QoE BASED MULTI CRITERIA DECISION MAKING APPROACHES FOR RESOURCE ALLOCATION AND SCHEDULING IN FOG COMPUTING

Thesis submitted in fulfillment of the requirements for the Degree of

DOCTOR OF PHILOSOPHY

By

SHEFALI VARSHNEY

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
Waknaghat, Solan – 173234, Himachal Pradesh, INDIA

May, 2024

DECLARATION

I hereby declare that the work reported in Ph.D. thesis entitled **"QoE based multi criteria decision making approaches for resource allocation and resource scheduling in Fog computing"** submitted at **"Jaypee University of Information Technology, Wakhnaghat, Solan (H.P), India"**, is an authentic record of my work carried out under the supervision of **"Prof. (Dr.) P.K. Gupta and Dr. Rajinder Sandhu"**. I have not submitted this work elsewhere for anyother degree or diploma. I am fully responsible for the contents of my Ph.D. Theses.

(Signature of Scholar)

(Shefali Varshney)

Department of Computer Science & Engineering

Jaypee University of Information Technology, Wakhnaghat,

Solan (H.P), India

Date:

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the Ph.D. thesis entitled **"QoE based multi criteria decision making approaches for resource allocation and resource scheduling in Fog computing"** submitted by **Ms. Shefali Varshney**, Enrollment no. 196205 at **Jaypee University of Information Technology, Wakhnaghat, Solan (HP), India**, is a bonafide record of her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

(Signature of Supervisor)
Dr. P.K. Gupta
Professor
Department of Computer Science & Engineering
Jaypee University of Information Technology, Wakhnaghat,
Solan (H.P), India
Date:

(Signature of External Supervisor)
Dr. Rajinder Sandhu
Data Scientist, TD
Toronto, Ontario,
Canada
Date:

ACKNOWLEDGEMENT

With the providential grace of "**Almighty God**", the expedition of my Ph.D. came to an end and my heart is overflowing with appreciation towards each and every person who has lent a hand in the form of support, belief, and efforts to accomplish this journey.

It is an immense pleasure to express my profound gratitude towards my supervisors **Dr. P.K. Gupta**, **Professor**, Department of Computer Science & Engineering and Information Technology, JUIT, Wakhnaghat, Solan, and **Dr. Rajinder Sandhu**Data Scientist at TD who graciously allowed me to work under their guidance. I am thankful for their patience, continuous support, optimistic approach, never-ending deliberations, time-to-time guidance, and liberty throughout this course. I will always stay indebted to them for bearing my shortcomings with their immense sense of awareness, maturity, thorough knowledge of the specific field, and consistency.

I would like to express my gratitude to our Honourable Vice-Chancellor **Prof. (Dr.) Rajendra Kumar Sharma** and Dean (Academics and Research) **Prof. (Dr.) Ashok Kumar Gupta** and Head Department of CSE/IT **Prof. (Dr.) Vivek Kumar Sehgal** to promote the research and facilitate resources in the institution. I would also like to pay my gratitude to the DPMC members**Dr. Ekta Gandotra, Dr. Ruchi Verma, and Dr. Saurav**for their thought-provoking interactive assessments, queries, and opinions. Their valuable motivation, help, suggestions, affirmative vision, magnificent supervision, and enormous confidence in my abilities made me face tough circumstances during the progress of the research work.

I am grateful to my entire family for supporting me through all these years in my PhD journey.Their belief in me has kept my spirits and motivation high during this process. I am obliged for all the support received from all the faculty members and staff of the Department of CSE & IT for their scholarly support and guidance. I thank my fellow Ph.D. friends for their consistent help and valuable discussions.

ABSTRACT

In order to address the demands of contemporary technologies like the Internet of Things (IoT), Artificial Intelligence (AI), 5G, and more similar elements, Fog computing is working as an extended platform of cloud computing. The advancement of numerous application scenarios, including healthcare, smart cities, transportation, entertainment, and agriculture, which have a substantial impact on people's daily lives, is being facilitated by the IoT paradigm. These apps must have the processing and storage power to handle the massive volume of data prepared by IoT devices. IoT devices cannot effectively process and store significant amounts of data due to their inherent resource limitations. Therefore, IoT devices need substitute resources to ensure the efficient execution of their diverse applications, some of which may be computation-intensive or latency-sensitive. One of the potential resource suppliers for IoT devices is the cloud. Although it impacts the amount of time IoT devices are actively using energy. Subsequently, the usage of smart apps that respond instantly has increased significantly along with the use of IoT-enabled devices. Numerous problems are imposed by this increasing demand, including scheduling, pricing, server overload, etc. Fog servers, in contrast to Cloud servers, have resource restrictions that restrict them from running all IoT application types, notably those that require a lot of computing.

