study of MCDM methods integrated with rapid visual seismic vulnerability assessment of existing RC structures," *Applied Sciences,* vol. 10, no. 18, p. 6411, 2020.

[200] J. Sidhu and S. Singh, "Design and comparative analysis of MCDM-based multi-dimensional trust evaluation schemes for determining trustworthiness of cloud service providers," *Journal of Grid Computing,* vol. 15, pp. 197-218, 2017.

[201] S. A. A. Alrababah, K. H. Gan and T.-P. Tan, "Comparative analysis of MCDM methods for product aspect ranking: TOPSIS and VIKOR," in *8th International Conference on Information and Communication Systems (ICICS),* Irbid, 2017.

[202] S. K. Kaya, "Evaluation of the effect of COVID-19 on countries' sustainable development level: a comparative MCDM framework," *Operational Research in Engineering Sciences: Theory and Applications,* vol. 3, no. 3, pp. 101-122, 2020.

[203] T. Li, A. Li and X. Guo, "The sustainable development-oriented development and utilization of renewable energy industry - a comprehensive analysis of MCDM methods," *Energy,* vol. 212, p. 118694, 2020.

[204] R. Sun, Z. Gong, G. Gao and A. A. Shah, "Comparative analysis of multi-criteria decision-making methods for flood disaster risk in the Yangtze River Delta," *International Journal of Disaster Risk Reduction,* vol. 51, p. 101768, 2020.

[205] F. Antoniou and G. N. Aretoulis, "Comparative analysis of multi-criteria decision making methods in choosing contract type for highway construction in Greece," *International Journal of Management and Decision Making,* vol. 17, no. 1, pp. 1-28, 2018.

[206] P. Madhu, C. S. Dhanalakshmi and M. Mathew, "Multi-criteria decision-making in the selection of a suitable biomass material for maximum bio-oil yield during pyrolysis," *Fuel,* vol. 277, p. 118109, 2020.

[207] S. Hezer, E. Gelmez and E. Özceylan, "Comparative analysis of TOPSIS, VIKOR and COPRAS methods for the COVID-19 regional safety assessment," *Journal of Infection and Public Health,* vol. 14, no. 6, pp. 775-786, 2021.

[208] M. Keshavarz-Ghorabaee, E. K. Zavadskas, Z. Turskis and J. Antucheviciene, "A comparative analysis of the rank reversal phenomenon in the EDAS and TOPSIS methods," *Economic Computation and Economic Cybernetics Studies and Research,* vol. 52, no. 3, pp. 121-134, 2018.

[209] N. Kokaraki, C. J. Hopfe, E. Robinson and E. Nikolaidou, "Testing the reliability of deterministic multi-criteria decision-making methods using building performance simulation," *Renewable and Sustainable Energy Reviews,* vol. 112, pp. 991-1007, 2019.

[210] S. Dožić and M. Kalić, "Aircraft type selection problem: application of different MCDM methods," in *Advanced Concepts, Methodologies and Technologies for Transportation and Logistics (EURO 2016, EWGT 2016). Advances in Intelligent Systems and Computing*, vol. 572, J. Żak, Y. Hadas and R. Rossi, Eds., Springer, Cham, 2018, pp. 156-175.

[211] C. Srisawat and J. Payakpate, "Comparison of MCDM methods for intercrop selection in rubber plantations," *Journal of Information and Communication Technology,* vol. 15, no. 1, pp. 165-182, 2016.

[212] M. M. D. Widianta, T. Rizaldi, D. P. S. Setyohadi and H. Y. Riskiawan, "Comparison of multi-criteria decision support methods (AHP, TOPSIS, SAW & PROMENTHEE) for employee placement," *Journal of Physics: Conf. Series,* vol. 953, p. 012116, 2018.

[213] B. C. Balusa and A. K. Gorai, "A comparative study of various multi-criteria decision-making models in underground mining method selection," *Journal of The Institution of Engineers (India): Series D,* vol. 100, pp. 105-121, 2019.

[214] A. Ishizaka and S. Siraj, "Are multi-criteria decision-making tools useful? An experimental comparative study of three methods," *European Journal of Operational Research,* vol. 264, no. 2, pp. 462-471, 2018.

[215] M. Alkahtani, A. Al-Ahmari, H. Kaid and M. Sonboa, "Comparison and evaluation of multi-criteria supplier selection approaches: a case study," *Advances in Mechanical Engineering,* vol. 11, no. 2, 2019.

[216] S. Vakilipour, A. Sadeghi-Niaraki, M. Ghodousi and S.-M. Choi, "Comparison between multi-criteria decision-making methods and evaluating the quality of life at different spatial levels," *Sustainability,* vol. 13, no. 7, p. 4067, 2021.

[217] R. E. Hodgett, "Comparison of multi-criteria decision-making methods for equipment selection," *The International Journal of Advanced Manufacturing Technology,* vol. 85, no. 5-8, pp. 1145-1157, 2016.

[218] F. Sari, "Forest fire susceptibility mapping via multi-criteria decision analysis techniques for Mugla, Turkey: a comparative analysis of VIKOR and TOPSIS," *Forest Ecology and Management,* vol. 480, p. 118644, 2021.

[219] S. Biswas, "Measuring performance of healthcare supply chains in India: A comparative analysis of multi-criteria decision making methods," *Decision Making: Applications in Management and Engineering,* vol. 3, no. 2, pp. 162-189, 2020.

[220] J. Anitha and R. Das, "A comparative analysis of multi-criteria decision-making techniques to optimize the process parameters in electro discharge machine," in *Recent Trends in Mechanical Engineering. Lecture Notes in Mechanical Engineering*, G. S. V. L. Narasimham, A. V. Babu, S. S. Reddy and R. Dhanasekaran, Eds., Springer, Singapore, 2021, pp. 675-686.

[221] R. K. Dewi, B. T. Hanggara and A. Pinandito, "A comparison between AHP and Hybrid AHP for mobile based culinary recommendation system," *International Journal of Interactive Mobile Technologies,* vol. 12, no. 1, pp. 133-140, 2018.

[222] A. Martin, T. Miranda Lakshmi and V. Prasanna Venkatesan, “A study on evaluation metrics for multi criteria decision making (MCDM) methods - TOPSIS, COPRAS & GRA,” *International Journal of Computing Algorithm,* vol. 7, no. 1, pp. 29-37, 2018.

[223] Z. Wu and G. Abdul-Nour, “Comparison of multi-criteria group decision-making methods for urban sewer network plan selection,” *CivilEng,* vol. 1, no. 1, pp. 26-48, 2020.

[224] A. Jozaghi, B. Alizadeh, M. Hatami, I. Flood, M. Khorrami, N. Khodaei and E. G. Tousi, “A comparative study of the AHP and TOPSIS techniques for dam site selection using GIS: a case study of Sistan and Baluchestan Province, Iran,” *Geosciences,* vol. 8, no. 12, p. 494, 2018.

[225] A. Kishor and R. Niyogi, “An evolutionary approach for optimal multi-objective resource allocation in distributed computing systems,” *Concurrent Engineering,* vol. 28, no. 2, pp. 97-109, 2020.

[226] F. Xhafa and A. Abraham, “Meta-heuristics for Grid Scheduling Problems,” in *Metaheuristics for Scheduling in Distributed Computing Environments. Studies in Computational Intelligence*, vol. 146, F. Xhafa and A. Abraham, Eds., Berlin, Heidelberg, Springer, 2008, pp. 1-37.

[227] A. Kuijl, M. Emmerich and H. Li, “A novel multi-objective optimization scheme for grid resource allocation,” in *6th international workshop on Middleware for grid computing (MGC '08)*, 2008.

[228] J. Chen, T. Du and G. Xiao, “A multi-objective optimization for resource allocation of emergent demands in cloud computing,” *Journal of Cloud Computing,* vol. 10 (Article number: 20), 2021.

[229] B. Shrimali and H. Patel, “Multi-objective optimization oriented policy for performance and energy efficient resource allocation in Cloud environment,” *Journal of King Saud University - Computer and Information Sciences,* vol. 32, no. 7, pp. 860-869, 2020.

[230] E. S. Alkayal, N. R. Jennings and M. F. Abulkhair, “Efficient task scheduling multi-objective particle swarm optimization in cloud computing,” in *IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops)*, Dubai, UAE, 2016.

[231] H. Wu, S. Deng, W. Li, M. Fu, J. Yin and A. Y. Zomaya, “Service selection for composition in mobile edge computing systems,” in *IEEE International Conference on Web Services (ICWS)*, San Francisco, USA, 2018.

[232] S. Midya, A. Roy, K. Majumder and S. Phadikar, “Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach,” *Journal of Network and Computer Applications,* vol. 103, pp. 58-84, 2018.

[233] X. Xu, R. Gu, F. Dai, L. Qi and S. Wan, “Multi-objective computation offloading for Internet of Vehicles in cloud-edge computing,” *Wireless Networks,* vol. 26, pp. 1611-1629, 2020.

[234] N. Bao, J. Zuo, H. Zhu and X. Bao, “Multi-objective optimization for SDN based resource selection,” in *IEEE 18th International Conference on Communication Technology (ICCT)*, Chongqing, China, 2018.

[235] H. Zhu, L. He and S. A. Jarvis, “Optimizing job scheduling on multicore computers,” in *22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, Paris, France, 2014.

[236] G. Wang, Y. Wang, H. Liu and H. Guo, “HSIP: a novel task scheduling algorithm for heterogeneous computing,” *Scientific Programming,* vol. 2016 (Article ID 3676149), 2016.

[237] M. Orr and O. Sinnen, “Optimal task scheduling for partially heterogeneous systems,” *Parallel Computing,* vol. 107, p. 102815, 2021.

[238] M. Akbari, H. Rashidi and S. H. Alizadeh, “An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems,” *Engineering Applications of Artificial Intelligence,* vol. 61, pp. 35-46, 2017.

[239] M. Sulaiman, Z. Halim, M. Lebbah, M. Waqas and S. Tu, “An evolutionary computing-based efficient hybrid task scheduling approach for heterogeneous computing environment,” *Journal of Grid Computing,* vol. 19 (Article number: 11), 2021.

[240] T. Biswas, P. Kuila and A. K. Ray, “A novel resource aware scheduling with multi-criteria for heterogeneous computing systems,” *Engineering Science and Technology, an International Journal,* vol. 22, p. 646–655, 2019.

[241] T. Biswas, P. Kuila and A. K. Ray, “A novel workflow scheduling with multi-criteria using particle swarm optimization for heterogeneous computing systems,” *Cluster Computing,* vol. 23, p. 3255–3271, 2020.

[242] T. Biswas, P. Kuila and A. K. Ray, "A novel scheduling with multi-criteria for high-performance computing systems: an improved genetic algorithm-based approach," *Engineering with Computers,* vol. 35, no. 4, p. 1475–1490, 2019.

[243] L. Zhang, K. Li, C. Li and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems," *Information Sciences,* vol. 379, pp. 241-256, 2017.

[244] T. Biswas, P. Kuila and A. K. Ray, "A novel energy efficient scheduling for high performance computing systems," in *9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Bengaluru, India, 2018.

[245] E. Gabaldon, F. Guirado, J. L. Lerida and J. Planes, "Particle swarm optimization scheduling for energy saving in cluster computing heterogeneous environments," in *4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Vienna, Austria, 2016.

[246] J. Hao, M. Xian, H. Wang, F. Tang and P. Xiao, "Mobile cloud computing: the state of art, application scenarios and challenges," in *4th International Conference on Computational Intelligence & Communication Technology (CICT)*, Ghaziabad, India, 2018.

[247] D. De, Mobile cloud computing: architectures, algorithms and applications, Boca Raton, FL: Chapman and Hall/CRC, 2015.

[248] S. C. Shah, "Mobile ad hoc computational grid: opportunities and challenges," in *IEEE Military Communications Conference*, San Diego, USA, 2013.

[249] P. K. D. Pramanik and P. Choudhury, "Mobility-aware service provisioning for delay tolerant applications in a mobile crowd computing environment," *SN Applied Sciences,* vol. 2, no. 3 (Article no. 403), pp. 1-17, 2020.

[250] H. Qian and D. Andresen, "An energy-saving task scheduler for mobile devices," in *IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, Las Vegas, USA, 2015.

[251] A. Ali, M. M. Iqbal, H. Jamil, F. Qayyum, S. Jabbar, O. Cheikhrouhou, M. Baz and F. Jamil, "An efficient dynamic-decision based task scheduler for task offloading optimization and energy management in mobile cloud computing," *Sensors (Basel),* vol. 21, no. 13, p. 4527, 2021.

[252] T. Liu, F. Chen, Y. Ma and Y. Xie, "An energy-efficient task scheduling for mobile devices based on cloud assistant," *Future Generation Computer Systems,* vol. 61, pp. 1-12, 2016.

[253] S. C. Shah and M.-S. Park, "An energy-efficient resource allocation scheme for mobile ad hoc computational grids," *Journal of Grid Computing,* vol. 9, pp. 303-323, 2011.

[254] S. C. Shah, Q.-U.-A. Nizamani, S. H. Chauhdary and M.-S. Park, "An effective and robust two-phase resource allocation scheme for interdependent tasks in mobile ad hoc computational Grids," *Journal of Parallel and Distributed Computing,* vol. 72, no. 12, pp. 1664-1679, 2012.

[255] W. Chen, C. T. Lea and L. Kenli, "Dynamic resource allocation in ad-hoc mobile cloud computing," in *IEEE Wireless Communications and Networking Conference (WCNC)*, San Francisco, USA, 2017.

[256] H. Bonan, X. Weiwei, Y. Zhang, Z. Qian, Y. Feng and L. Shen, "Dependent task assignment algorithm based on swarm optimization and simulated annealing in ad-hoc mobile cloud," *journal of southeast university (English Edition),* vol. 34, no. 4, pp. 430-438, 2018.

[257] T. Shi, M. Yang, X. Li, Q. Lei and Y. Jiang, "An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds," *Pervasive and Mobile Computing,* vol. 27, pp. 90-105, 2016.

[258] S. C. Shah, S. H. Chauhdary, A. K. Bashir and M. S. Park, "A centralized locationbased based job scheduling algorithm for interdependent jobs in mobile ad hoc computational grids," *Journal of Applied Sciences,* vol. 10, no. 3, p. 174–181, 2010.

[259] H. Kim, Y. e. Khamra, I. Rodero, S. Jha and M. Parashar, "Autonomic management of application workflows on hybrid computing infrastructure," *Telecomm. Sys.,* vol. 19, no. 2-3, p. 75–89, 2011.

[260] X. Wang, Y. Sui, C. Yuen, X. Chen and C. Wang, "Traffic-aware task allocation for cooperative execution in mobile cloud computing," in *IEEE/CIC International Conference on Communications in China (ICCC)*, Chengdu, China, 2016.

[261] H. Topcuoglu, S. Hariri and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 13, no. 3, pp. 260-274, 2002.

[262] S. Gupta, V. Kumar and G. Agarwal, "Task scheduling in multiprocessor system using genetic algorithm," in *Second International Conference on Machine Learning and Computing*, Bangalore, India, 2010.

[263] P. Damodaran and M. C. Vélez-Gallego, "A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times," *Expert Systems with Applications: An International Journal* , vol. 39, no. 1, pp. 1451-1458, 2012.

[264] X. Zuo, G. Zhang and W. Tan, "Self-adaptive learning PSO-based deadline," *IEEE Transactions on Automation Science and Engineering,* vol. 11, no. 2, pp. 564-573, 2014.

[265] B. Keshanchi, A. Souri and N. J. Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing," *Journal of Systems and Software,* vol. 124, pp. 1-21, 2017.

[266] H. Arabnejad, "List based task scheduling algorithms on heterogeneous systems - an overview," *Doctoral Symposium in Informatics Engineering,* vol. 93, 2013.

[267] J. Brevik, D. Nurmi and R. Wolski, "Automatic methods for predicting machine availability in desktop Grid and peer-to-peer systems," in *IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2004)*, Chicago, USA, 2004.

[268] A. Andrzejak, D. Kondo and D. P. Anderson, "Ensuring collective availability in volatile resource pools via forecasting," in *Managing Large-Scale Service Deployment (DSOM 2008), Lecture Notes in Computer Science*, vol. 5273, F. De Turck, W. Kellerer and G. Kormentzas, Eds., Berlin, Heidelberg, Springer, 2008, pp. 149-161.

[269] S. S. Vaithiya and S. M. S. Bhanu, "Mobility and battery power prediction based job scheduling in mobile grid environment," in *International Conference on Parallel Distributed Computing Technologies and Applications (PDCTA 2011)*, 2011.

[270] V. V. Selvi, S. Sharfraz and R. Parthasarathi, "Mobile ad hoc grid using trace based mobility model," in *Advances in Grid and Pervasive Computing (GPC 2007)*, Paris, France, 2007.