Fog servers, in contrast to Cloud servers, have resource restrictions that restrict them from running all IoT application types, notably those that require a lot of computing. Therefore, by storing the data on local Fog Nodes (FN), rather than adding to the load on the cloud, Fog Computing expands its services to include cloud computing. The two most pressing problems that Fog Computing has are Resource Allocation and Resource Management. The Fog computing paradigm as a result is very dynamic, distributed, and heterogeneous. Thus, it is challenging to fully realize the potential of this computing paradigm for various IoT-driven application scenarios without efficient scheduling approaches for the administration of IoT applications. As a result, it can minimize network congestion and speed up the delivery of application services. The major processing is done by Fog nodes which are heterogeneous and dispersed in nature. Whereas, some significant nodes have resource and spatial sharing limitations. Therefore, for various smart apps use cases it might be complicated to leverage its benefits without effective administration. The administration of computing resources includes the management of applications. By locating appropriate placement alternatives for the applications within the computer infrastructure, it may be made sure. The problems with

resource management are caused by resource heterogeneity and resource depletion. These problems have been thought to be significant concerns in the Fog environment. It appears that creating such resource management strategies in the Fog is quite challenging. In this kind of issue, multi-criteria decision-making (MCDM) strategies are very beneficial. Additionally, problems with resource management fall under MCDM challenges. One of the well-known MCDM methods that has been taken into account for selection and ranking is the Analytical Hierarchical Process (AHP). Selected Quality of Experience (QoE) criteria have been used to evaluate this resource management strategy. The suggested method helps in monitoring the Fog resources about their value and ranking.

Moreover, the matter of efficient resource allocation in the Fog layer is imposed by an increasing number of smart apps that are aware of the delay. For resource allocation and ranking, we have implemented an efficient MCDM-based solution in this study. The suggested algorithms incorporate the TOPSIS and AHP technique improved versions. This framework takes into account QoE parameters, such as network bandwidth, no. of cores, and average latency. The advised framework surpassedthe performance of the other existing methods when compared with different performance metrics. For the distribution of smart applications, a cost-effective scheduling strategy has been developed to address these issues. The purpose of this study is to maximize user benefits fromFog the environment while lowering thecost of smartapplications. The suggested framework was assessed using a test bed which consists of three analysis phases, and it is compared using five metrics: average allocation time, average Fog Environment profit, averagecost of smart apps,resource utilization, andnumber ofapplications running within a certain latency. Performance analysis shows that the used technique is performing better in all the criteria.

LIST OF ABBREVIATIONS

AMCDM	Adaptive Multi criteria decision making
AHP	Analytical Hierarchical Process
ANP	Analytical Network Process
AI	Artificial Intelligence
AGWO	Adaptive Grey Wolf Optimization
BLA	Bee Life Algorithm
BSPP	Basic Service Placement Problem
CRACCR	Communication resource-aware cooperative with computation resources
CDCS	Cloud data center
CSC	Cloud service customers
DRL	Deep Reinforcement Learning
DDLD	Delay-Driven Load Distribution
DOTS	delay-optimal task scheduling
DO	Device Only
DC	Device to Cloud
ENORM	Edge NOde Resource Management
EGA	Efficient resource allocation
EE	Energy efficiency
ETSI	European Telecommunications Standard Institute
EM	Expectation maximization
F2C	Fog-to-cloud
FCFS	First Come First Served
FCMCPS	Fog computing supported MCPS
FN	Fog Nodes
FNSA	FN scale adjustment
FOCAN	Fog Computing Architecture Network
FOE	Fog of Everything
FON	Fog Orchestrator Node
FR	Free Resource Fog
GA	Genetic Algorithm
GSP	Greatest satisfactory proportion
GKS	Greedy knapsack-based scheduling

HPC	High Performance Computing
HTTP	Hyper Text Transfer Protocol
IGA	Improved Genetic Algorithm
ILP	Integer Linear Programming
IT	Information Technology
ITU-T	International Telecommunication Union
IoE	Internet Of Everything
IoT	Internet Of Things
lsp	Least Satisfied Percentage
LP	Linear Programming
MtLDF	Multi-tenant Load Distribution Technique for Fog Situations
MEETS	Maximal energy-efficient task scheduling
MCDA	Multi-criteria decision alternative
MCPSs	Medical Cyber-Physical Systems
mdcs	Micro Data Center
MILP	Mixed-Integer Linear Programming
MM	Multilevel And Multidimensional
MDs	Mobile Devices
MPSO	Modified Particle Swarm Optimization
MCDM	Multi-Criteria Decision Making
NFV	Network Function Virtualization
NRR	Network Relaxation Ration
NSGA II	non-dominated sorting genetic algorithm II
NIST	The National Institute of Standards and Technology
OTA	Over The Air
PRA	Partial Rounding Algorithm
PASHE	Privacy-Aware Scheduling In A Heterogeneous Fog Environment
PTPN	Priced-Timed Petri Nets
PSO	Particle Swarm Optimization
PTRR	Processing Time Relaxation Ratio
QoE	Quality of Experience
QoS	Quality of Service
RECAP	REliableCApacity Provisioning and enhanced remediation for distributed cloud applications
RPM	Resource Pool Manager
RG	Resource Gain

SDN	Software Defined Network
SCATTER	Clustering Of Fog Devices And Requirement-Sensitive Service First
SLA	Service Level Agreement
SLOs	Service Level Objectives
SPP	Service Placement Problem
SJF	Shortest Job First
SNA	Social Network Analysis
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
TCP\IP	Transfer Control Protocol\Internet Protocol
VM	Virtual Machine
VNET	Voicenet
VFN	virtual network functions
VIKOR	VIšekriterijumsko K OmpromisnoRangiranje

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

ABSTRACT

In order to address the demands of contemporary technologies like the Internet of Things (IoT), Artificial Intelligence (AI), 5G, and more similar elements, Fog computing is working as an extended platform of cloud computing. The advancement of numerous application scenarios, including healthcare, smart cities, transportation, entertainment, and agriculture, which have a substantial impact on people's daily lives, is being facilitated by the IoT paradigm. These apps must have the processing and storage power to handle the massive volume of data prepared by IoT devices. IoT devices cannot effectively process and store significant amounts of data due to their inherent resource limitations. Therefore, IoT devices need substitute resources to ensure the efficient execution of their diverse applications, some of which may be computation-intensive or latency-sensitive. One of the potential resource suppliers for IoT devices is the cloud. Although it impacts the amount of time IoT devices are actively using energy. Subsequently, the usage of smart apps that respond instantly has increased significantly along with the use of IoT-enabled devices. Numerous problems are imposed by this increasing demand, including scheduling, pricing, server overload, etc. Fog servers, in contrast to Cloud servers, have resource restrictions that restrict them from running all IoT application types, notably those that require a lot of computing.

Fog servers, in contrast to Cloud servers, have resource restrictions that restrict them from running all IoT application types, notably those that require a lot of computing. Therefore, by storing the data on local Fog Nodes (FN), rather than adding to the load on the cloud, Fog Computing expands its services to include cloud computing. The two most pressing problems that Fog Computing has are Resource Allocation and Resource Management. The Fog computing paradigm as a result is very dynamic, distributed, and heterogeneous. Thus, it is challenging to fully realize the potential of this computing paradigm for various IoT-driven application scenarios without efficient scheduling approaches for the administration of IoT applications. As a result, it can minimize network congestion and speed up the delivery of application services. The major processing is done by Fog nodes which are heterogeneous and dispersed in nature. Whereas, some significant nodes have resource and spatial sharing limitations. Therefore, for various smart apps use cases it might be complicated to leverage its benefits without effective administration. The administration of computing resources includes the management of applications. By locating appropriate placement alternatives for the applications within the computer infrastructure, it may be made sure. The problems with

resource management are caused by resource heterogeneity and resource depletion. These problems have been thought to be significant concerns in the Fog environment. It appears that creating such resource management strategies in the Fog is quite challenging. In this kind of issue, multi-criteria decision-making (MCDM) strategies are very beneficial. Additionally, problems with resource management fall under MCDM challenges. One of the well-known MCDM methods that has been taken into account for selection and ranking is the Analytical Hierarchical Process (AHP). Selected Quality of Experience (QoE) criteria have been used to evaluate this resource management strategy. The suggested method helps in monitoring the Fog resources about their value and ranking.