[271] M. Á. Sipos and P. Ekler, "Predicting availability of mobile peers in large peer-to-peer networks," in *3rd Eastern European Regional Conference on the Engineering of Computer Based Systems*, Budapest, Hungary, 2013.

[272] S. C. Haryanti and R. F. Sari, "Improving resource allocation performance in mobile ad hoc grid with mobility prediction," in *International Conference on Intelligent Green Building and Smart Grid (IGBSG)*, Taipei, Taiwan, 2014.

[273] S. Dargan, M. Kumar, M. R. Ayyagari and G. Kumar, "A survey of deep learning and its applications: a new paradigm to machine learning," *Archives of Computational Methods in Engineering,* vol. 27, pp. 1071-1092, 2020.

[274] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen and S. S. Iyengar, "A survey on deep learning: algorithms, techniques, and applications," *ACM Computing Surveys,* vol. 51, no. 5, pp. 1-36, 2019.

[275] C. A. Dhawale, K. Dhawale and R. Dubey, "A review on deep learning applications," in *Deep Learning Techniques and Optimization Strategies in Big Data Analytics*, J. J. Thomas, P. Karagoz, B. B. Ahamed and P. Vasant, Eds., USA, IGI GLobal, 2020, pp. 21-31.

[276] S. Khan and T. Yairi, "A review on the application of deep learning in system health management," *Mechanical Systems and Signal Processing,* vol. 107, pp. 241-265, 2018.

[277] R. Miotto, F. Wang, S. Wang, X. Jiang and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in Bioinformatics,* vol. 19, no. 6, pp. 1236-1246, 2018.

[278] B. Yang and Y. Xu, "Applications of deep-learning approaches in horticultural research: a review," *Horticulture Research,* vol. 8 (Article number: 123), 2021.

[279] A. Carrio, C. Sampedro, A. Rodriguez-Ramos and P. Campoy, "A review of deep learning methods and applications for unmanned aerial vehicles," *Journal of Sensors,* vol. 2017 (Article ID 3296874), 2017.

[280] J. Huang, J. Chai and S. Cho, "Deep learning in finance and banking: A literature review and classification," *Frontiers of Business Research in China,* vol. 14 (Article number: 13) , 2020.

[281] A. Sarraf, M. Azhdari and S. Sarraf, "A comprehensive review of deep learning architectures for computer vision applications," *American Scientific Research Journal for Engineering, Technology, and Sciences,* vol. 77, no. 1, pp. 1-29, 2021.

[282] Y. Zhang, J. Yan, S. Chen, M. Gong, D. Gao, M. Zhu and W. Gan, "Review of the applications of deep learning in bioinformatics," *Current Bioinformatics,* vol. 15, no. 8, pp. 898-911, 2020.

[283] M. I. Tariq, N. A. Memon, S. Ahmed, S. Tayyaba, M. T. Mushtaq, N. A. Mian, M. Imran and M. W. Ashraf, "A review of deep learning security and privacy defensive techniques," *Mobile Information Systems,* vol. 2020 (Article ID 6535834), 2020.

[284] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing,* vol. 323, no. 5, pp. 203-213, 2019.

[285] J. Choi and B. Lee, "Combining LSTM network ensemble via adaptive weighting for improved time series forecasting," *Mathematical Problems in Engineering,* vol. 2018 (Article ID 2470171), 2018.

[286] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems,* vol. 11, no. 2, 3, pp. 68-75, 2017.

[287] P. B. Weerakody, K. W. Wong, G. Wang and W. Ela, "A review of irregular time series data handling with gated recurrent neural networks," *Neurocomputing,* vol. 441, pp. 161-178, 2021.

[288] Z. Che, S. Purushotham, K. Cho, D. Sontag and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific Reports,* vol. 8, p. 6085, 2018.

[289] K. Lu, X. R. Meng, W. X. Sun, R. G. Zhang, Y. K. Han, S. Gao and D. Su, "GRU-based encoder-decoder for short-term CHP heat load forecast," *IOP Conference Series: Materials Science and Engineering,* vol. 392, no. 6, p. 062173, 2018.

[290] N. Xue, I. Triguero, G. P. Figueredo and D. Landa-Silva, "Evolving deep CNN-LSTMs for inventory time series prediction," in *IEEE Congress on Evolutionary Computation (CEC)*, Wellington, New Zealand, 2019.

[291] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, India, 2017.

[292] J. M.-T. Wu, Z. Li, N. Herencsar, B. Vo and J. C.-W. Lin, "A graph-based CNN-LSTM stock price prediction algorithm with leading indicators," *Multimedia Systems,* 2021.

[293] T. Kim and H. Y. Kim, "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data," *PLoS ONE,* vol. 14, no. 2, p. e0212320, 2019.

[294] W. Lu, J. Li, Y. Li, A. Sun and J. Wang, "A CNN-LSTM-based model to forecast stock prices," *Complexity,* vol. 2020 (Article ID 6622927), 2020.

[295] I. E. Livieris, E. Pintelas and P. Pintelas, "A CNN–LSTM model for gold price time-series forecasting," *Neural Computing and Applications,* vol. 32, pp. 17351-17360, 2020.

[296] Y. Li and W. Dai, "Bitcoin price forecasting method based on CNN-LSTM hybrid neural network model," *The Journal of Engineering,* vol. 2020, no. 13, pp. 344-347, 2020.

[297] T. Ni, L. Wang, P. Zhang, B. Wang and W. Li, "Daily tourist flow forecasting using SPCA and CNN-LSTM neural network," *Concurrency and Computation: Practice and Experience,* vol. 33, no. 5, p. e5980, 2021.

[298] J. Zhao, J. Lin, S. Liang and M. Wang, "Sentimental prediction model of personality based on CNN-LSTM in a social media environment," *Journal of Intelligent & Fuzzy Systems,* vol. 40, no. 2, pp. 3097-3106, 2021.

[299] T.-Y. Kim and S.-B. Cho, "Predicting the household power consumption using CNN-LSTM hybrid networks," in *Intelligent Data Engineering and Automated Learning (IDEAL 2018). Lecture Notes in Computer Science,* vol. 11314, H. Yin, D. Camacho, P. Novais and A. Tallón-Ballesteros, Eds., Springer, Cham, 2018, pp. 481-490.

[300] T.-Y. Kim and S.-B. Cho, "Predicting residential energy consumption using CNN-LSTM neural networks," *Energy,* vol. 182, no. Sept, pp. 72-81, 2019.

[301] M. Tovar, M. Robles and F. Rashid, "PV power prediction, using CNN-LSTM hybrid neural network model. Case of study: Temixco-Morelos, México," *Energies,* vol. 13, no. 24, p. 6512, 2020.

[302] H. Kuang, Q. Guo, S. Li and H. Zhong, "Short-term wind power forecasting model based on multi-feature extraction and CNN-LSTM," *IOP Conference Series: Earth and Environmental Science,* vol. 702, p. 012019, 2021.

[303] C. Ding, G. Wang, X. Zhang, Q. Liu and X. Liu, "A hybrid CNN-LSTM model for predicting PM2.5 in Beijing based on spatiotemporal correlation," *Environmental and Ecological Statistics,* vol. 28, pp. 503-522, 2021.

[304] T. Li, M. Hua and X. Wu, "A hybrid CNN-LSTM model for forecasting particulate matter (PM2.5)," *IEEE Access,* vol. 8, pp. 26933-26940, 2020.

[305] W. He, J. Li, Z. Tang, B. Wu, H. Luan, C. Chen and H. Liang, "A novel hybrid CNN-LSTM scheme for nitrogen oxide emission prediction in FCC unit," *Mathematical Problems in Engineering,* vol. 2020 (Article ID 8071810), 2020.

[306] I. E. Livieris, E. Pintelas, N. Kiriakidou and S. Stavroyiannis, "An advanced deep learning model for short-term forecasting U.S. natural gas price and movement," in *Applications and Innovations (AIAI 2020), IFIP WG 12.5, International Workshopsmhdw 2020 and 5G-PINE 2020,* Neos Marmaras, Greece, 2020.

[307] W. Boulila, H. Ghandorh, M. A. Khan, F. Ahmed and J. Ahmad, "A novel CNN-LSTM-based approach to predict urban expansion," *Ecological Informatics,* vol. 64, no. Sept, p. 101325, 2021.

[308] K. Cao, H. Kim, C. Hwang and H. Jung, "CNN-LSTM coupled model for prediction of waterworks operation data," *Journal of Information Processing Systems,* vol. 14, no. 6, pp. 1508-1520, 2018.

[309] P. K. Jonnakuti and U. B. T. V. Sai, "A hybrid CNN-LSTM based model for the prediction of sea surface temperature using time-series satellite data," in *22nd EGU General Assembly (EGU2020-817),* Online, 2020.

[310] R. Chen, X. Wang, W. Zhang, X. Zhu, A. Li and C. Yang, "A hybrid CNN-LSTM model for typhoon formation forecasting," *Geoinformatica,* vol. 23, no. 3, pp. 375-396, 2019.

[311] S. Khaki, L. Wang and S. V. Archontoulis, "A CNN-RNN framework for crop yield prediction," *Frontiers in Plant Science,* vol. 10, p. 1750, 2020.

[312] M. ZabirulIslam, M. M. Islam and A. Asraf, "A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using X-ray images," *Informatics in Medicine Unlocked,* vol. 20, p. 100412, 2020.

[313] A. G. Dastider, F. Sadik and S. A. Fattah, "An integrated autoencoder-based hybrid CNN-LSTM model for COVID-19 severity prediction from lung ultrasound," *Computers in Biology and Medicine,* vol. 132, no. May, p. 104296, 2021.

[314] S. Dutta, S. K. Bandyopadhyay and T.-H. Kim, "CNN-LSTM model for verifying predictions of Covid-19 cases," *Asian Journal of Research in Computer Science,* vol. 5, no. 4, pp. 25-32, 2020.

[315] S. A. Rahman and D. A. Adjeroh, "Deep learning using convolutional LSTM estimates biological age from physical activity," *Scientific Reports,* vol. 9 (Article number: 11425), 2019.

[316] Y. Zhang, J. Yao and H. Guan, "Intelligent cloud resource management with deep reinforcement learning," *IEEE Cloud Computing,* vol. 4, pp. 60-69, 2017.

[317] Y. Lu, L. Liu, J. Panneerselvam, B. Yuan, J. Gu and N. Antonopoulos, "A GRU-based prediction framework for intelligent resource management at cloud data centres in the age of 5G," *IEEE Transactions on Cognitive Communications and Networking,* vol. 6, no. 2, pp. 486-498, 2020.

[318] H. Jing, Y. Zhang, J. Zhou, W. Zhang, X. Liu, G. Min and Z. Zhang, "LSTM-based service migration for pervasive cloud computing," in *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData),* Halifax, Canada, 2018.

[319] Y. Zhu, W. Zhang, Y. Chen and H. Gao, "A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment," *EURASIP Journal on Wireless Communications and Networking,* vol. 2019 (Article number: 274), 2019.

[320] J. Kumar, R. Goomer and A. K. Singh, "Long Short Term Memory Recurrent Neural Network (LSTM-RNN) Based Workload Forecasting Model For Cloud Datacenters," *Procedia Computer Science,* vol. 125, pp. 676-682, 2018.

[321] Anupama K C, Shivakumar B R, Nagaraja R, "Resource utilization prediction in cloud computing using hybrid model," *International Journal of Advanced Computer Science and Applications,* vol. 12, no. 4, pp. 373-381, 2021.

[322] H. Li, K. Ota and M. Dong, "Learning IoT in edge: deep learning for the Internet of Things with edge computing," *IEEE Network,* vol. 32, no. 1, pp. 96-101, 2018.

[323] J. Shuja, K. Bilal, W. Alasmary, H. Sinky and E. Alanazi, "Applying machine learning techniques for caching in next-generation edge," *Journal of Network and Computer Applications,* vol. 181, p. 103005, 2021.

[324] J. Violos, E. Psomakelis, D. Danopoulos, S. Tsanakas and T. Varvarigou, "Using LSTM neural networks as resource utilization predictors: the case of training deep learning models on the edge," in *Economics of Grids, Clouds, Systems, and Services. GECON 2020. Lecture Notes in Computer*

Science, vol, vol. 12441, K. Djemame, J. Altmann, J. Á. Bañares, O. Agmon Ben-Yehuda, V. Stankovski and B. Tuffin, Eds., Springer, Cham, 2020, pp. 67-74.

[325] F. Hussain, S. A. Hassan, R. Hussain and E. Hossain, "Machine learning for resource management in cellular and IoT networks: potentials, current solutions, and open challenges," *IEEE Communications Surveys & Tutorials,* vol. 22, no. 2, pp. 1251-1275, 2020.

[326] B. Gu, X. Zhang, Z. Lin and M. Alazab, "Deep multi-agent reinforcement learning-based resource allocation for internet of controllable things," *IEEE Internet of Things Journal,* vol. 8, no. 5, pp. 3066-3074, 2021.

[327] K. Li, W. Ni, E. Tovar and A. Jamalipour, "Deep Q-learning based resource management in UAV-assisted wireless powered IoT networks," in *IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 2020.

[328] U. Challita, L. Dong and W. Saad, "Proactive resource management for LTE in unlicensed Spectrum: a deep learning perspective," *IEEE Transactions on Wireless Communications,* vol. 17, no. 7, pp. 4674-4689, 2018.

[329] R. C. Bhaddurgatte, B. P. Vijaya Kumar and S. M. Kusuma, "Machine learning and prediction-based resource management in IoT considering QoS," *International Journal of Recent Technology and Engineering,* vol. 8, no. 2, pp. 687-694, 2019.

[330] Y. Peng, G. Zhang, J. Shi, B. Xu and L. Zheng, "SRA-LSTM: social relationship attention LSTM for human trajectory prediction," *arXiv:2103.17045v1 [cs.CV],* 2021.

[331] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei and S. Savarese, "Social LSTM: human trajectory prediction in crowded spaces," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016.

[332] H. Manh and G. Alaghband, "Scene-LSTM: A model for human trajectory prediction," *arXiv:1808.04018v2 [cs.CV],* 2019.

[333] N. Nikhil and B. T. Morris, "Convolutional neural network for trajectory prediction," in *Computer Vision – ECCV 2018 Workshops. Lecture Notes in Computer Science*, vol. 11131, L. Leal-Taixé and S. Roth, Eds., Springer, Cham, 2019, pp. 186-196.

[334] X. Song, K. Chen, X. Li, J. Sun, B. Hou, Y. Cui, B. Zhang, G. Xiong and Z. Wang, "Pedestrian trajectory prediction based on deep convolutional LSTM network," *IEEE Transactions on Intelligent Transportation Systems,* 2020.

[335] G. Xie, A. Shangguan, R. Fei, W. Ji, W. Ma and X. Hei, "Motion trajectory prediction based on a CNN-LSTM sequential model," *SCIENCE CHINA Information Sciences,* vol. 63, no. 11, p. 212207, 2020.

[336] H. R. Pamuluri, "Predicting user mobility using deep learning methods," Master's Thesis, Department of Computer Science, Blekinge Institute of Technology, Karlskrona, Sweden, 2020.

[337] C. Cui, M. Zhao and K. Wong, "An LSTM-method-based availability prediction for optimized offloading in mobile edges," *Sensors (Basel),* vol. 19, no. 20, p. 4467, 2019.

[338] R. Li, C. Wang, Z. Zhao, R. Guo and H. Zhang, "The LSTM-based advantage actor-critic learning for resource management in network slicing with user mobility," *IEEE Communications Letters,* vol. 24, no. 9, pp. 2005-2009, 2020.

[339] S. C. Shah, A. K. Bashir, S. H. Chauhdary, C. Jiehui and M.-S. Park, "Mobile ad hoc computational grid for low constraint devices," in *International Conference on Future Computer and Communication*, Kuala Lumpar, Malaysia, 2009.

[340] P. G. Tiburcio and M. A. Spohn, "Ad hoc grid: an adaptive and self-organizing peer-to-peer computing grid," in *10th IEEE International Conference on Computer and Information Technology*, Bradford, UK, 2010.

[341] G. A. McGilvary, A. Barker and M. Atkinson, "Ad hoc cloud computing," in *IEEE 8th International Conference on Cloud Computing*, New York, USA, 2015.

[342] C. Barca, C. Barca, C. Cucu, M.-R. Gavriloaia, R. Vizireanu, O. Fratu and S. Halunga, "A virtual cloud computing provider for mobile devices," in *8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Ploiesti, Romania, 2016.

[343] M. Gonzlez, C. Hidalgo and A. L. Barabsi, "Understanding individual human mobility patterns," *Nature,* vol. 453, pp. 479-482, 2008.

[344] W. Navidi and T. Camp, "Stationary distributions for the random waypoint mobility model," *IEEE Transactions on Mobile Computing,* vol. 3, no. 1, pp. 99-108, 2004.

[345] J. Chan and A. Seneviratne, “A practical user mobility prediction algorithm for supporting adaptive QoS in wireless networks,” in *Seventh IEEE International Conference on Networks (ICON'99)*, 1999.