Moreover, the matter of efficient resource allocation in the Fog layer is imposed by an increasing number of smart apps that are aware of the delay. For resource allocation and ranking, we have implemented an efficient MCDM-based solution in this study. The suggested algorithms incorporate the TOPSIS and AHP technique improved versions. This framework takes into account QoE parameters, such as network bandwidth, no. of cores, and average latency. The advised framework surpassedthe performance of the other existing methods when compared with different performance metrics. For the distribution of smart applications, a cost-effective scheduling strategy has been developed to address these issues. The purpose of this study is to maximize user benefits fromFog the environment while lowering thecost of smartapplications. The suggested framework was assessed using a test bed which consists of three analysis phases, and it is compared using five metrics: average allocation time, average Fog Environment profit, averagecost of smart apps,resource utilization, andnumber ofapplications running within a certain latency. Performance analysis shows that the used technique is performing better in all the criteria.

CHAPTER 1
INTRODUCTION

The term Quality of Experience (QoE) describes how satisfied a user feels when utilizing a specific system, application, or service. QoE and fog computing are closely related since the environment seeks to improve user experience by bringing high-performance, low-latency computing resources closer to the network edge. Enhancing Fog computing QoE is important for several reasons. QoE in fog computing is crucial for driving user satisfaction, achieving business success, gaining a competitive advantage, meeting application requirements, and optimizing resource utilization.

1.1 BACKGROUND

IoT has made several changes to the world of physical environment by interconnecting several computing components such as living and non-living things with the help of the internet. These devices help them to observe their surroundings as sensors and generate actions based on the commands using actuators [1]. Recently IoT has been used in a variety of fields, such as smart cities [2,3], smart homes [4,5], autonomous vehicles [6,7], smart energy [8,9], and healthcare [10, 11]. The architecture of IoT is shown in Figure 1.1 below. With the increase in data, the use of IoT has been rapidly increasing and has become an important part of our day-to-day lives. However, it is anticipated that by 2030, there will be over 1.2 trillion IoT devices in operation, with an annual economic effect of $15 trillion [12]. This enormous volume of data must be processed for the benefit of its services and requirements. Although, cloud computing as a platform is accessible with nearly unlimited resources to execute the IoT smart application. IoT-based smart applications go through issues such as latency and bandwidth in interacting with the cloud servers [13].

The data that is generated by the smart applications causes congestion on cloud servers [14]. Due to this reason, the cloud servers are not managed to fulfill the demands of the resource-intensive IoT applications. Specifically, the original nature of the cloud does not support the IoT device's decentralized nature. The Cloud data centers have high delays, geographically dense, network congestion, and poor QoS for faraway requests.

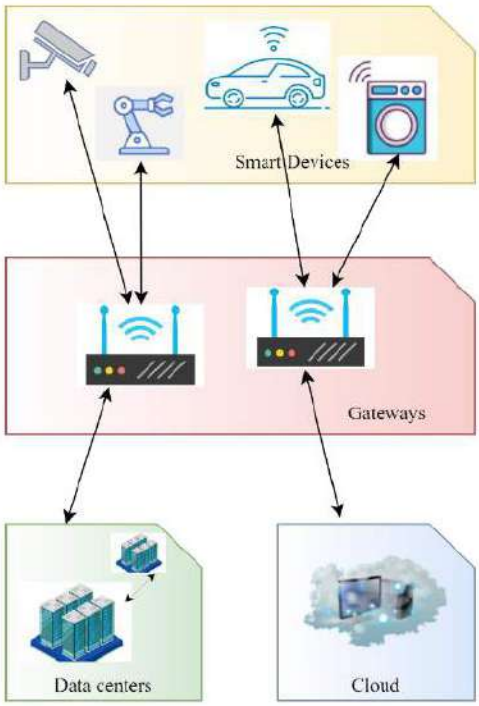


Figure 1.1 IoT System Architecture

The National Institute of Standards and Technology (NIST) states that cloud computing is a pay-as-you-go approach with a sizable pool of virtualized resources that scale workloads and are configured dynamically [15-16]. It also includes remote management of resources, data, and services, among other things. Hosting subscription-based resources and application services has been made possible by cloud computing. Applications for various IoT-enabled CPSs are also executed using it [17]. Data and computer servers are stored in cloud datacenters to provide users with storage and virtualized computing instances [18]. Traditional IT architecture includes its internal infrastructure that offers storage and backup services. Large machines are used to deliver this service, and the extensive infrastructure required to support it is challenging to maintain. To get around these issues, cloud computing was developed. It offers the infrastructure benefits like increased performance, dependability, scalability, etc. Users can employ the computing resources following their needs due to features like the on-demand service offered by cloud computing. Resource Pool in which a collection of resources is assembled to serve users. Access to the network is utilized to serve user requests for resources. The resources are increased or decreased according to user demand as the last step. The original purpose of cloud computing was to allow users to use computing resources from anywhere at any time. These cloud computing applications and

resources can be accessed quickly. The current advancement in processing power and equipment that produces a lot of data necessitates the presence of a cloud at the network edge. These geographically dispersed devices require low-latency data processing. However, when a large number of IoT devices start data-driven interactions with remote applications the network gets heavily loaded and severely congested. Additionally, it makes cloud datacenters' computational overhead higher [19]. Later Fog computing has been introduced to process and compute closer to devices to meet this demand with minimal latency.

1.2 FOG COMPUTING

A fog computing environment was introduced to satisfy the demands of IoT applications and control these constraints. Fog computing technology is placed between cloud and IoT devices which joins the core network to the edge network. Therefore, the process of utilizing the resources to process the data is called Fog computing [31-32]. The descriptive architecture of Fog computing is shown in Figure 1.2. The main working of Fog computing takes place in data centers which are geographically near to the users and are small. IoT-based smart applications with network connectivity, computing capabilities, and storage might be a Fog node. Fog computing includes a large number of heterogeneous Fog nodes that work in a group to supply the required resources [33]. Fog nodes now allow the IoT-based smart applications to be executed closer to the resources without any involvement of the cloud which leads to fewer communication delays.

The term "Fog computing" was coined by the CISCO systems in 2012 [27]. It is a different paradigm that might be useful in smart devices, the usage of automobiles, and sensors. In compliance with this model, the computing duties and job processing must be divided equally [20]. Instead of creating a single data network the network uses numerous devices. The network latency and bandwidth can be minimized by working from the end user to the cloud. Fog computing model processes the data produced by the sensors and then loads it to the cloud. This paradigm offers several amenities, including speedier connection, increased power of execution, and the ability to track and analyze IoT services [21].

The use of information technology (IT) is essential to everyone's life. The way people live has radically changed as a result of IT advancement. Significant IT resources, i.e. storage,

processing power, and network bandwidth, affect numerous industries, including healthcare, agriculture, cooperative banks, entertainment, and many more. Every day, there is more need for these IT resources. Numerous computing technologies, including utility computing, grid computing, parallel computing, and cloud computing, have evolved in response to the rising demand for IT resources. Cloud computing is one of these computer technologies that enable customers to execute services as needed and pay for them based on usage. On-demand services for computing resources are now possible as a result of cloud computing.

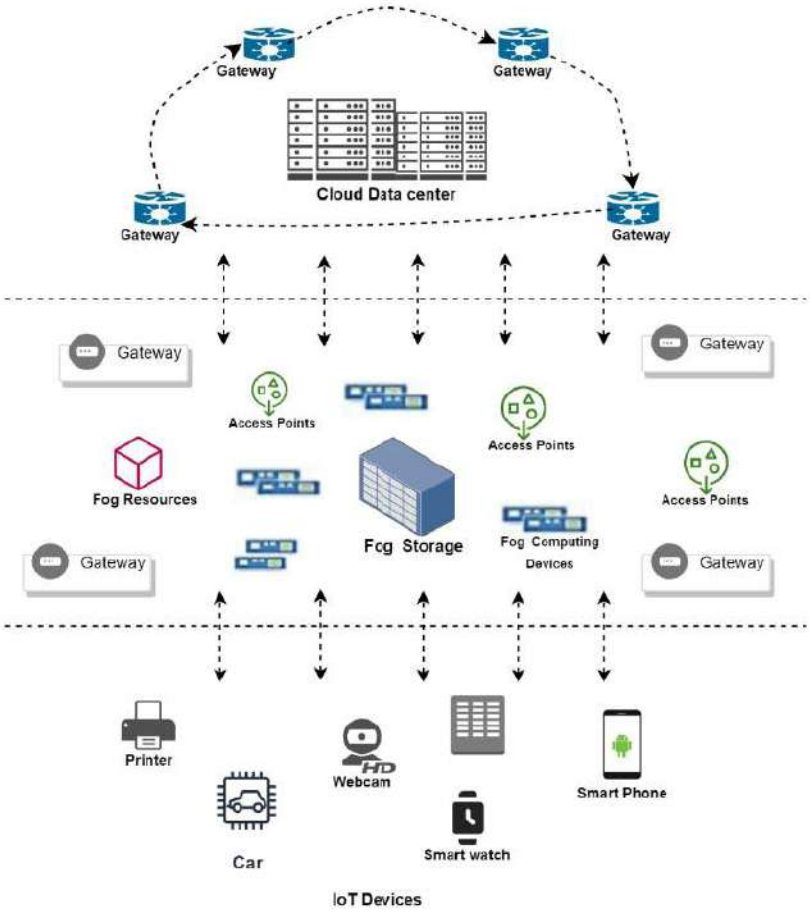


Figure 1.2 Fog computing descriptive architecture

IoT is expanding in a variety of public and commercial settings, necessitating adjustments to the current methodology. The foundation of all IoT services, cloud computing executes all IoT services in a centralized cloud. Cloud computing is unable to fulfill the specified requirements due to a lack of location awareness, excessive latency, and a lack of geo-

distributed data centers close to IoT devices. Due to the problem of latency, cloud computing may be the practical solution to meet the needs of distributed IoT-based applications. [1].