[346] G. Lium and G. Maguire Jr., “A class of mobile motion prediction algorithms for wireless mobile computing and communications,” *Mobile Networks Application,* vol. 1, no. 2, pp. 113-121, 1996.

[347] T. Liu, P. Bahl and I. Chlamtac, “Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks,” *IEEE Journal on Selected Areas in Communications,* vol. 16, no. 6, pp. 922- 936, 1998.

[348] D. Levine, I. Akyildiz and M. Naghshineh, “A resource estimation and call admission algorithm for wireless multimedia networks using the shadow cluster concept,” *IEEE/ACM Transactions on Networking,* vol. 5, no. 1, pp. 1-12, 1997.

[349] W. Su, S. J. Lee and M. Gerla, “Mobility prediction and routing in ad hoc wireless networks,” *International Journal on Network Management,* vol. 11, no. 1, pp. 3-30, 2001.

[350] A. Agarwal and S. R. Das, “Dead reckoning in mobile ad hoc networks,” in *IEEE Wireless Communications and Networking (WCNC)*, New Orleans, USA, 2003.

[351] T. Camp, J. Boleng and V. Davies, “A survey of mobility models for ad hoc network research,” *Wireless Communication & Mobile Computing,* vol. 2, no. 5, pp. 483-502, 2002.

[352] F. Erbas, J. Steuer, K. Kyamakya, D. Eggesieker and K. Jobmann, “A regular path recognition method and prediction of user movements in wireless networks,” in *IEEE Vehicular Technology Conference (VTC)*, 2001.

[353] W. Wang, X. Guan, B. Wang and Y. Wang, “A novel mobility model based on semi-random circular movement in mobile Ad hoc network,” *Journal of Information Sciences,* vol. 180, no. 3, pp. 399-413, 2010.

[354] T. Gross and C. Truduce, “A mobility model based on wlan traces and its validation,” in *24th Annual Joint Conference of the IEEE Computer and Communications Societies*, Miami, USA, 2005.

[355] M. Kim, D. Kotz and S. Kim, “Extracting a mobility model from real user traces,” in *25th IEEE International Conference on Computer Communication (INFOCOM)*, 2006.

[356] I. Khalifa and H. Abbas, “Mobility Prediction in Dynamic Grids,” *Journal on Computer and Information Science,* vol. 5, no. 3, 2012.

[357] L. Song, D. Kotz, R. Jain and X. He, “Evaluating location predictors with extensive Wi-Fi mobility data,” in *IEEE INFOCOM*, Hong Kong, 2004.

[358] I. Burbey and T. L. Martin, “Predicting future locations using prediction-by-partial-match,” in *First ACM international workshop on Mobile entity localization and tracking in GPS-less environments (MELT '08)*, San Francisco; USA, 2008.

[359] M. Musolesi and C. Mascolo, “Designing mobility models based on social network theory,” *ACM SIGMOBILE Mobile Computing and Communications Review,* vol. 11, no. 3, 2007.

[360] X. Li, N. Mitton and D. Simplot-Ryl, “Mobility prediction based neighborhood discovery in mobile ad hoc networks,” in *NETWORKING 2011. Lecture Notes in Computer Science* , vol. 6640, J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont and C. Scoglio, Eds., Berlin, Heidelberg, Springer, 2011, pp. 241-253.

[361] H. Velayos and G. Karlsson, “Limitations of range estimation in wireless LAN,” in *1st Workshop on Positioning, Navigation and Communication (WPNC'04)* , Hannover, Germany, 2004.

[362] P. Basu, N. Khan and T. Little, “A mobility based metric for clustering in mobile ad hoc networks,” in *21st International Conference on Distributed Computing Systems Workshops*, Mesa, AZ, USA, 2001.

[363] J. Yu and P. Chong, “A survey of clustering schemes for mobile ad hoc networks,” *IEEE Communications Surveys & Tutorials,* vol. 7, no. 1, pp. 32-48, 2005.

[364] I. Er and W. Seah, “Mobility-based d-hop clustering algorithm for mobile ad hoc networks,” in *IEEE Wireless Communications and Networking Conference,* Atlanta, GA, USA, 2004.

[365] R. Palit, E. Hossain and P. Thulasiraman, “MAPLE: a framework for mobility-aware pro-active low energy clustering in ad hoc mobile wireless networks,” *Wireless Communications and Mobile Computing,* vol. 6, no. 6, pp. 773-789, 2006.

[366] J. B. Y. Tsui, Fundamental of global positioning system receivers: a software approach, 2nd ed., John Wiley and Sons, 2004.

[367] G. Wang, L. Zhang and J. Cao, "A virtual circle-based clustering algorithm with mobility prediction in large-scale MANETs," in *Networking and Mobile Computing (ICCNMC 2005). Lecture Notes in Computer Science*, vol. 3619, X. Lu and W. Zha, Eds., Berlin, Heidelberg, Springer, pp. 364-374.

[368] D. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482-1493, 1991.

[369] H. S. Hassanein, H. Du and C. Yeh, "Robust route establishment in high mobility MANETS," in *International Computer Engineering Conference*, 2004.

[370] P. Brown, J. A. Estefan, K. Laskey, F. G. McCabe and D. Thornton, "OASIS reference architecture foundation for service oriented architecture version 1.0," 4 December 2012. [Online]. Available: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.pdf. [Accessed 10 March 2019].

[371] Z. Smoreda, A.-M. O. Raimond and T. Couronné, "Spatiotemporal data from mobile phones for personal mobility assessment," in *Transport Survey Methods: Best Practice for Decision Making*, J. Zmud, M. Lee-Gosselin, J. A. Carrasco and M. A. Munizaga, Eds., Emerald, 2013.

[372] S. Faye, W. Bronzi, I. Tahirou and T. Engel, "Characterizing user mobility using mobile sensing systems," *International Journal of Distributed Sensor Networks*, vol. 13, no. 8, 2017.

[373] N. E. Williams, T. A. Thomas, M. Dunbar, N. Eagle and A. Dobra, "Measures of human mobility using mobile phone records enhanced with GIS data," *PLOS ONE*, vol. 10, no. 7, p. e0133630, 2015.

[374] W. Wang and I. F. A. Yildiz, "On the estimation of user mobility pattern for location tracking in wireless networks," in *IEEE Global Telecommunications Conference (GLOBECOM '02)*, Taipei, Taiwan, 2002.

[375] W. Ma, Y. Fang and P. Lin, "Mobility management strategy based on user mobility patterns in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 1, 2007.

[376] S. Deng, L. Huang, J. Taheri, J. Yin, M. Zhou and A. Y. Zomaya, "Mobility-aware service composition in mobile communities," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 3, pp. 555-568, 2017.

[377] S. Tyagi, S. Som and Q. P. Ranab, "A reliability based variant of AODV in MANETs: proposal, analysis and comparison," *Procedia Computer Science*, vol. 79, pp. 903-911, 2016.

[378] K. K. Vadde and V. R. Syrotiuk, "Factor interaction on service delivery in mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 7, pp. 1335-1346, 2004.

[379] Z. Gao, Y. Yang, J. Zhao, J. Cui and X. Li, "Service discovery protocols for MANETs: a survey," in *Mobile Ad-hoc and Sensor Networks (MSN 2006)*, J. Cao, I. Stojmenovic, X. Jia and S. K. Das, Eds., Berlin, Heidelberg, Springer , pp. 232-243.

[380] V. Lenders, M. May and B. Plattner, "Service discovery in mobile ad hoc networks: A field theoretic approach," *Pervasive and Mobile Computing*, vol. 1, pp. 343-370, 2005.

[381] C. Chang, S. N. Srirama and S. Ling, "An adaptive mediation framework for mobile P2P social content sharing," in *Service-Oriented Computing (ICSOC 2012). Lecture Notes in Computer Science*, vol. 7636, C. Liu, H. Ludwig, F. Toumani and Q. Yu, Eds., Springer, Berlin, Heidelberg, 2012, pp. 374-388.

[382] G. Carvalho, B. Cabral, V. Pereira and J. Bernardino, "Edge computing: current trends, research challenges and future directions," *Computing*, vol. 103, pp. 993-1023, 2021.

[383] M. Larouia, B. Nour, H. Moungla, M. A. Cherif, H. Afifi and M. Guizani, "Edge and fog computing for IoT: A survey on current research activities & future directions," *Computer Communications*, vol. 180, pp. 210-231, 2021.

[384] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder and B. Koldehofe, "Mobile fog," in *2nd ACM SIGCOMM Workshop on Mobile Cloud Computing (MCC'13)*, Hong Kong, China, 2013.

[385] A. Chandra, JonWeissman and B. Heintz, "Decentralized edge clouds," *IEEE Internet Computing*, vol. 17, no. 5, pp. 70-73, 2013.

[386] A. Jonathan, M. Ryden, K. Oh, A. Chandra and JonWeissman, "Nebula: distributed edge cloud for data intensive computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3229-3242, 2017.

[387] M. Hirsch, C. Mateos, A. Zunino, T. A. Majchrzak, T.-M. Grønli and H. Kaindl, "A simulation-based performance evaluation of heuristics for dew computing," in *54th Hawaii International Conference on System Sciences*, Maui, Hawaii, 2021.

[388] M. Hirsch, C. Mateos, A. Zunino, T. Majchrzak, T. Grønli and H. Kaindl, "A task execution scheme for dew computing with state-of-the-art smartphones," *Electronics,* vol. 10, no. 16, p. 2006, 2021.

[389] X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin and D. O. Wu, "Improving cloud gaming experience through mobile edge computing," *IEEE Wireless Communications,* vol. 26, no. 4, pp. 178-183, 2019.

[390] H. Xing, L. Liu, J. Xu and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Transactions on Communications,* vol. 67, no. 6, pp. 4193-4207, 2019.

[391] Y. Pan, L. Gao, J. Luo, T. Wang and J. Luo, "A multi-dimensional resource crowdsourcing framework for mobile edge computing," in *IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 2020.

[392] F.-J. Ferrández-Pastor, H. Mora, A. Jimeno-Morenilla and B. Volckaert, "Deployment of IoT edge and fog computing technologies to develop smart building services," *Sustainability,* vol. 10, no. 11, p. 3832, 2018.

[393] A. Seitz, J. O. Johanssen, B. Bruegge, V. Loftness, V. Hartkopf and M. Sturm, "A fog architecture for decentralized decision making in smart buildings," in *2nd International Workshop on Science of Smart City Operations and Platforms Engineering (SCOPE '17)*, Pittsburgh, Pennsylvania, 2017.

[394] R. Vilalta, V. Lopez, A. Giorgetti, S. Peng, V. Orsini, L. Velasco, R. Serral-Gracià, D. Morris, S. D. Fina, F. Cugini, P. Castoldi, A. Mayoral, R. Casellas, R. Martínez, C. Verikoukis and R. Muñoz, "TelcoFog: a unified flexible fog and cloud computing architecture for 5G networks," *IEEE Communications Magazine,* vol. 55, no. 8, pp. 36-43, 2017.

[395] S. Goodwin, Smart ohme automation with Linux and Raspberry Pi, 2nd ed., Apress, 2013.

[396] M. Mahadi Abdul Jamil and M. Shukri Ahmad, "A pilot study: development of home automation system via raspberry Pi," in *2nd International Conference on Biomedical Engineering (ICoBE)*, Penang, Malaysia, 2015.

[397] M. Aftab, C. Chen, C.-K. Chau and T. Rahwan, "Automatic HVAC control with real-time occupancy recognition and simulation-guided model predictive control in low-cost embedded system," *Energy and Buildings,* vol. 154, pp. 141-156, 2017.

[398] Z. A. Ali, M. Shafiq, M. Aamir, F. M. and K. Hessan, "Heating, ventilation and air conditioning system using Raspberry Pi and interfacing touch screen," *International Research Journal of Engineering and Technology,* vol. 2, no. 1, 2015.

[399] P. Khot and N. Hulle, "Raspberry Pi based wireless sensor network in heating, ventilation and air condition application," *International Research Journal of Engineering and Technology,* vol. 4, no. 12, 2017.

[400] E. G. Swedin and D. L. Ferro, Computers: the life story of a technology, Baltimore, Maryland: Johns Hopkins University Press, 2007.

[401] I. Foster and C. Kesselman, Eds., The grid: blueprint for a new computing infrastructure, San Francisco, United States: Morgan Kaufmann Publishers, 1998.

[402] E. Brynjolfsson, P. Hofmann and J. Jordan, "Cloud computing and electricity: beyond the utility model," *Communications of the ACM,* vol. 53, no. 5, pp. 32-34, 2010.

[403] T. Korri, "Cloud computing: utility computing over the Internet," in *TKK T-110.5190 Seminar on Internetworking*, 2009.

[404] R. Buyya, "Market-oriented cloud computing: vision, hype, and reality of delivering computing as the 5th utility," in *4th ChinaGrid Annual Conference*, Yangtai, China, 2009.

[405] C. Bonnington, "In less than two years, a smartphone could be your only computer," 10 February 2015. [Online]. Available: http://www.wired.com/2015/02/smartphone-only-computer/. [Accessed 27 June 2016].

[406] StatCounter Global Stats, "Mobile and tablet internet usage exceeds desktop for first time worldwide," 1 November 2016. [Online]. Available: http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide. [Accessed 2 November 2016].

[407] Digit NewsDesk, "Turing wants to bring the future flagship smartphone by 2017," 2 September 2016. [Online]. Available: http://www.digit.in/mobile-phones/this-is-turings-vision-of-a-future-flagship-smartphone-31600.html. [Accessed 3 September 2016].

[408] NVIDIA, "The benefits of multiple CPU cores in mobile devices," NVIDIA Corporation, 2010.

[409] “ARM and QUALCOMM: enabling the next mobile computing revolution with highly integrated ARMv8-A based SoCs,” ARM/Qualcomm, 2014.

[410] C. Ziegler, “LG Optimus 2X: first dual-core smartphone launches with Android, 4-inch display, 1080p video recording,” 15 December 2010. [Online]. Available: https://www.engadget.com/2010/12/15/lg-optimus-2x-first-dual-core-smartphone-launches-with-android/. [Accessed 211 August 2022].

[411] S. Choudhury, “List of phones with Snapdragon 8 gen 1 to buy in 2022,” 6 January 2022. [Online]. Available: https://www.dealntech.com/snapdragon-898-processor-phones/. [Accessed 24 July 2022].

[412] A. Asaduzzaman, D. Gummadi and C. M. Yip, “A talented CPU-to-GPU memory mapping technique,” in *IEEE SOUTHEASTCON 2014*, Lexington, KY, 2014.

[413] C. Cullinan, C. Wyant and T. Frattesi, “Computing performance benchmarks among CPU, GPU, and FPGA,” MathWorks, 2012.

[414] J. Nickolls and W. J. Dally, “The GPU computing era,” IEEE Computer Society, 2010.

[415] N. Muralidharan, S. Wunnava and A. Noel, “The system on chip technology,” in *2nd Latin American and Caribbean Conference for Engineering and Technology (LACCEI’2004)*, Miami, Florida, 2004.

[416] S. Anthony, “SoC vs. CPU – the battle for the future of computing,” 19 April 2012. [Online]. Available: http://www.extremetech.com/computing/126235-soc-vs-cpu-the-battle-for-the-future-of-computing. [Accessed 11 August 2022].

[417] N. Rajovicxz, P. M. Carpenterx, I. Geladox, N. Puzovicx, A. Ramirezxz and M. Valero, “Supercomputing with commodity CPUs: are mobile SoCs ready for HPC?,” in *International Conference on High Performance Computing, Networking, Storage and Analysis (SC ’13)*, Denver, USA, 2013.

[418] W. Oh, “India will overtake US to become world's second largest smartphone market by 2017,” 01 July 2015. [Online]. Available: https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-press-releases/strategy-analytics-press-release/2015/07/01/India-will-overtake-US-to-become-world's-second-largest-smartphone-market-by-2017#.VuHPKPl97IX. [Accessed 11 March 2016].

[419] Cisco, “Cisco visual networking index: global mobile data traffic forecast update, 2015–2020,” Cisco, February 2016.

[420] GSMA Intelligence, “The mobile economy 2022,” GSMA, 2022.

[421] Newsroom, “Gartner says worldwide smartphone sales grew 3.9 percent in first quarter of 2016,” Gartner, 19 May 2016. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2016-05-19-gartner-says-worldwide-smartphone-sales-grew-4-percent-in-first-quarter-of-2016. [Accessed 11 August 2022].

[422] S. O'Dea, “Smartphone subscriptions worldwide 2016-2027,” 23 February 2022. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Accessed 13 July 2022].

[423] S. O'Dea, “Number of smartphone connections 2025, by country,” 29 2021 April. [Online]. Available: https://www.statista.com/statistics/982135/smartphone-connections-by-country/. [Accessed 13 July 2022].