Information systems for Internet of Things applications that use a centralized, international approach and rely on remote management systems for IoT devices[1]. However, the model has a weakness in terms of agility. Users seek prompt replies in many real-time applications, including those connected to healthcare, ambient assisted living, and environmental analytics. Even when mobile internet speed has increased, the distant centralized model's latency is still relatively high. Fog computing offers data filtering with computers accessible at nearby data centers at the edge network of IoT systems and end-user apps to address this issue [27].

Since the foundation of Fog computing has been verified, the purpose of allocating the IoT services is still a complex issue in the Fog environment. The mechanism that is required for the placement of services issue should transfer services, delay, and reduce costs in the Fog computing environment. In Literature several strategies have been proposed to decode the service placement issue in the Fog computing environment [32, 34]. Whereas, seeing the significance of this problem many researchers are still working on this issue. Most of these issues aim to enhance the system performance and improve QoS, but the usage of Fog resources also needs attention.

1.2.1 GROWTH OF FOG COMPUTING

Fog computing as an evolving technology that provides reduced latency by enhancing cloud computing facilities to network endpoints [22, 116]. To facilitate communication and support decentralized system models multiple network devices carry out the computational operations. However, in distributed computing, several novel computation paradigms have advanced. In Fig. 1.3, seven distinguishable phases are depicted.

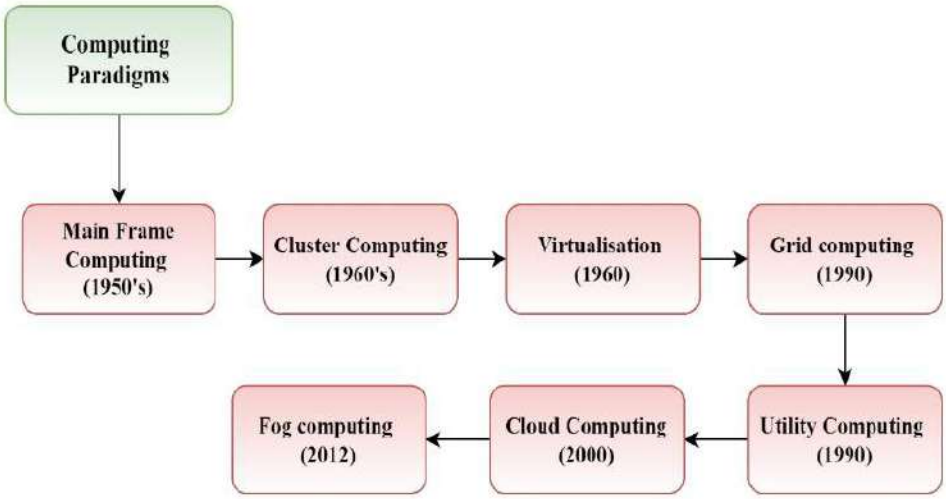


Figure 1.3 Growth of Fog Computing

Mainframe computing, which employs batch processing, is the first stage. For the examination of the influence of technology integration capacity, the mainframe environment was suitable[23]. Cluster computing was conceived in the early 1960s. Virtualization as a notion dates back to the late 1960s. In the 1990s, a computing paradigm known as "grid computing" and utility computing[24-25] evolved in which a grid of interconnected computers makes computational decisions collectively. Utility computing anticipated the concept of cloud computing. In the early 2000s, cloud computing [26] gained popularity. In 2012, fog computing—which incorporates computation via end devices including mobile phones, sensor boards, and control systems entered the computing environment.

Faster access could be provided to users with better involvement in Fog computing. Thus, to address a distinct range of applications the computing capacity of cloudlets has assisted the edge capacity of IoT devices [27]. However, the cloud and Fog work with cloudlets to help the applications. The development of all Fog computing applications in networked systems enables high-performance computing (HPC) [22].

All the data and processing migrate when users and devices transfer from one access point to another [28]. Be means of data migration users may easily retrieve their data in urgent situations. Whereas, in sensitive situations such as in healthcare and transportation systems delays might result in uncertain conditions [29]. In such cases as with time-centric applications quick access to resources is offered by the Fog computing paradigm.