[424] GSMA, “Smartphones to account for two thirds of world's mobile market by 2020 says new GSMA intelligence study,” 11 September 2014. [Online]. Available: https://www.gsma.com/newsroom/press-release/smartphones-account-two-thirds-worlds-mobile-market-2020/. [Accessed 17 August 2018].

[425] GSMA Newsroom, “Two-thirds of mobile connections running on 4G/5G networks by 2025, finds new GSMA study,” 26 February 2018. [Online]. Available: https://www.gsma.com/newsroom/press-release/two-thirds-mobile-connections-running-4g-5g-networks-2025-finds-new-gsma-study/. [Accessed 13 July 2022].

[426] A. Weissberger, “Development of “IMT vision for 2030 and beyond” from ITU-R WP 5D,” 15 June 2021. [Online]. Available: https://techblog.comsoc.org/2021/06/15/development-of-imt-vision-for-2030-and-beyond-from-itu-r-wp-5d/. [Accessed 13 July 2022].

[427] Orange, “Orange’s vision for 6G,” Orange, March 2022.

[428] Next G Alliance Working Groups, “National 6G roadmap,” 2022. [Online]. Available: https://nextgalliance.org/working_group/national-6g-roadmap/. [Accessed 13 July 2022].

[429] UT News, “New 6G research center unites industry leaders and UT wireless experts,” 07 July 2021. [Online]. Available: https://news.utexas.edu/2021/07/07/new-6g-research-center-unites-industry-leaders-and-ut-wireless-experts/. [Accessed 13 July 2022].

[430] Oppo, “6G AI-cube intelligent networking,” July 2021.

[431] Ericsson Press Release, “Ericsson and MIT enter into collaboration agreements to research next generation of mobile networks,” 2021 July 8. [Online]. Available: https://www.ericsson.com/en/press-releases/6/2021/7/ericsson-and-mit-enter-into-collaboration-agreements-to-research-next-generation-of-mobile-networks. [Accessed 13 July 2022].

[432] R. Heydon, Bluetooth low energy: the developer's handbook, Prentice Hall, 2012.

[433] P. K. D. Pramanik, A. Nayyar and G. Pareek, “WBAN: driving e-healthcare beyond telemedicine to remote health monitoring. Architecture and protocols,” in *Telemedicine Technologies: Big data, Deep Learning, Robotics, Mobile and Remote Applications for Global Healthcare*, D. J. Hemanth and V. E. Balas, Eds., Elsevier, 2019, pp. 89-119.

[434] D. Schneider, K. Moraes, J. M. d. Souza and M. G. P. Esteves, “CSCWD: five characters in search of crowds,” in *IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, Wuhan, China, 2012.

[435] R. Buyya and S. Venugopal, “A gentle introduction to grid computing and technologies,” *Database*, vol. 2, no. R3, 2005.

[436] B. Jacob, M. Brown, K. Fukui and N. Trivedi, Introduction to grid computing, IBM Redbooks, 2005.

[437] J. Joseph, Grid computing, Pearson Education India, 2004.

[438] C. Cerin and G. Fedak, Eds., Desktop grid computing, Chapman and Hall/CRC, 2019.

[439] Z. Constantinescu-Fuløp, “A desktop grid computing approach for scientific computing and visualization,” 2008.

[440] C. Wu, R. Buyya and K. Ramamohanarao, “Cloud pricing models: taxonomy, survey, and interdisciplinary challenges,” *ACM Computing Surveys*, vol. 52, no. 6, pp. 1-36, 2020.

[441] H. Jin, S. Ibrahim, T. Bell, W. Gao, D. Huang and S. Wu, “Cloud types and services,” in *Handbook of Cloud Computing*, B. Furht and A. Escalante, Eds., Boston, MA, Springer, 2010, pp. 335-355.

[442] Q. Zhang, L. Cheng and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *Journal of Internet Services and Applications*, vol. 1, pp. 7-18, 2010.

[443] C. S. Yeo, R. Buyya, H. Pourreza, R. Eskicioglu, P. Graham and F. Sommers, “Cluster computing: high-performance, high-availability, and high-throughput processing on a network of computers,” in *Handbook of Nature-Inspired and Innovative Computing*, A. Y. Zomaya, Ed., Boston, MA, Springer, 2006, pp. 521-551.

[444] M. Baker, R. Buyya and D. Hyde, “Cluster computing: a high-performance contender,” *Computer*, vol. 32, no. 7, pp. 79-80,83, 1999.

[445] M. Baker, “Cluster computing white paper,” *arXiv*, vol. arXiv:cs/0004014v2, 2000.

[446] A. Martínez, S. Prieto, N. Gallego, R. Nou, J. Giralt and T. Cortes, “XtreemOS-MD: grid computing from mobile devices,” in *Mobile Wireless Middleware, Operating Systems, and Applications (MOBILWARE 2010). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 48, Y. Cai, T. Magedanz, M. Li, J. Xia and C. Giannelli, Eds., Berlin, Heidelberg, Springer, 2010, pp. 45-58.

[447] C. B. Wehner, M. F. Wehner and S. A. Snow, “Mobile grid computing”. USA Patent US20100281095A1, 4 November 2010.

[448] J. Furthmüller and O. P. Waldhorst, “Survey on grid computing on mobile consumer devices,” in *Grid and Cloud Computing: Concepts, Methodologies, Tools and Applications*, IGI Global, 2012, pp. 1197-1220.

[449] T. H. Noor, S. Zeadally, A. Alfazi and Q. Z. Sheng, “Mobile cloud computing: challenges and future research directions,” *Journal of Network and Computer Applications*, vol. 115, pp. 70-85, 2018.

[450] M. Shiraz, M. Sookhak, A. Gani and S. A. A. Shah, “A study on the critical analysis of computational offloading frameworks for mobile cloud computing,” *Journal of Network and Computer Applications*, vol. 47, pp. 47-60, 2015.

[451] N. Fernando, S. W. Loke and W. Rahayu, “Mobile cloud computing: a survey,” *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84-106, 2013.

[452] S. Greengard, “Following the crowd,” *Communications of the ACM*, vol. 54, p. 20–22, 2011.

[453] M. Vukovic and C. Bartolini, “Towards a research agenda for enterprise crowdsourcing,” in *Leveraging Applications of Formal Methods, Verification, and Validation*, T. Margaria and B. Steffen, Eds., Berlin/Heidelberg, Springer, 2010, p. 425–434.

[454] R. Buettner, “A systematic literature review of crowdsourcing research from a human resource management perspective,” in *48th Annual Hawaii International Conference on System Sciences*, Kauai, Hawaii, 2015.

[455] G. Chatzimilioudis, A. Konstantinidis, C. Laoudias and D. Zeinalipour-Yazti, “Crowdsourcing with smartphones,” *IEEE Internet Computing,* vol. 16, no. 5, pp. 36-44, 2012.

[456] A. Ray, C. Chowdhury, S. Bhattacharya and S. Roy, “A survey of mobile crowdsensing and crowdsourcing strategies for smart mobile device users,” *CCF Transactions on Pervasive Computing and Interaction,* 2022.

[457] J. Phuttharak and S. W. Loke, “A review of mobile crowdsourcing architectures and challenges: toward crowd-empowered Internet-of-Things,” *IEEE Access,* vol. 7, pp. 304-324, 2018.

[458] X. Kong, X. Liu, B. Jedari, M. Li, L. Wan and F. Xia, “Mobile crowdsourcing in smart cities: technologies, applications, and future challenges,” *IEEE Internet of Things Journal,* vol. 6, no. 5, pp. 8095-8113, 2019.

[459] IBM Corporation, “Running on Android,” 2021. [Online]. Available: https://www.worldcommunitygrid.org/help/viewTopic.do?shortName=android. [Accessed 2021 July 16].

[460] D. P. Anderson, “BOINC: a platform for volunteer computing,” *Journal of Grid Computing,* vol. 18, pp. 99-122, 2020.

[461] M. Curiel, D. F. Calle, A. S. Santamaría, D. F. Suarez and L. Flórez, “Parallel processing of images in mobile devices using BOINC,” *Open Engineering,* vol. 8, no. 1, pp. 87-101, 2018.

[462] M. M. Mohamed, V. A. Srinivas and D. Janakiram, “Moset: an anonymous remote mobile cluster computing paradigm,” *Journal of Parallel and Distributed Computing,* vol. 65, no. 10, pp. 1212-1222, 2005.

[463] T. Kandappu, A. Misra, S.-F. Cheng, N. Jaiman, R. Tandriansiyah, C. Chen, H. C. Lau, D. Chander and K. Dasgupta, “Campus-scale mobile crowd-tasking: deployment & behavioral insights,” in *19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16),* San Francisco; USA, 2016.

[464] L. W. McKnight, J. Howison and S. Bradner, “Guest editors' introduction: wireless grids - distributed resource sharing by mobile, nomadic, and fixed devices,” *IEEE Internet Computing,* vol. 8, pp. 24-31, 2004.

[465] P. K. D. Pramanik, N. Sinhababu, A. Nayyar, M. Masud and P. Choudhury, “Predicting resource availability in local mobile crowd computing using convolutional GRU,” *Computers, Materials and Continua,* vol. 70, no. 3, pp. 5199-5212, 2021.

[466] P. K. D. Pramanik, G. Bandyopadhyay and P. Choudhury, “Predicting relative topological stability of mobile users in a P2P mobile cloud,” *SN Applied Sciences,* vol. 2, 2020.

[467] L. S. h. Li and E. C. Ifeachor, “Challenges of mobile ad-hoc grids and their applications in e-healthcare,” in *2nd International Conference on Computational Intelligence in Medicine and Healthcare (CIMED2005),* 2005.

[468] M. C. Dan, M. M. Gabriela, Y. Ji, B. Ladislau and H. J. Siegel, “Ad hoc grids: communication and computing in a power constrained environment,” in *IEEE International Conference on Performance, Computing, and Communications*, Phoenix, USA, 2003.

[469] K. Karra, *Wireless distributed computing on the Android platform,* Virginia Polytechnic Institute and State University, 2012.

[470] C. Storm, “Fault tolerance in distributed computing,” in *Specification and Analytical Evaluation of Heterogeneous Dynamic Quorum-Based Data Replication Schemes,* Vieweg+Teubner Verlag, 2012, pp. 13-79.

[471] F. Cristian, H. Aghili, H. R. Strong and D. Dolev, “Atomic broadcast: from simple message diffusion to Byzantine agreement,” *Information and Computation,* vol. 118, no. 1, pp. 158-179, 1995.

[472] F. Cristian, “Understanding fault-tolerant distributed systems,” *Communications of the ACM,* vol. 34, no. 2, pp. 56-78, 1991.

[473] A. Sari and M. Akkaya, “Fault tolerance mechanisms in distributed systems,” *International Journal of Communications, Network and System Sciences,* vol. 8, no. 12, pp. 471-482, 2015.

[474] F. C. Gärtner, “Fundamentals of fault-tolerant distributed computing in asynchronous environments,” *ACM Computing Surveys,* vol. 31, no. 1, p. 1–26, 1999.

[475] D. Poola, M. A. Salehi, K. Ramamohanarao and R. Buyya, “A taxonomy and survey of fault-tolerant workflow management systems in cloud and distributed computing environments,” in *Software Architecture for Big Data and the Cloud*, I. Mistrik, R. Bahsoon, N. Ali, M. Heisel and B. Maxim, Eds., Morgan Kaufmann, 2017, pp. 285-320.

[476] E. N. Elnozahy, L. Alvisi, Y.-M. Wang and D. B. Johnson, “A survey of rollback-recovery protocols in message-passing systems,” *ACM Computing Surveys,* vol. 34, no. 3, pp. 375-408, 2002.

[477] L. Alvisi and K. Marzullo, “Message logging: pessimistic, optimistic, and causal,” in *15th International Conference on Distributed Computing, Systems (ICDCS 1995)*, Vancouver, 1995.

[478] T. Mengistu, A. Alahmadi, A. Albuali, Y. Alsenani and D. Che, “A "no data center" solution to cloud computing,” in *IEEE 10th International Conference on Cloud Computing (CLOUD)*, Honololu, USA, 2017.

[479] B. Moyer, “Is crowd computing the next big thing?,” 25 November 2019. [Online]. Available: https://www.eejournal.com/article/is-crowd-computing-the-next-big-thing/. [Accessed 12 July 2022].

[480] P. K. D. Pramanik, S. Biswas, S. Pal, D. Marinković and P. Choudhury, “A comparative analysis of multi-criteria decision-making methods for resource selection in mobile crowd computing,” *Symmetry,* vol. 13, no. 9, p. 1713, 2021.

[481] P. K. D. Pramanik, N. Sinhababu, A. Nayyar and P. Choudhury, “Predicting device availability in mobile crowd computing using ConvLSTM,” in *7th International Conference on Optimization and Applications (ICOA)*, Wolfenbüttel, Germany, 2021.

[482] D. Fu and Y. Liu, “Fairness of task allocation in crowdsourcing workflows,” *Mathematical Problems in Engineering,* vol. 2021 (Article ID 5570192), 2021.

[483] F. Basık, B. Gedik, H. Ferhatosmanoğlu and K.-L. Wu, “Fair task allocation in crowdsourced delivery,” *IEEE Transactions on Services Computing,* vol. 14, no. 4, pp. 1040-1053, 2021.

[484] V. Kravtsov, D. Carmeli, W. Dubitzky, A. Orda, A. Schuster, M. Silberstein and B. Yoshpa, “Quasi-opportunistic supercomputing in grid environments,” in *8th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2008)*, Cyprus, 2008.

[485] A. Dogac, E. Gokkoca, S. Arpinar, P. Koksal, I. Cingil, B. Arpinar, N. Tatbul, P. Karagoz, U. Halici and M. Altinel, “Design and implementation of a distributed workflow management system: METUFlow,” in *Workflow Management Systems and Interoperability. NATO ASI Series*, vol. 164, A. Doğaç, L. Kalinichenko, M. T. Özsu and A. Sheth, Eds., Berlin, Heidelberg, Springer, 1998, pp. 61-91.

[486] L. Wang, W. Jie and H. Zhu, “State-of-arts: workflow management for grid computing,” in *Grid Technologies: Emerging from Distributed Architectures to Virtual Organizations*, Southampton, UK, WIT Press, 2006, pp. 241-270.

[487] J. Yu and R. Buyya, “A taxonomy of scientific workflow systems for grid computing,” *ACM SIGMOD Record,* vol. 34, no. 3, pp. 44-49, 2005.

[488] G. Alonso, R. Günthör, M. Kamath, D. Agrawal, A. El Abbadi and C. Mohan, “Exotica/FMDC: a workflow management system for mobile and disconnected clients,” in *Databases and Mobile Computing*, D. Barbara, R. Jain and N. Krishnakumar, Eds., Boston, Springer, 1996, pp. 27-45.

[489] F. Tang, M. Guo, M. Dong, M. Li and H. Guan, “Towards context-aware workflow management for ubiquitous computing,” in *International Conference on Embedded Software and Systems*, Chengdu, China, 2008.

[490] S. Tarkoma, M. Siekkinen, E. Lagerspetz and Y. Xiao, “Overview,” in *Smartphone Energy Consumption: Modeling and Optimization*, Cambridge, Cambridge University Press, 2014, pp. 227-233.

[491] S. A. Gordon, “8 things you need to know about Nvidia's groundbreaking Tegra X1 mobile super chip,” 05 January 2015. [Online]. Available: https://www.androidpit.com/nvidia-tegra-x1. [Accessed 04 March 2016].

[492] J. Yu, E. Williams and M. Ju, “Analysis of material and energy consumption of mobile phones in China,” *Energy Policy,* vol. 38, no. 8, p. 4135–4141, August 2010.

[493] P. K. D. Pramanik, S. Pal and P. Choudhury, "Green and sustainable high-performance computing with smartphone crowd computing: benefits, enablers, and challenges," *Scalable Computing: Practice and Experience,* vol. 20, no. 2, pp. 259-283, 2019.

[494] P. K. D. Pramanik, N. Sinhababu, B. Mukherjee, S. Padmanaban, A. Maity, B. K. Upadhyaya, J. B. Holm-Nielsen and P. Choudhury, "Power consumption analysis, measurement, management, and issues: a state-of-the-art review on smartphone battery and energy usage," *IEEE Access,* vol. 7, no. 1, pp. 182113-182172, 2019.

[495] Q. Zhang, Z. Xu and B. Lu, "Strongly coupled MoS2–3D graphene materials for ultrafast charge slow discharge LIBs and water splitting applications," *Energy Storage Materials,* vol. 4, pp. 84-91, 2016.

[496] W. Zou, F.-J. Xia, J.-P. Song, L. Wu, L.-D. Chen, H. Chen, Y. Liu, W.-D. Dong, S.-J. Wu, Z.-Y. Hu, J. Liu, H.-E. Wang, L.-H. Chen, Y. Li, D.-L. Peng and B.-L. Su, "Probing and suppressing voltage fade of Li-rich Li1.2Ni0.13Co0.13Mn0.54O2 cathode material for lithium-ion battery," *Electrochimica Acta,* vol. 318, pp. 875-882, 2019.

[497] P. Wu, G. Shao, C. Guo, Y. Lu, X. Dong, Y. Zhong and A. Liu, "Long cycle life, low self-discharge carbon anode for Li-ion batteries with pores and dual-doping," *Journal of Alloys and Compounds,* vol. 802, pp. 620-627, 2019.

[498] L. L. Perreault, F. Colò, G. Meligrana, K. Kim, S. Fiorilli, B. Federico, R. N. Jijeesh, V.-B. Chiara, F. Justyna, K. Freddy and G. Claudio, "Spray-dried mesoporous mixed Cu-Ni Oxide@Graphene nanocomposite microspheres for high power and durable Li-ion battery anodes," *Advanced Energy Materials,* vol. 8, no. 35, p. 1802438, 2018.

[499] S. W. Lee, N. Yabuuchi, B. M. Gallant, S. Chen, B. S. Kim, P. T. Hammond and Y. Shao-Horn, "High-power lithium batteries from functionalized carbon-nanotube electrodes," *Nature nanotechnology,* vol. 5, no. 7, p. 531, 2010.

[500] F. B. Spingler, W. Wittmann, J. Sturm, B. Rieger and A. Jossen, "Optimum fast charging of lithium-ion pouch cells based on local volume expansion criteria," *Journal of Power Sources,* vol. 393, pp. 152-160, 2018.

[501] V. H. Pham, J. A. Boscoboinik, D. J. Stacchiola, E. C. Self, P. Manikandan, S. Nagarajan, Y. Wang, V. G. Pol, J. Nanda, E. Paek and D. Mitlin, "Selenium-sulfur (SeS) fast charging cathode for sodium and lithium metal batteries," *Energy Storage Materials,* vol. 20, pp. 71-79, 2019.

[502] J. Zheng, M. H. Engelhard, D. Mei, S. Jiao, B. J. Polzin, J.-G. Zhang and W. Xu, "Electrolyte additive enabled fast charging and stable cycling lithium metal batteries," *Nature Energy,* vol. 2, 2017.

[503] Y. Gao, Z. Yan, J. L. Gray, X. He, D. Wang, T. Chen, Q. Huang, Y. C. Li, H. Wang, S. H. Kim, T. E. Mallouk and D. Wang, "Polymer–inorganic solid–electrolyte interphase for stable lithium metal batteries under lean electrolyte conditions," *Nature Materials,* vol. 18, pp. 384-389, 2019.

[504] W. Xing, "High energy/power density, safe Lithium battery with nonflammable electrolyte," *ECS Transactions,* vol. 85, no. 13, pp. 109-114, 2018.

[505] X. Fan, E. Hu, X. Ji, Y. Zhu, F. Han, S. Hwang, J. Liu, S. Bak, Z. Ma, T. Gao, S.-C. Liou, J. Bai, X.-Q. Yang, Y. Mo, K. Xu, D. Su and C. Wang, "High energy-density and reversibility of iron fluoride cathode enabled via an intercalation-extrusion reaction," *Nature Communications,* vol. 9 (Article number: 2324), 2018.

[506] A. R. Mainar, L. C. Colmenares, H.-J. Grande and J. A. Blázquez, "Enhancing the cycle life of a Zinc–air battery by means of electrolyte additives and Zinc surface protection," *Batteries,* vol. 4, no. 3, p. 46, 2018.

[507] L. Edwards, "Nanowire battery can extend your phone battery life by hundreds of thousands of times," 21 April 2016. [Online]. Available: https://www.pocket-lint.com/gadgets/news/137387-nanowire-battery-can-extend-your-phone-battery-life-by-hundreds-of-thousands-of-times. [Accessed 17 July 2019].

[508] F.-G. Efrén, G. Espinosa-Medina, D. d. L.-Z. Ramón, A. D. d. l. Rosa-Zapata and J. V. González-Fernández, "Analysis and design of a simple wireless charger for mobile phones," in *IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC),* Ixtapa, Mexico, 2019.

[509] X. Lu, P. Wang, D. Niyato, D. I. Kim and Z. Han, "Wireless charging technologies: fundamentals, standards, and network applications," *IEEE Communications Surveys & Tutorials,* vol. 18, no. 2, pp. 1413-1452, 2016.

[510] N. Ahuja, M. M. Eshaghian-Wilner, Z. Ge, R. Liu, A. S. N. Pati, K. Ravicz, M. Schlesinger, S. H. Wu and K. Xie, "Wireless power for implantable devices: a technical review," in *Wireless Computing in Medicine: From Nano to Cloud with Its Ethical and Legal Implications*, M. M. Eshaghian-Wilner, Ed., Wiley, 2016, pp. 187-209.

[511] O. A. Saraereh, A. Alsaraira, I. Khan and B. J. Choi, "A hybrid energy harvesting design for on-body Internet-of-Things (IoT) networks," *Sensors,* vol. 20, no. 2, p. 407, 2020.

[512] X. Fan, J. Chen, J. Yang, P. Bai, Z. Li and Z. L. Wang, "Ultrathin, rollable, paper-based triboelectric nanogenerator for acoustic energy harvesting and self-powered sound recording," *ACS Nano,* vol. 9, no. 4, pp. 4236-4243, 2015.

[513] N. Jain, X. Fan, W. D. Leon-Salas and A. M. Lucietto, "Extending battery life of smartphones by overcoming idle power consumption using ambient light energy harvesting," in *IEEE International Conference on Industrial Technology (ICIT)*, Lyon, France, 2018.

[514] E. Bulut, M. E. Ahsen and B. K. Szymanski, "Opportunistic wireless charging for mobile social and sensor networks," in *IEEE Globecom Workshops (GC Wkshps)*, Austin, USA, 2014.

[515] S. Nikoletseas, T. P. Raptis and C. Raptopoulos, "Wireless charging for weighted energy balance in populations of mobile peers," *Ad Hoc Networks,* vol. 60, pp. 1-10, 2017.

[516] E. Bulut, S. Hernandez, A. Dhungana and B. K. Szymanski, "Is crowdcharging possible?," in *27th International Conference on Computer Communication and Networks (ICCCN)*, Hangzhou, China, 2018.

[517] D. Luis, "Tech war: Nvidia Tegra X1 takes on Snapdragon 810 with raw GPU power," 15 January 2015. [Online]. Available: http://www.phonearena.com/news/Tech-war-Nvidia-Tegra-X1-takes-on-Snapdragon-810-with-raw-GPU-power_id64748. [Accessed 11 August 2022].

[518] T. Ahmed, M. Bhouri, D. Groulx and M. A. White, "Passive thermal management of tablet PCs using phase change materials: intermittent operation," *Applied Sciences,* vol. 9, no. 5, p. 902, 2019.

[519] Y. Tomizawa, K. Sasaki, A. Kuroda, R. Takeda and Y. Kaito, "Experimental and numerical study on phase change material (PCM) for thermal management of mobile devices," *Applied Thermal Engineering,* vol. 98, p. 320–329, 2016.

[520] C. Wang, L. Hua, H. Yan, B. Li, Y. Tu and R. Wang, "A thermal management strategy for electronic devices based on moisture sorption-desorption processes," *Joule,* vol. 4, no. 2, pp. 435-447, 2020.

[521] A. K. Singh, S. Dey, K. McDonald-Maier, K. R. Basireddy, G. V. Merrett and B. M. Al-Hashimi, "Dynamic energy and thermal management of multi-core mobile platforms: a survey," *IEEE Design & Test,* vol. 37, no. 5, pp. 25-33, 2020.

[522] Y. G. Kim, M. Kim, J. Kong and S. W. Chung, "An adaptive thermal management framework for heterogeneous multi-core processors," *IEEE Transactions on Computers,* vol. 69, no. 6, pp. 894-906, 2020.

[523] S. Chetoui and S. Reda, "Coordinated self-tuning thermal management controller for mobile devices," *IEEE Design & Test,* vol. 37, no. 5, pp. 34-41, 2020.

[524] A. Iranfar, F. Terraneo, G. Csordas, M. Zapater, W. Fornaciari and D. Atienza, "Dynamic thermal management with proactive fan speed control through reinforcement learning," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, 2020.

[525] J. Park, S. Lee and H. Cha, "App-oriented thermal management of mobile devices," in *International Symposium on Low Power Electronics and Design (ISLPED '18)*, Seattle, USA, 2018.

[526] M. Hao, J. Li, S. Park, S. Moura and C. Dames, "Efficient thermal management of Li-ion batteries with a passive interfacial thermal regulator based on a shape memory alloy," *Nature Energy,* vol. 3, p. 899–906, 2018.

[527] X. Feng, D. Ren, X. He and M. Ouyang, "Mitigating thermal runaway of Lithium-ion batteries," *Joule,* vol. 4, no. 4, pp. 743-770, 2020.

[528] K. Abinav, P. Palani Rajeshwar, J. S. Punnoose, J. Daniel and M. Sreekanth, "Heat transfer enhancement in a smart phone," *International Journal of Engineering Research and Application,* vol. 7, no. 4 (Part-5), pp. 12-23, 2017.

[529] Y. Wang, D. Zhu, Y. Yang, K. Lee, R. Mishra, G. Go, S.-H. Oh, D.-H. Kim, K. Cai, E. Liu, S. D. Pollard, S. Shi, J. Lee, K. L. Teo, Y. Wu, K.-J. Lee and H. Yang, "Magnetization switching by magnon-mediated spin torque through an antiferromagnetic insulator," *Science,* vol. 366, no. 6469, p. 1125, 2019.

[530] J. Rogerson, “An unlikely name is going to stop your phone overheating,” 17 March 2015. [Online]. Available: http://www.techradar.com/news/phone-and-communications/mobile-phones/an-unlikely-name-is-going-to-stop-your-phone-overheating-1288525. [Accessed 26 February 2016].

[531] C. Pei, Z. Wang, Y. Zhao, Z. Wang, Y. Meng, D. Pei, Y. Peng, W. Tang and X. Qu, “Why it takes so long to connect to a WiFi access point?,” in *IEEE Conference on Computer Communications (IEEE INFOCOM)*, Atlanta, USA, 2017.

[532] LinkLabs, “WiFi's future: examining 802.11ad, 802.11ah HaLow (& others),” 1 February 2018. [Online]. Available: https://www.link-labs.com/blog/future-of-wifi-802-11ah-802-11ad. [Accessed 11 August 2022].

[533] Y. Heisler, “Future iPhones may contain Li-Fi, a technology with transfer speeds 100x faster than Wi-Fi,” 18 January 2016. [Online]. Available: http://bgr.com/2016/01/18/iphone-li-fi-ios-wireless-data-transfer-speeds/. [Accessed 22 May 2016].

[534] B. Crew, “Li-Fi has just been tested in the real world, and it's 100 times faster than Wi-Fi,” 24 November 2015. [Online]. Available: http://www.sciencealert.com/li-fi-tested-in-the-real-world-for-the-first-time-is-100-times-faster-than-wi-fi. [Accessed 22 May 2016].

[535] K. Yang, K. Zhang, J. Ren and X. Shen, “Security and privacy in mobile crowdsourcing networks: challenges and opportunities,” *IEEE Communications Magazine,* vol. 53, no. 8, pp. 75-81, 2015.

[536] W. Feng, Z. Yan, H. Zhang, K. Zeng, Y. Xiao and Y. T. Hou, “A survey on security, privacy, and trust in mobile crowdsourcing,” *IEEE Internet of Things Journal,* vol. 5, no. 4, pp. 2971-2992, 2018.

[537] Y. Ma, Y. Sun, Y. Lei, N. Qin and J. Lu, “A survey of blockchain technology on security, privacy, and trust in crowdsourcing services,” *World Wide Web,* vol. 23, pp. 393-419(, 2020.

[538] M. Allahbakhsh, A. Ignjatovic, B. Benatallah, S.-M.-R. Beheshti, E. Bertino and N. Foo, “Reputation management in crowdsourcing systems,” in *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, Pittsburgh, USA, 2012.

[539] D. G. Padmavathi and M. D. Shanmugapriya, “A survey of attacks, security mechanisms and challenges in wireless sensor networks,” *International Journal of Computer Science and Information Security,* vol. 4, no. 1 & 2, 2009.

[540] M. Kaur and M. M. Bansal, “A survey on security and privacy challenges in mobile grid computing,” *International Journal of Advances in Cloud Computing and Computer Science,* vol. 1, no. 2, pp. 20-26, 2015.

[541] K. Watanabe, M. Fukushi and M. Kameyama, “Adaptive group-based job scheduling for high performance and reliable volunteer computing,” *Journal of Information Processing,* vol. 19, pp. 39-51, 2011.

[542] L. F. G. Sarmenta, “Volunteer computing,” PhD Thesis, Massachusetts Institute of Technology, 2001.

[543] K. Watanabe and M. Fukushi, “Generalized spot-checking for sabotage-tolerance in volunteer computing systems,” in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, Melbourne, Australia, 2010.

[544] K. Watanabe, M. Fukushi and S. Horiguchi, “Expected-credibility-based job scheduling for reliable volunteer computing,” *IEICE Transactions on Information and Systems,* vol. E93.D, no. 2, pp. 306-314, 2010.

[545] L. Buttyan and J.-P. Hubaux, “Enforcing service availability in mobile ad-hoc WANs,” in *First Annual Workshop on Mobile and Ad Hoc Networking and Computing (MobiHOC)*, Boston, USA, 2010.

[546] I. Bibi, A. Akhunzada, J. Malik, M. K. Khan and M. Dawood, “Secure distributed mobile volunteer computing with Android,” *ACM Transactions on Internet Technology,* vol. 22, no. 1, pp. 1-21, 2022.

[547] S. Rasool, M. Iqbal, T. Dagiuklas, Z. Ul-Qayyum and S. Li, “Reliable data analysis through blockchain based crowdsourcing in mobile ad-hoc cloud,” *Mobile Networks and Applications,* vol. 25, pp. 153-163, 2019.

[548] W. Feng and Z. Yan, “MCS-chain: decentralized and trustworthy mobile crowdsourcing based on blockchain,” *Future Generation Computer Systems,* vol. 95, pp. 649-666, 2019.

[549] J. Zhang, W. Cui, J. Ma and C. Yang, “Blockchain-based secure and fair crowdsourcing scheme,” *International Journal of Distributed Sensor Networks,* vol. 15, no. 7, 2019.

[550] Y. Lu, Q. Tang and G. Wang, "ZebraLancer: private and anonymous crowdsourcing system atop open blockchain," in *IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Austria, 2018.

[551] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang and R. H. Deng, "CrowdBC: a blockchain-based decentralized framework for crowdsourcing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 30, no. 6, pp. 1251-1266, 2019.

[552] S. Seebacher and R. Schüritz, "Blockchain technology as an enabler of service systems: a structured literature review," in *International Conference on Exploring Services Science*, Italy, 2017.

[553] E. Bellini, Y. Iraqi and E. Damiani, "Blockchain-based distributed trust and reputation management systems: a survey," *IEEE Access,* vol. 8, pp. 21127-21151, 2020.

[554] C. Huang, Z. Wang, H. Chen, Q. Hu, Q. Zhang, W. Wang and X. Guan, "RepChain: a reputation based secure, fast and high incentive blockchain system via sharding," *IEEE Internet of Things Journal,* vol. 8, no. 6, pp. 4291-4304, 7 February 2021.

[555] A. Shahid, U. Sarfraz, M. W. Malik, M. S. Iftikhar, A. Jamal and N. Javaid, "Blockchain-based reputation system in agri-food supply chain," in *Advanced Information Networking and Applications (AINA 2020). Advances in Intelligent Systems and Computing*, vol. 1151, L. Barolli, F. Amato, F. Moscato, T. Enokido and M. Takizawa, Eds., Springer, Cham, 2020, pp. 12-21.

[556] X. Zhang, G. Xue, R. Yu, D. Yang and J. Tang, "Countermeasures against false-name attacks on truthful incentive mechanisms for crowdsourcing," *IEEE Journal on Selected Areas in Communications,* vol. 35, no. 2, pp. 478-485, 2017.

[557] K. Wang, X. Qi, L. Shu, D.-j. Deng and J. J. P. C. Rodrigues, "Toward trustworthy crowdsourcing in the social internet of things," *IEEE Wireless Communications,* vol. 23, no. 5, pp. 30-36, 2016.

[558] G. A. K. Kamhoua, "Mitigating colluding attacks in online social networks and crowdsourcing platforms," PhD Thesis, Florida International University, 2019.

[559] Q. Yang, T. Wang, W. Zhang, B. Yang, Y. Yu, H. Li, J. Wang and Z. Qiao, "PrivCrowd: a secure blockchain-based crowdsourcing framework with fine-grained worker selection," *Wireless Communications and Mobile Computing,* vol. 2021 (Article ID 3758782), 2021.