The management of resources needs to be improved to achieve less cost, better performance, and reaction time. Therefore, applying Fog computing to a real case situation is a major priority. The processing of resources is complicated by the large volume, data velocity, and variety, which might have an impact on resource usage [30].

1.2.2 FOG COMPUTING ADVANTAGES:

- **LOW LATENCY:**

  Real-time processing is necessary for interactive services in critical applications. Robotic clouds, intelligent cars, for instance, are built using data collected by these devices' sensors. The cloud's control structure holds the generated data because processing and storing it could take a long time due to the volume of requests [84]. FC was established to support the cloud computing control system while keeping these devices close by and ensuring real-time delivered service. Placing the process close to the device reduces latency compared to the Device to Cloud (DC) architecture because the physical distance is less and the potential response time in the center may be avoided [85]. These can be less expensive than the Device Only (DO) architecture since computationally difficult tasks take longer on sensing devices with limited capabilities and can be moved to more powerful Fog calculation nodes. Making the delay predictable could also act as a motivator [86]. The majority of crucial applications need a quick response to process the data and make a choice. This kind of task is not appropriate for cloud computing. Fog Computing will therefore be helpful in this kind of situation [87]. It is necessary to support the quick reaction time and low response time, as well as to offer adequate processing and network infrastructure for various IoT applications, to fully appreciate the benefits of IoT. IoT applications with large amounts of storage and processing power are mostly provided by cloud computing [88]. IoT systems supported by the cloud, however, suffer numerous challenges due to their remote location from users. High response time, lack of mobility, and a significant strain on cloud servers are some of these issues. The low latency of Fog computing may also be advantageous for traffic security, online games facility, and, monitoring purposes according to Baccarelli et al. [89]. When data flow is established in the cloud, this shorter time is often moved to long-

term storage devices to minimize the amount of unprocessed data that a set of devices acquires. To gain knowledge and create a smaller data collection that is then saved, this information can be processed, filtered, and aggregated. Describe other devices as loop sensors as soon as possible. The Fog computation model can reduce network traffic from the edge to the data center in certain circumstances [90]. The overall advantages of Fog computing are presented in Figure 1.4 which are shown below.

- **ENERGY EFFICIENT:**

Fog computing could be used as a way to minimize energy usage compared to cloud computing, which uses a lot of energy. Numerous studies have shown that a device with a poor connection that generates static data for the edge user site consumes little energy. The duration of time the link is idle also affects energy consumption [91].FC, on the other hand, affects how much energy sensor devices use differently. First, the gateway can serve as a communication channel to lengthen the device's sleep cycle. The gateway handles all requests and updates while the sensor is in sleep mode and only begins processing when the sensor is enabled [92]. Second, the battery-powered nodes can be relieved of energy-intensive accounts and other services. Additionally, Fog nodes are widely dispersed throughout the spatial domain and are all wirelessly connected to the network. These doses lack a battery that can only be charged with sustainable energy sources like solar and wind energy. FC aims to lower networking and computing's energy usage by adaptively scaling the total pool of resources that are readily available on both a horizontal and vertical plane [93].