[560] Y. Gong, L. Wei, Y. Guo, C. Zhang and Y. Fang, "Optimal task recommendation for mobile crowdsourcing with privacy control," *IEEE Internet of Things Journal,* vol. 3, no. 5, pp. 745-756, 2016.

[561] B. Zhao, S. Tang, X. Liu, X. Zhang and W.-N. Chen, "iTAM: bilateral privacy-preserving task assignment for mobile crowdsensing," *IEEE Transactions on Mobile Computing,* vol. 20, no. 12, pp. 3351-3366, 2021.

[562] J. Shu, X. Jia, K. Yang and H. Wang, "Privacy-preserving task recommendation services for crowdsourcing," *IEEE Transactions on Services Computing,* vol. 14, no. 1, pp. 235-247, 2021.

[563] Z. Chi, Y. Wang, Y. Huang and X. Tong, "The novel location privacy-preserving CKD for mobile crowdsourcing systems," *IEEE Access,* vol. 6, pp. 5678-5687, 2017.

[564] L. Meftah, R. Rouvoy and I. Chrisment, "Empowering mobile crowdsourcing apps with user privacy control," *Journal of Parallel and Distributed Computing,* vol. 147, pp. 1-15, 2021.

[565] G. Qiu, Y. Shen, K. Cheng, L. Liu and S. Zeng, "Mobility-aware privacy-preserving mobile crowdsourcing," *Sensors,* vol. 21, no. 7, p. 2474, 2021.

[566] Y. Xu, H. Liu and C. Yan, "A privacy-preserving exception handling approach for dynamic mobile crowdsourcing applications," *EURASIP Journal on Wireless Communications and Networking,* Vols. 2019, Article number: 113, 2019.

[567] S. Zhu, H. Hu, Y. Li and W. Li, "Hybrid blockchain design for privacy preserving crowdsourcing platform," in *IEEE International Conference on Blockchain*, Atlanta, USA, 2019.

[568] J. Wang, G. Sun, Y. Gu and K. Liu, "ConGradetect: blockchain-based detection of code and identity privacy vulnerabilities in crowdsourcing," *Journal of Systems Architecture,* vol. 114, p. 101910, 2020.

[569] C. Lin, D. He, S. Zeadally, N. Kumar and K.-K. R. Choo, "SecBCS: a secure and privacy-preserving blockchain-based crowdsourcing system," *Science China Information Sciences,* vol. 63 (Article number: 130102), 2020.

[570] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi and W. Dou, "A blockchain-powered crowdsourcing method with privacy preservation in mobile environment," *IEEE Transactions on Computational Social Systems,* vol. 6, no. 6, pp. 1407-1419, 2019.

[571] J. Shu and X. Jia, "Secure task recommendation in crowdsourcing," in *IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, 2016.

[572] H. Qin, Y. Zhang and B. Li, "Truthful mechanism for crowdsourcing task assignment," in *IEEE 10th International Conference on Cloud Computing (CLOUD)*, Honololu, USA, 2017.

[573] Y. Gao, X. Li, J. Li and Y. Gao, "A dynamic-trust-based recruitment framework for mobile crowd sensing," in *IEEE International Conference on Communications (ICC)*, Paris, France, 2017.

[574] A. Khanfor, A. Hamrouni, H. Ghazzai, Y. Yang and Y. Massoud, "A trustworthy recruitment process for spatial mobile crowdsourcing in large-scale social IoT," in *IEEE Technology & Engineering Management Conference (TEMSCON)*, Novi, USA, 2020.

[575] T. Halabi and M. Zulkernine, "Reliability-driven task assignment in vehicular crowdsourcing: a matching game," in *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Portland, USA, 2019.

[576] H. Wu, B. Düdder, L. Wang, S. Sun and G. Xue, "Blockchain-based reliable and privacy-aware crowdsourcing with truth and fairness assurance," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3586-3598, 2022.

[577] M. Bahutair, A. Bougeuttaya and A. G. Neiat, "Adaptive trust: usage-based trust in crowdsourced IoT services," in *IEEE International Conference on Web Services (ICWS)*, Milan, Italy, 2019.

[578] M. Bahutair, A. Bouguettaya and A. G. Neiat, "Just-in-time memoryless trust for crowdsourced IoT services," in *IEEE International Conference on Web Services (ICWS)*, Beijing, China, 2020.

[579] M. Bahutair, A. Bouguettaya and A. G. Neiat, "Multi-perspective trust management framework for crowdsourced IoT services," *IEEE Transactions on Services Computing,* vol. 15, no. 4, pp. 2396-2409, 2022.

[580] K. Liu, W. Chen and Z. Zhang, "Blockchain-empowered decentralized framework for secure and efficient software crowdsourcing," in *IEEE World Congress on Services (SERVICES)*, Beijing, China, 2020.

[581] W. Feng, Z. Yan, L. T. Yang and Q. Zheng, "Anonymous authentication on trust in blockchain-based mobile crowdsourcing," *IEEE Internet of Things Journal,* vol. 9, no. 16, pp. 14185-14202, 2022.

[582] C. Li, X. Qu and Y. Guo, "TFCrowd: a blockchain-based crowdsourcing framework with enhanced trustworthiness and fairness," *EURASIP Journal on Wireless Communications and Networking,* vol. 2021, no. 1, 2021.

[583] L. Tan, H. Xiao, X. Shang, Y. Wang, F. Ding and W. Li, "A blockchain-based trusted service mechanism for crowdsourcing system," in *IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, Antwerp, Belgium, 2020.

[584] X. Zhu, Y. Li, L. Fang and P. Chen, "An improved proof-of-trust consensus algorithm for credible crowdsourcing blockchain services," *IEEE Access,* vol. 8, pp. 102177-102187, 2020.

[585] Y. Sun and N. Zhang, "A resource-sharing model based on a repeated game in fog computing," *Saudi Journal of Biological Sciences,* vol. 24, no. 3, pp. 687-694, 2017.

[586] L. Islam, S. T. Alvi, M. N. Uddin and M. Rahman, "Obstacles of mobile crowdsourcing: a survey," in *IEEE Pune Section International Conference (PuneCon)*, Pune, India, 2019.

[587] "Volunteer computing," BOINC, 2018. [Online]. Available: https://boinc.berkeley.edu/trac/wiki/VolunteerComputing. [Accessed 10 August 2022].

[588] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing and X. Mao, "Incentives for mobile crowd sensing: a survey," *IEEE Communications Surveys & Tutorials,* vol. 18, no. 1, pp. 54-67, 2016.

[589] C. Muldoon, M. J. O'Grady and G. M. P. O'Hare, "A survey of incentive engineering for crowdsourcing," *The Knowledge Engineering Review,* vol. 33, p. E2, 2018.

[590] distributed.net, "What kinds of problems are well-suited for distributed computing?," [Online]. Available: http://faq.distributed.net/cache/280.html. [Accessed 10 August 2022].

[591] C. Hu, M. Xiao, L. Huang and G. Gao, "Truthful incentive mechanism for vehicle-based nondeterministic crowdsensing," in *IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, Beijing, China, 2016.

[592] Z. Ju, C. Huang, Y. Chen and L. Ma, "A truthful auction mechanism for resource provisioning in mobile crowdsensing," in *IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, San Diego, USA, 2017.

[593] Y. Fan, H. Sun and X. Liu, "Truthful incentive mechanisms for dynamic and heterogeneous tasks in mobile crowdsourcing," in *IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, Vietri sul Mare, Italy, 2015.

[594] C. Huang, H. Yu, R. A. Berry and J. Huang, "Using truth detection to incentivize workers in mobile crowdsourcing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 2257-2270, 2022.

[595] Q. Li, H. Cao, S. Wang and X. Zhao, "A reputation-based multi-user task selection incentive mechanism for crowdsensing," *IEEE Access*, vol. 8, pp. 74887-74900, 2020.

[596] J. Sun, F. Hou and S. Ma, "Reputation-aware incentive mechanism for participatory sensing," in *IEEE/CIC International Conference on Communications in China (ICCC)*, Shenzhen, China, 2015.

[597] X. Ma, J. Ma, H. Li, Q. Jiang and S. Gao, "RTRC: a reputation-based incentive game model for trustworthy crowdsourcing service," *China Communications*, vol. 13, no. 12, pp. 199-215, 2016.

[598] L.-Y. Jiang, F. He, Y. Wang, L.-J. Sun and H.-p. Huang, "Quality-aware incentive mechanism for mobile crowd sensing," *Journal of Sensors*, vol. 2017 (Article ID 5757125), 2017.

[599] D. Peng, F. Wu and G. Chen, "Pay as how well you do: a quality based incentive mechanism for crowdsensing," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '15)*, Hangzhou, China, 2015.

[600] J. Wang, J. Tang, D. Yang, E. Wang and G. Xue, "Quality-aware and fine-grained incentive mechanisms for mobile crowdsensing," in *IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, Nara, Japan, 2016.

[601] Y. Ueyama, M. Tamai, Y. Arakawa and K. Yasumoto, "Gamification-based incentive mechanism for participatory sensing," in *IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, Budapest, Hungary, 2014.

[602] L. Pang, G. Li, X. Yao and Y. Lai, "An incentive mechanism based on a Bayesian game for spatial crowdsourcing," *IEEE Access*, vol. 7, pp. 14340-14352, 2019.

[603] Q. Zhang, Q. Zhang, X. Liu, J. Dai and X. Zhang, "The evolutionary game analysis of incentive mechanism for crowd sensing of public environment," *Journal of Physics: Conference Series*, vol. 1187, no. 5, 2019.

[604] S. Luo, Y. Sun, Y. Ji and D. Zhao, "Stackelberg game based incentive mechanisms for multiple collaborative tasks in mobile crowdsourcing," *Mobile Networks and Applications*, vol. 21, pp. 506-522, 2016.

[605] X. Yang, J. Zhang, J. Peng and L. Lei, "Incentive mechanism based on Stackelberg game under reputation constraint for mobile crowdsensing," *International Journal of Distributed Sensor Networks*, vol. 17, no. 6, 2021.

[606] D. Yang, G. Xue, X. Fang and J. Tang, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing," in *18th annual international conference on Mobile computing and networking (Mobicom '12)*, Istanbul, Turkey, 2012.

[607] Q. Ma, L. Gao, Y.-F. Liu and J. Huang, "A contract-based incentive mechanism for crowdsourced wireless community networks," in *14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, Tempe, USA, 2016.

[608] N. Zhao, M. Fan, C. Tian and P. Fan, "Contract-based incentive mechanism for mobile crowdsourcing networks," *Algorithms*, vol. 10, no. 3, p. 104, 2017.

[609] Y. Zhang, C. Jiang, L. Song, M. Pan, Z. Dawy and Z. Han, "Incentive mechanism for mobile crowdsourcing using an optimized tournament model," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 4, pp. 880-892, 2017.

[610] Y. Zhang, Y. Gu, L. Song, M. Pan, Z. Dawy and Z. Han, "Tournament based incentive mechanism designs for mobile crowdsourcing," in *IEEE Global Communications Conference (GLOBECOM)*, San Diego, USA, 2015.

[611] D. Yang, G. Xue, X. Fang and J. Tang, "Incentive mechanisms for crowdsensing: crowdsourcing with smartphones," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1732-1744, 2016.

[612] Y. Chen, H. Chen, S. Yang, X. Gao, Y. Guo and F. Wu, "Designing incentive mechanisms for mobile crowdsensing with intermediaries," *Wireless Communications and Mobile Computing*, vol. 2019 (Article ID 8603526), 2019.

[613] H. Zhang, B. Liu, H. Susanto and G. Xue, "Auction-based incentive mechanisms for dynamic mobile ad-hoc crowd service," *arXiv*, vol. 1503.06819v1 [cs.NI], 2015.

[614] Y. Liu, H. Li, G. Zhao and J. Duan, "A reverse auction based incentive mechanism for mobile crowdsensing," in *IEEE International Conference on Communications (ICC)*, Kansas City, USA, 2018.

[615] H. Jin, L. Su, D. Chen, K. Nahrstedt and J. Xu, "Quality of information aware incentive mechanisms for mobile crowd sensing systems," in *16th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '15)*, Hangzhou, China, 2015.

[616] T. Zhou, B. Jia and W. Li, "A reverse auction incentive mechanism based on the participant's behavior in crowdsensing," in *Security and Privacy in New Computing Environments (SPNCE 2019). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 284, J. Li, Z. Liu and H. Peng, Eds., Springer, Cham, 2019, pp. 637-646.

[617] G. Yang, S. He, Z. Shi and J. Chen, "Promoting cooperation by the social incentive mechanism in mobile crowdsensing," *IEEE Communications Magazine,* vol. 55, no. 3, pp. 86-92, 2017.

[618] L. G. Jaimes, I. Vergara-Laurens and A. Chaker, "SPREAD, a crowd sensing incentive mechanism to acquire better representative samples," in *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, Budapest, Hungary, 2014.

[619] L. G. Jaimes, A. Chakeri, J. Lopez and A. Raij, "A cooperative incentive mechanism for recurrent crowd sensing," in *SoutheastCon*, Fort Lauderdale, USA, 2015.

[620] R. F. E. Khatib, N. Zorba and H. S. Hassanein, "A fair reputation-based incentive mechanism for cooperative crowd sensing," in *IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, 2018.

[621] Y. Tong, L. Wang, Z. Zhou, L. Chen, B. Du and J. Ye, "Dynamic pricing in spatial crowdsourcing: a matching-based approach," in *International Conference on Management of Data (SIGMOD '18)*, Houston, USA, 2018.

[622] H. Wang, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz and Q. Cheng, "Real-time crowdsourcing incentive for radio environment maps: a dynamic pricing approach," in *IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, 2018.

[623] M. Asghari, "Dynamic pricing and task assignment in real-time spatial crowdsourcing platforms," PhD Thesis, University of Southern California, 2018.

[624] BOINC, "Create a virtual campus supercomputing center (VCSC)," 2011. [Online]. Available: http://boinc.berkeley.edu/trac/wiki/VirtualCampusSupercomputerCenter. [Accessed 10 August 2022].

[625] J. Deign, "How the Internet of Things is keeping trains on track," 31 March 2014. [Online]. Available: https://www.govtech.com/fs/how-the-internet-of-things-is-keeping-trains-on-track.html. [Accessed 18 August 2016].

[626] A. Jain and N. Tyagi, "Collision detection and avoidance in railways using WiMAX," *Indian Journal of Computer Science and Engineering,* vol. 3, no. 6, pp. 789-795, 2013.

[627] C. Elliott, "These airlines have the best Wi-Fi in the world," 14 January 2016. [Online]. Available: http://fortune.com/2016/01/14/airlines-wifi-internet/. [Accessed 25 August 2016].

[628] Chelsa, "List of airlines offering inflight WiFi," eDreams Blog, 27 July 2015. [Online]. Available: http://www.edreams.com/blog/in-flight-wifi/. [Accessed 10 August 2022].

[629] R. Qubein, "These 11 airlines offer fliers free in-flight Wi-Fi," Road Warrior Voices, 4 February 2016. [Online]. Available: https://www.usatoday.com/story/travel/roadwarriorvoices/2016/02/04/these-11-airlines-offer-fliers-free-in-flight-wi-fi/83276604/. [Accessed 10 August 2022].

[630] M. Williams, "How does airplane Wi-Fi work? And will it ever get any better?," FutureTech, 9 August 2013. [Online]. Available: http://www.in.techradar.com/news/world-of-tech/future-tech/How-does-airplane-Wi-Fi-work-And-will-it-ever-get-any-better/articleshow/38758474.cms. [Accessed 10 August 2022].

[631] B. Rapolu, "Internet of aircraft things: an industry set to be transformed," 18 January 2016. [Online]. Available: http://aviationweek.com/connected-aerospace/internet-aircraft-things-industry-set-be-transformed. [Accessed 10 August 2022].

[632] M. Satyanarayanan, "Mobile computing: the next decade," in *1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (MCS'10)*, New York, USA, 2010.

[633] A. Kate and I. Goldberg, "Distributed private-key generators for identity based cryptography," in *Security and Cryptography for Networks. Lecture Notes in Computer Science*, vol. 6280, J. A. Garay and R. De Prisco, Eds., Berlin, Heidelberg, Springer, 2010, pp. 436-453.

[634] T. Chang, C. Chen, H. Hsiao and G. Lai, "The cryptanalysis of WPA & WPA2 using the parallel-computing with GPUs," in *Mobile Internet Security (MobiSec 2016). Communications in Computer and Information Science*, vol. 797, I. You, F. Y. Leu, H. C. Chen and I. Kotenko, Eds., Singapore, Springer, 2018, pp. 118-127.

[635] L. Yong-lei and J. Zhi-gang, "Distributed method for cracking WPA/WPA2-PSK on multi-core CPU and GPU architecture," *International Journal of Communication Systems,* vol. 28, no. 4, pp. 723-742, 2015.

[636] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: vision and challenges," *IEEE Internet of Things Journal,* vol. 3, no. 5, pp. 637-646, 2016.

[637] P. K. D. Pramanik and P. Choudhury, "IoT data processing: the different archetypes and their security & privacy assessments," in *Internet of Things (IoT) Security: Fundamentals, Techniques and Applications*, S. K. Shandilya, S. A. Chun, S. Shandilya and E. Weippl, Eds., River Publishers, 2018, pp. 37-54.

[638] Team Digit, "Best 10 GB RAM mobile phones," Digit, 18 June 2019. [Online]. Available: https://www.digit.in/top-products/best-10-gb-ram-mobile-phones-592.html. [Accessed 2019 June 18].

[639] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley and L. V. Gool, "AI benchmark: running deep neural networks on Android smartphones," *arXiv,* no. 1810.01109v2, 2018.

[640] A. Iosup, O. Sonmez, S. Anoep and D. Epema, "The performance of bags-of-tasks in large-scale distributed systems," in *17th international symposium on High performance distributed computing (HPDC '08)*, 2008.

[641] M. Hussin, A. Abdullah and S. K. Subramaniam, "Adaptive resource allocation for reliable performance in heterogeneous distributed systems," in *Algorithms and Architectures for Parallel Processing (ICA3PP 2013). Lecture Notes in Computer Science*, vol. 8286, R. Aversa, J. Kołodziej, J. Zhang, F. Amato and G. Fortino, Eds., Springer, Cham, 2013, pp. 51-58.

[642] L. F. Bittencourt, A. Goldman, E. R. M. Madeira, N. L. d. Fonseca and R. Sakellariou, "Scheduling in distributed systems: A cloud computing perspective," *Computer Science Review,* vol. 30, no. November, pp. 31-54, 2018.

[643] A. Jahan, K. L. Edwards and M. Bahraminasab, Multi-criteria decision analysis for supporting the selection of engineering materials in product design, 2nd ed., Kidlington, Oxford: Butterworth-Heinemann, Elsevier, 2016.

[644] M. Žižović, D. Pamučar, M. Albijanić, P. Chatterjee and I. Pribićević, "Eliminating rank reversal problem using a new multi-attribute model - the RAFSI method," *Mathematics,* vol. 8, no. 6, p. 1015, 2020.

[645] J. Figueira, S. Greco and M. Ehrogott, Multiple criteria decision analysis: state of the art surveys, Springer, New York, 2005.

[646] A. Alinezhad and J. Khalili, New methods and applications in multiple attribute decision making (MADM), Springer, 2019.

[647] C. L. Hwang and K. Yoon, Multiple attribute decision making: methods and applications, New York, NY, USA: Springer, 1981.

[648] C.-L. Hwang, Y.-J. Lai and T.-Y. Liu, "A new approach for multiple objective decision making," *Computers & Operations Research,* vol. 20, no. 8, pp. 889-899, 1993.

[649] M. Keshavarz Ghorabaee, E. K. Zavadskas, L. Olfat and Z. Turskis, "Multi-criteria inventory classification using a new method of evaluation based on distance from average solution (EDAS)," *Informatica,* vol. 26, no. 3, pp. 435-451, 2015.

[650] D. Pamučar and G. Ćirović, "The selection of transport and handling resources in logistics centers using Multi-Attributive Border Approximation area Comparison (MABAC)," *Expert systems with applications,* vol. 42, no. 6, pp. 3016-3028, 2015.

[651] A. Alinezhad and J. Khalili, "MABAC Method," in *New Methods and Applications in Multiple Attribute Decision Making (MADM). International Series in Operations Research & Management Science*, vol. 277, Springer, Cham, 2019, pp. 193-198.

[652] E. K. Zavadskas and Z. Turskis, "A new additive ratio assessment (ARAS) method in multicriteria decision-making," *Technological and Economic Development of Economy,* vol. 16, no. 2, pp. 159-172, 2010.

[653] A. Alinezhad and J. Khalili, "ARAS method," in *New Methods and Applications in Multiple Attribute Decision Making (MADM). International Series in Operations Research & Management Science*, vol. 277, Springer, Cham, 2019, pp. 67-71.

[654] K. R. MacCrimmon, Decisionmaking among multiple-attribute alternatives: a survey and consolidated approach, Santa Monica Ca: Research Memorandum, 1968.

[655] E. K. Zavadskas, A. Kaklauskas and V. Sarka, "The new method of multi-criteria complex proportional assessment of projects," *Technological and Economic Development of Economy,* vol. 1, no. 3, pp. 131-139, 1994.

[656] A. Alinezhad and J. Khalili, "COPRAS method," in *New Methods and Applications in Multiple Attribute Decision Making (MADM). International Series in Operations Research & Management Science,* vol. 277, Springer, Cham, 2019, pp. 87-91.

[657] L. Duckstein and S. Opricovic, "Multiobjective optimization in river basin development," *Water Resources Research,* vol. 16, no. 1, pp. 14-20, 1980.

[658] S. Opricovic and G. H. Tzeng, "Compromise solution by MCDM methods: a comparative analysis of VIKOR and TOPSIS," *European Journal of Operational Research,* vol. 156, no. 2, pp. 445-455, 2004.

[659] M. Yazdani, P. Zarate, E. K. Zavadskas and Z. Turskis, "A combined compromise solution (CoCoSo) method for multi-criteria decision-making problems," *Management Decision,* vol. 57, no. 9, pp. 2501-2519, 2019.

[660] Ž. Stević, D. Pamučar, A. Puška and P. Chatterjee, "Sustainable supplier selection in healthcare industries using a new MCDM method: Measurement of alternatives and ranking according to COmpromise solution (MARCOS)," *Computers & Industrial Engineering,* vol. 140, p. 106231, 2020.

[661] M. Keshavarz Ghorabaee, E. K. Zavadskas, M. Amiri and Z. Turskis, "Extended EDAS method for fuzzy multi-criteria decision-making: An application to supplier selection," *International Journal of Computers Communications & Control,* vol. 11, pp. 358-371, 2016.

[662] D. Stanujkic and R. Jovanovic, "Measuring a quality of faculty website using ARAS method," in *Proceeding of the International Scientific Conference Contemporary Issues in Business, Management and Education,* 2012.

[663] E. K. Zavadskas, Z. Turskis and T. Vilutiene, "Multiple criteria analysis of foundation instalment alternatives by applying additive ratio assessment (ARAS) method," *Archives of Civil and Mechanical Engineering,* vol. 10, no. 3, pp. 123-141, 2010.

[664] C. Ghenai, M. Albawab and M. Bettayeb, "Sustainability indicators for renewable energy systems using multi-criteria decision-making model and extended SWARA/ARAS hybrid method," *Renewable Energy,* vol. 146, pp. 580-597, 2020.

[665] L. Balezentiene and A. Kusta, "Reducing greenhouse gas emissions in grassland ecosystems of the central Lithuania: multi-criteria evaluation on a basis of the ARAS method," *The Scientific World Journal,* no. Article ID 908384, 2012.

[666] P. Hoan and Y. Ha, "ARAS-FUCOM approach for VPAF fighter aircraft selection," *Decision Science Letters,* vol. 10, no. 1, pp. 53-62, 2021.

[667] J. Roy, A. Ranjan, A. Debnath and S. Kar, "An extended MABAC for multi-attribute decision making using trapezoidal interval type-2 fuzzy numbers," *arXiv preprint ,* no. arXiv:1607.01254, 2016.

[668] Z. Bobar, D. Božanić, K. Djurić and D. Pamučar, "Ranking and assessment of the efficiency of social media using the fuzzy AHP-Z number model-fuzzy MABAC," *Acta Polytech Hungarica,* vol. 17, pp. 43-70, 2020.

[669] G. Büyüközkan, E. Mukul and E. Kongar, "Health tourism strategy selection via SWOT analysis and integrated hesitant fuzzy linguistic AHP-MABAC approach," *Socio-Economic Planning Sciences,* vol. 74, p. 100929, 2021.

[670] S. Biswas, G. Bandyopadhyay, B. Guha and M. Bhattacharjee, "An ensemble approach for portfolio selection in a multi-criteria decision making framework," *Decision Making: Applications in Management and Engineering,* vol. 2, no. 2, pp. 138-158, 2019.

[671] H. K. Sharma, J. Roy, S. Kar and O. Prentkovskis, "Multi criteria evaluation framework for prioritizing Indian railway stations using modified rough AHP-MABAC method," *Transport and Telecommunication Journal,* vol. 19, no. 2, pp. 113-127, 2018.

[672] J. Roy, K. Chatterjee, A. Bandyopadhyay and S. Kar, "Evaluation and selection of medical tourism sites: A rough analytic hierarchy process based multi-attributive border approximation area comparison approach," *Expert Systems,* vol. 35, no. 1, p. e12232, 2018.

[673] S. M. Yu, J. Wang and J. Q. Wang, "An interval type-2 fuzzy likelihood-based MABAC approach and its application in selecting hotels on a tourism website," *International Journal of Fuzzy Systems,* vol. 19, no. 1, pp. 47-61, 2017.

[674] P. Chatterjee and S. Chakraborty, "Flexible manufacturing system selection using preference ranking methods: A comparative study," *International Journal of Industrial Engineering Computations,* vol. 5, no. 2, pp. 315-338, 2014.

[675] E. K. Zavadskas, A. Kaklauskas, Z. Turskis and J. Tamošaitienė, "Multi-attribute decision-making model by applying grey numbers," *Informatica,* vol. 20, no. 2, pp. 305-320, 2009.

[676] Ž. Stević and N. Brković, "A novel integrated FUCOM-MARCOS model for evaluation of human resources in a transport company," *Logistics,* vol. 4, no. 1, p. 4, 2020.

[677] M. Stanković, Ž. Stević, D. K. Das, M. Subotić and D. Pamučar, "A new fuzzy MARCOS method for road traffic risk analysis," *Mathematics,* vol. 8, no. 3, p. 457, 2020.

[678] C. E. Shannon, "A mathematical theory of communication," *Bell Systems Technical Journal,* vol. 27, no. 3, pp. 379-423, 1948.

[679] Y. Suh, Y. Park and D. Kang, "Evaluating mobile services using integrated weighting approach and fuzzy VIKOR," *PLoS ONE,* vol. 14, no. 6, p. e0217786, 2019.

[680] M. Z. Abidin, R. Rusli and A. M. Shariff, "Technique for order performance by similarity to ideal solution (TOPSIS) - entropy methodology for inherent safety design decision making tool," *Procedia Engineering,* vol. 148, pp. 1043-1050, 2016.

[681] P. Liu and X. Zhang, "Research on the supplier selection of a supply chain based on entropy weight and improved ELECTRE-III method," *International Journal of Production Research,* vol. 49, no. 3, pp. 637-646, 2011.

[682] S. Laha and S. Biswas, "A hybrid unsupervised learning and multi-criteria decision making approach for performance evaluation of Indian banks," *Accounting,* vol. 5, no. 4, pp. 169-184, 2019.

[683] S. Gupta, G. Bandyopadhyay, M. Bhattacharjee and S. Biswas, "Portfolio selection using DEA-COPRAS at risk–return interface based on NSE (India)," *International Journal of Innovative Technology and Exploring Engineering,* vol. 8, no. 10, pp. 4078-4086, 2019.

[684] P. Karmakar, P. Dutta and S. Biswas, "Assessment of mutual fund performance using distance based multi-criteria decision making techniques - An Indian perspective," *Research Bulletin,* vol. 44, no. 1, pp. 17-38, 2018.

[685] X. Li, K. Wang, L. Liu, J. Xin, H. Yang and C. Gao, "Application of the entropy weight and TOPSIS method in safety evaluation of coal mines," *Procedia Engineering,* vol. 26, pp. 2085-2091, 2011.

[686] Z. H. Zou, Y. Yi and J. N. Sun, "Entropy method for determination of weight of evaluating indicators in fuzzy synthetic evaluation for water quality assessment," *Journal of Environmental Sciences,* vol. 18, no. 5, pp. 1020-1023, 2006.

[687] R. Simanaviciene and L. Ustinovichius, "Sensitivity analysis for multiple criteria decision making methods: TOPSIS and SAW," *Procedia-Social and Behavioral Sciences,* vol. 2, no. 6, pp. 7743-7744, 2010.

[688] I. Mukhametzyanov and D. S. Pamučar, "A sensitivity analysis in MCDM problems: A statistical approach," *Decision Making: Applications in Management and Engineering,* vol. 1, no. 2, pp. 51-80, 2018.

[689] S. Biswas and D. S. Pamučar, "Facility location selection for b-schools in Indian context: A multi-criteria group decision based analysis," *Axioms,* vol. 9, no. 3, p. 77, 2020.

[690] D. S. Pamučar, G. Ćirović and D. Božanić, "Application of interval valued fuzzy-rough numbers in multi-criteria decision making: the IVFRN-MAIRCA model," *Yugoslav Journal of Operations Research,* vol. 29, no. 2, pp. 221-247, 2019.

[691] D. S. Pamučar, D. Božanić and A. Ranđelović, "Multi-criteria decision making: An example of sensitivity analysis," *Serbian Journal of Management,* vol. 12, no. 1, pp. 1-27, 2017.

[692] Z. Ali, T. Mahmood, K. Ullah and Q. Khan, "Einstein geometric aggregation operators using a novel complex interval-valued Pythagorean fuzzy setting with application in green supplier chain management," *Reports in Mechanical Engineering,* vol. 2, no. 1, pp. 105-134, 2021.

[693] S. Biswas and O. P. Anand, "Logistics competitiveness index-based comparison of BRICS and G7 countries: an integrated PSI-PIV approach," *IUP Journal of Supply Chain Management,* vol. 17, no. 2, pp. 32-57, 2020.

[694] K. E. Muller and B. A. Fetterman, Regression and ANOVA: an integrated approach using SAS software, New York, United States: John Wiley & Sons, 2003.

[695] M. Allen, Ed., "Post hoc tests," in *The SAGE Encyclopedia of Communication Research Methods*, Vols. 1-4, SAGE Publications, 2017.

[696] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks (ICNN'95)*, Perth, Australia, 1995.

[697] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Evolutionary Programming VII (EP 1998). Lecture Notes in Computer Science*, vol. 1447, V. W. Porto, N. Saravanan, D. Waagen and A. E. Eiben, Eds., Berlin, Heidelberg, Springer, 1998, pp. 611-616.

[698] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *USENIX conference on USENIX annual technical conference (USENIXATC'10)*, Berkeley, United States, 2010.

[699] B. Zhu and Y. Wei, "Carbon price forecasting with a novel hybrid ARIMA and least squares support vector machines methodology," *Omega,* vol. 41, no. 3, pp. 517-524, 2013.

[700] N. H. Miswan, N. A. Ngatiman, K. Hamzah and Z. Z. Zamzamin, "Comparative performance of ARIMA and GARCH models in modelling and forecasting volatility of Malaysia market properties and shares," *Applied Mathematical Sciences,* vol. 8, no. 140, pp. 7001-7012, 2014.

[701] R. Fu, Z. Zhang and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, Wuhan, China, 2016.

[702] N. N. Zakaria, M. Othman, R. Sokkalingam, H. Daud, L. Abdullah and E. A. Kadir, "Markov chain model development for forecasting air pollution index of Miri, Sarawak," *Sustainability,* vol. 11, p. 5190, 2019.

[703] S. Elgharbi, M. Esghir, O. Ibrihich, A. Abarda, S. El Hajji and S. Elbernoussi, "Grey-Markov model for the prediction of the electricity production and consumption," in *Big Data and Networks Technologies (BDNT 2019). Lecture Notes in Networks and Systems*, vol. 81, Y. Farhaoui, Ed., Cham, Springer, 2020, pp. 206-219.

[704] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks,* vol. 5, no. 2, pp. 157-166, 1994.

[705] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[706] A. Yadava, C. K. Jhaa and A. Sharan, "Optimizing LSTM for time series prediction in Indian stock market," *Procedia Computer Science,* vol. 167, pp. 2091-2100, 2020.

[707] Y.-g. Zhang, J. Tang, Z.-y. He, J. Tan and C. Li, "A novel displacement prediction method using gated recurrent unit model with time series analysis in the Erdaohe landslide," *Natural Hazards,* vol. 105, pp. 783-813, 2021.

[708] Q. Tan, M. Ye, B. Yang, S. Liu, A. J. Ma, T. C.-F. Yip, G. L.-H. Wong and P. Yuen, "DATA-GRU: dual-attention time-aware gated recurrent unit for irregular multivariate time series," *Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 34, no. 1, pp. 930-937, 2020.

[709] M. d. Caux, F. Bernardini and J. V. Viterbo, "Short-term forecasting in Bitcoin time series using LSTM and GRU RNNs," in *Symposium on Knowledge Discovery, Mining and Learning (KDMILE 2020)*, Rio Grande, Brazil, 2020.

[710] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics,* vol. 36, pp. 193-202, 1980.

[711] Y. LeCun, B. Boser, J. S. Denker, R. E. Howard, W. Hubbard, L. D. Jackel and D. Henderson, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems,* vol. 2, pp. 396-404, 1990.

[712] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[713] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang and W. Xu, "CNN-RNN: a unified framework for multi-label image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016.

[714] A. Borovykh, S. Bohte and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv*, no. 1703.04691v5, 2018.

[715] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao and S. Yan, "HCP: a flexible CNN framework for multi-label image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1901-1907, 2016.

[716] C.-L. Liu, W.-H. Hsaio and Y.-C. Tu, "Time series classification with multivariate convolutional neural network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4788-4797, 2019.

[717] W. Shen, Z. Wei, C. Yang and R. Zhang, "Travel pattern modelling and future travel behaviour prediction based on GMM and GPR," *Int. J. Simulation and Process Modelling*, vol. 13, no. 6, pp. 548-556, 2018.

[718] F. Jelinek, R. L. Mercer, L. R. Bahl and J. Baker, "Perplexity - a measure of the difficulty of speech recognition tasks," *The Journal of the Acoustical Society of America*, vol. 62, no. S1, p. S63, 1977.

[719] F Chollet and others, "Keras," 2015. [Online]. Available: https://keras.io/. [Accessed 13 February 2021].

[720] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," *arXiv:1412.6980v9 [cs.LG]*, 2017.

[721] R. Muthukrishnan and R. Rohini, "LASSO: a feature selection technique in predictive modeling for machine learning," in *IEEE International Conference on Advances in Computer Applications (ICACA)*, Coimbatore, India, 2016.

[722] F. Perez and B. E. Granger, "IPython: a system for interactive scientific computing," *Computing in Science & Engineering*, vol. 9, no. 3, 2007.

[723] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, et al., "TensorFlow: large-scale machine learning on heterogeneous distributed systems," Google, 2015.

[724] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, et al., "Array programming with NumPy," *Nature*, vol. 585, pp. 357-362, 2020.

[725] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, et al., "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261-72, 2020.

[726] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, et al., "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825-2830, 2011.

[727] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, 2007.

[728] NVIDIA, P. Vingelmann and F. H. Fitzek, "CUDA, release: 10.2.89," 2020. [Online]. Available: https://developer.nvidia.com/cuda-toolkit. [Accessed 3 July 2021].

[729] S. Yang, X. Yu and Y. Zhou, "LSTM and GRU neural network performance comparison study: taking Yelp review dataset as an example," in *International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, Shanghai, China, 2020.

[730] Y. Bengio, A. Courville and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, 2013.

[731] M. McNett and G. M. Voelker, "Access and mobility of wireless PDA users," *Mobile Computing and Communications Review*, vol. 9, no. 2, 2005.

[732] N. Balakrishnan, Handbook of the logistic distribution, CRC Press, 2019.

[733] J. M. Hilbe, Logistic regression models, Chapman and Hall/CRC, 2017.

[734] A. Agresti, Categorical data analysis, 3rd ed., Wiley, 2012.

[735] D. Hosmer Jr., S. Lemeshow and R. X. Sturdivant, Applied logistic regression, 3rd ed., Wiley, 2013.

[736] C. V. Zavgren, "Assessing the vulnerability of American industrial firms: a logistic analysis," *Journal of Business Finance and Accounting*, vol. 12, no. 1, pp. 19-45, 1985.

[737] J. A. Ohlson, "Financial ratios and the Probabilistic prediction of bankruptcy," *Journal of Accounting Research*, vol. 18, no. 1, pp. 109-131, 1p80.

[738] S. W. Menard, Quantitative applications in the social sciences: applied logistic regression analysis, 2nd ed., Thousand Oaks, CA: SAGE Publications, Inc., 2002.

[739] P. Choudhury, S. Nandi and N. C. Debnath, “A publish/subscribe system using distributed broker for SOA based MANET applications,” *Journal of Computational Methods in Sciences and Engineering,* vol. 12, no. S1, pp. 129-138, 2012.

[740] J. Surbiryala and C. Rong, “Cloud computing: history and overview,” in *IEEE Cloud Summit*, Washington, DC, USA, 2019.

[741] E. F. Coutinho, F. Sousa, P. Rego, D. Gomes and J. Souza, “Elasticity in cloud computing: a survey,” *Annals of Telecommunications - annales des télécommunications,* vol. 70, p. 289–309, 2015.

[742] S. Yi, Z. Hao, Z. Qin and Q. Li, “Fog computing: platform and applications,” in *Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, Washington DC, 2015.

[743] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, “The case for VM-based cloudlets in mobile computing,” *IEEE Pervasive Computing,* vol. 8, no. 4, pp. 14-23, 2009.

[744] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu and B. Amos, “Edge analytics in the Internet of Things,” *IEEE Pervasive Computing,* vol. 14, no. 2, pp. 24-31, 2015.

[745] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, “Fog computing and its role in the Internet of Things,” in *MCC workshop on Mobile cloud computing (MCC '12)*, Helsinki, Finland, 2012.

[746] L. M. Vaquero and L. Rodero-Merino, “Finding your way in thefFog: towards a comprehensive definition of fog computing,” *ACM SIGCOMM Computer Communication Review,* vol. 44, no. 5, pp. 27-32, 2014.

[747] Y. C. Hu; M. Patel; D. Sabella; N. Sprecher; V. Young, “Mobile edge computing: a key technology towards 5G,” ETSI (European Telecommunications Standards Institute), September 2015.

[748] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta and D. Sabella, “On multi-access edge computing: a survey of the emerging 5G network edge cloud architecture and orchestration,” *IEEE Communications Surveys & Tutorials,* vol. 19, no. 3, pp. 1657-1681, 2017.

[749] N. H. Chun, *Resource auction in crowdsourced edge computing platform,* Hong Kong University of Science and Technology, 2021.

[750] P. Antoniadis and I. Apostol, “The right (s) to the hybrid city and the role of DIY networking," *The Journal of Community Informatics,* vol. 10, no. 3, 2014.

[751] L. Peterson, T. Anderson, S. Katti, N. McKeown, G. Parulkar, J. Rexford, M. Satyanarayanan, O. Sunay and A. Vahdat, “Democratizing the network edge,” *ACM SIGCOMM Computer Communication Review ,* vol. 49, no. 2, pp. 31-36, 2019.

[752] B. Yang, F. Haghighat, B. C. M. Fung and K. Panchabikesan, “Season-based occupancy prediction in residential buildings using machine learning models,” *e-Prime - Advances in Electrical Engineering, Electronics and Energy,* vol. 1 (100003), 2021.

[753] W. Goetzler, R. Shandross, J. Young, O. Petritchenko, D. Ringo and S. McClive, “Energy savings potential and RD&D opportunities for commercial building HVAC systems,” U.S. Department of Energy, 2017.

[754] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra and P. Bahl, “MAUI: making smartphones last longer with code offload,” in *8th international conference on mobile systems, applications, and services*, San Francisco, 2010.

[755] K. Ha, P. Pillai, G. Lewis, S. Simanta, S. Clinch, N. Davies and M. Satyanarayanan, “The impact of mobile multimedia applications on data center consolidation,” in *IEEE International Conference on Cloud Engineering*, San Francisco, CA, 2013.

[756] M. Satyanarayanan, “A brief history of cloud offload: a personal journey from odyssey through cyber foraging to cloudlets,” *GetMobile: Mobile Computing and Communications,* vol. 18, no. 4, pp. 19-23, 2015.

[757] R. G. Steadman, “The assessment of sultriness. Part I: a temperature-humidity index based on human physiology and clothing science,” *Journal of Applied Meteorology and Climatology,* vol. 18, no. 7, p. 861–873, 1979.

[758] A. Quinn, J. D. Tamerius, M. Perzanowski, J. S. Jacobson, I. Goldstein, L. Acosta and J. Shaman, “Predicting indoor heat exposure risk during extreme heat events,” *Science of The Total Environment,* vol. 490, p. 686–693, 2014.

[759] Paroscientific Inc., “MET4 and MET4A calculation of dew point,” 26 May 2012. [Online]. Available: https://archive.is/20120526034637/http://www.paroscientific.com/dewpoint.htm#selection-413.0-413.39. [Accessed 7 June 2020].

[760] A. W. T. Barenbrug, Psychrometry and psychrometric charts, 3rd ed., Cape Town: Cape and Transvaal Printers Ltd., 1974.

[761] G. B. Anderson, M. L. Bell and R. D. Peng, "Methods to calculate the heat index as an exposure metric in environmental health research," *Environmental health perspectives,* vol. 121, no. 10, pp. 1111-1119, 2013.

[762] "NodeMCU v3," Zerynth, 2020. [Online]. Available: https://docs.zerynth.com/latest/official/board.zerynth.nodemcu3/docs/index.html. [Accessed 21 March 2022].

[763] T. Liu, "Digital-output relative humidity & temperature sensor/module: DHT22," [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf. [Accessed 28 March 2020].

[764] Z. Juhasz, "Quantitative cost comparison of on-premise and cloud infrastructure based EEG data processing," *Cluster Computing,* vol. 24, pp. 625-641, 2021.

[765] B. Daigle, "Data centers around the world: a quick look," May 2021. [Online]. Available: https://www.usitc.gov/publications/332/executive_briefings/ebot_data_centers_around_the_world.pdf. [Accessed 12 February 2022].

[766] M. Koot and F. Wijnhoven, "Usage impact on data center electricity needs: a system dynamic forecasting model," *Applied Energy,* vol. 291, 2021.

[767] P. Kanani and M. Padole, "An effort to reduce the CO2 emission in computation for green computation," *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology,* vol. 12, no. SUP 1, pp. 340-346, 2020.

[768] T. Pirson and D. Bol, "Assessing the embodied carbon footprint of IoT edge devices with a bottom-up life-cycle approach," *Journal of Cleaner Production,* vol. 322, p. 128966, 2021.

[769] M. A. E. Aziz, "How smartphones are contributing to climate change," 7 February 2022. [Online]. Available: https://infomineo.com/how-smartphones-are-contributing-to-climate-change/#:~:text=Research%20on%20the%20annual%20carbon,per%20day%2C%20for%20a%20year.. [Accessed 23 June 2022].

[770] E. M. Ercan, "Global warming potential of a smartphone using life cycle assessment methodology," Royal Institute of Technology, Stockholm, 2013.

[771] Five Winds International, "Toxic and hazardous materials in electronics an environmental scan of toxic and hazardous materials in IT and telcomm products and waste," Five Winds International, Ottawa, Canada, 2001.

[772] ITU Telecommunication Development Sector, "The global e-waste monitor 2020," 2020. [Online]. Available: https://www.itu.int/en/ITU-D/Environment/Pages/Spotlight/Global-Ewaste-Monitor-2020.aspx. [Accessed 28 December 2022].

[773] S. Dalul, "To solve the smartphone e-waste problem we first need fewer disposable devices," Android Authority, 20 July 2020. [Online]. Available: https://www.androidauthority.com/e-waste-smartphones-1133322/. [Accessed 5 May 2022].

[774] M. Chatterji, "Repairing – not recycling – is the first step to tackling e-waste from smartphones. Here's why.," World Economic Forum, 19 July 2021. [Online]. Available: https://www.weforum.org/agenda/2021/07/repair-not-recycle-tackle-ewaste-circular-economy-smartphones/#:~:text=Emissions%20and%20waste&text=The%20total%20annual%20carbon%20footprint,50%20million%20tonnes%20in%202019. [Accessed 5 May 2022].

[775] Sims Lifecycle Services, "Sustainable data center decommissioning," Sims Lifecycle Services, 2003. [Online]. Available: https://www.simslifecycle.com/resources/white-paper-data-center/. [Accessed 06 May 2022].

[776] Sims Lifecycle Services, "How we do it," Sims Lifecycle Services, 2003. [Online]. Available: https://www.simslifecycle.com/business/e-waste-recycling/how-we-do-it/. [Accessed 06 May 2022].

[777] Z. Chen, M. Yang, Q. Shi, X. Kuang, H. J. Qi and T. Wang, "Recycling waste circuit board efficiently and environmentally friendly through small-molecule assisted dissolution," *Scientific Reports,* vol. 17902, pp. 1-9, 2019.

[778] H. Vermeșan , A.-E. Tiuc and M. Purcar, "Advanced recovery techniques for waste materials from IT and telecommunication equipment printed circuit boards," *Sustainability,* vol. 12, no. 1, pp. 1-23, 2020.

[779] G. P. Thomas, "Recycling of mobile phones," AZO Cleantech, 30 August 2012. [Online]. Available: https://www.azocleantech.com/article.aspx?ArticleID=275. [Accessed 6 May 2022].

[780] W. Hajji and F. P. Tso, “Understanding the performance of low power Raspberry Pi cloud for big data,” *Electronics,* vol. 5, no. 2, p. 29, 2016.

[781] A. Komninos, I. Simou, N. Gkorgkolis and J. Garofalakis, “Performance of Raspberry Pi microclusters for edge machine learning in tourism,” in *Joint Proceeding of the Poster and Workshop Sessions of AmI-2019, the 2019 European Conference on Ambient Intelligence*, Rome, Italy, 2019.

[782] N. Naji, M. R. Abid, N. Krami and D. Benhaddou, “Energy-aware wireless sensor networks for smart buildings: a review,” *Journal of Sensor and Actuator Networks,* vol. 10, no. 4, p. 67, 2021.

[783] N. Malpani, J. L. Welch and N. Vaidya, “Leader election algorithms for mobile ad hoc networks,” in *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM '00)*, Boston, USA, 2000.

[784] R. Septiana, I. Roihan and R. A. Koestoer, “Testing a calibration method for temperature sensors in different working fluids,” *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences,* vol. 68, no. 2, pp. 84-93, 2020.

[785] R. A. Koestoer, N. Pancasaputra, I. Roihan and H. Harinaldi, “A simple calibration methods of relative humidity sensor DHT22 for tropical climates based on Arduino data acquisition system,” *AIP Conference Proceedings,* vol. 2062, no. 1, p. 020009, 2019 .

[786] N. Li, G. Calis and B. Becerik-Gerber, “Measuring and monitoring occupancy with an RFID based system for demand-driven HVAC operations,” *Automation in Construction,* vol. 24, pp. 89-99, 2012.

[787] H. Saha, A. R. Florita, G. P. Henze and S. Sarkar, “Occupancy sensing in buildings: A review of data analytics approaches,” *Energy and Buildings,* vol. 188–189, pp. 278-285, 2019.

[788] C. Lee and D. Lee, “Self-error detecting and correcting algorithm for accurate occupancy tracking using a wireless sensor network,” in *4th International Conference on Smart and Sustainable Technologies (SpliTech),* Split, Croatia, 2019.

[789] L. Klein, J.-y. Kwak, G. Kavulya, F. Jazizadeh, B. Becerik-Gerber, P. Varakantham and M. Tambe, “Coordinating occupant behavior for building energy and comfort management using multi-agent systems,” *Automation in Construction,* vol. 22, pp. 525-536, 2012.

[790] P. W. Tien, S. Wei, J. K. Calautit, J. Darkwa and C. Wood, “A vision-based deep learning approach for the detection and prediction of occupancy heat emissions for demand-driven control solutions,” *Energy and Buildings,* vol. 226 (Article 110386), 2020.

[791] Y. Benezeth, H. Laurent, B. Emile and C. Rosenberger, “Towards a sensor for detecting human presence and characterizing activity,” *Energy and Buildings,* vol. 43, pp. 305-314, 2011.

[792] V. L. Erickson, Y. Lin, A. Kamthe, R. Brahme, A. Surana, A. E. Cerpa, M. D. Sohn and S. Narayanan, “Energy efficient building environment control strategies using real-time occupancy measurements,” in *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings,* New York, NY, USA, 2009.

[793] H. Shan, Y. Da, A. Jingjing, G. Siyue and Q. Mingyang, “Investigation and analysis of Chinese residential building occupancy with large-scale questionnaire surveys,” *Energy and Buildings,* vol. 193, pp. 289-304, 2019.

[794] H. Zou, Y. Zhou, H. Jiang, S.-C. Chien, L. Xie and C. J. Spanos, “WinLight: a WiFi-based occupancy-driven lighting control system for smart building,” *Energy and Buildings,* vol. 158, pp. 924-938, 2018.

[795] C. B. Smith and K. E. Parmenter, “Management of heating and cooling,” in *Energy Management Principles: Applications, Benefits, Savings*, 2nd ed., Elsevier, 2016, pp. 125-187.

[796] D. Youn, “Air conditioning system using enthalpy of outside air”. United States Patent US20030183380A1, 2 October 2003.

[797] C.-L. Li and S.-L. Chung, “Enthalpy-based automatic temperature control for automobiles,” in *IEEE Control Applications (CCA) & Intelligent Control (ISIC)*, St. Petersburg, Russia, 2009.

[798] R. Kosonen and J. Penttinen, “The effect of free cooling and demand-based ventilation on energy consumption of self-regulating and traditional chilled beam systems in cold climate,” *Indoor and Built Environment,* vol. 26, no. 2, pp. 256-271, 2017.