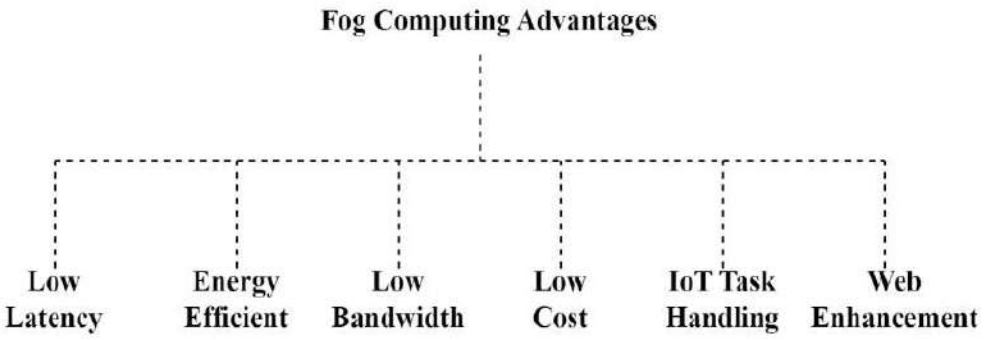


Figure 1.4 Advantages of Fog Computing

- **LOW BANDWIDTH:**

  In contrast to the DC architecture, Fog computing minimizes the volume of data transported to data centers. A smaller quantity of data is sent to the data centers after being processed through filtering, pre-processing, analysis, or compression [86]. Based on locally cached data, the local node can similarly respond to the device's inquiries, negating the need for communication with the data center [94]. In a related study, an FC reduced bandwidth use by implementing data processing services at the network's edges to reduce data flow from the edge users to the center [95]. Every function is used for a variety of tasks and was created to satisfy the demands of application service providers. To assess the framework's performance, they developed a structure that manages data compression between a cloud server and an end device. Results analyses revealed network infrastructure savings at the expense of an increase in latency. In the process of conducting picture compression for video surveillance, the designed prototype displays increased bandwidth when the scene doesn't vary from frame to frame. [96].

- **WEB ENHANCEMENT:**

  Website performance is improved with Fog computing. Fog nodes don't need to come and go since they can process, execute, and combine all of the HTTP request's contents—including redirects, images, and scripts immediately using FC through the web. In addition, users can be recognized using various cookies or MAC addresses, user tracking requests, and the knowledge of the state of the local network and cache files [97]. Websites can also contain feedback scripts that track how quickly a user's browser renders an image. The current user zone reception, wireless graphics resolution, and network congestion are immediately informed by this to the Fog node [86].

- **LOW COST:**

  The low bandwidth of fog computing results in reduced operational costs. Because FC architecture transmits less data to data centers than DC architecture, this is the cause. Instead of sending data to the cloud for review, FC analyses locally chosen data. FC also enables the merging of various platforms and physical settings between

numerous services. Additionally, FC uses less energy than cloud computing, which uses a lot [101].

- **IoT TASK:**

  The majority of IoT tasks, questions, and requests concern the environment. These inquiries and requests are handled via fog computing, independent of the cloud. For instance, a sports task application will keep tabs on nearby players of the same sport [33]. Another illustration is a smart car, which will track information at a hundred-meter distance. Fog computing assists in handling all of these types of local requests at the network's edge. Therefore, Fog computing is suitable for IoT tasks and queries.

1.3 MAJOR ISSUES IN FOG COMPUTING

Based on its functionality, fog computing has three main application areas:

- **FOG NETWORKING:**

  The fog network is diverse since it is located at the Internet's edge. The fog network is accountable for connecting every element of the fog. It can be difficult to run such a network, ensure connectivity, and provide services on top of it, especially in large-scale IoT scenarios. To create adaptable network environments that are easy to maintain, solutions like software-defined networking (SDN) and other one-network function virtualization (NFV) are considered. The combination of SDN and NFV helps streamline deployment and maintenance and minimize cost in many phases of fog computing, including allocating resources, moving VMs, observing traffic, controlling with application awareness, and configurable interfaces. The major issues are shown in Figure 1.5.

- **INTERFACING AND PROGRAMMING MODEL:**

  To make it simpler for developers to port their programs to the fog computing platform, there is required a programming and interface strategy. First, environment components will be application-aware and enable the right optimizations for different applications. Second, to build interoperable apps across several platforms, developers must organize dynamic, hierarchical, and heterogeneous resources. For large-scale,

regionally dispersed, latency-sensitive Internet applications with on-demand scaling, the Hong et al high-level programming model is proposed [22]. Their plan, however, is built on a tree-based network structure with set positions for fog nodes. Therefore, more generalized techniques are required for varied networks.

1.5 Issues of Fog Computing

- **COMPUTATION OFFLOADING:**

Offloading can assist mobile devices overcome resource limitations since it can improve the performance of programs, save storage, and increase battery life for particular computation-intensive operations. Six metrics can be used to categorize current compute offloading efforts for mobile cloud computing: goals, scheme granularity, adaptation, communication, and distributed execution[12]. Handling dynamics is the most difficult offloading problem in fog computing. Three things to determine the dynamics:1) Access to wireless and radio networks is extremely dynamic, and 2) The very dynamic nature of the fog network's nodes. There are a lot of moving parts and resources in the fog. Device-fog cloud is a three-layer design that the fog and cloud federation genuinely show us. There are new opportunities and problems for computation offloading in such infrastructure. Questions include how to dynamically divide an application for offloading on fog and cloud, and also how to select the right granularity for offloading at various levels of fog and cloud hierarchy.

- **ACCOUNTING AND MONITORING:**

  Without a viable business model, fog computing cannot be successful. The following entities can be fog computing providers, according to recent studies and proposals: 1) Wireless or Internet service providers that can build fog around their infrastructure. 2) Cloud service distributors who seek to extend their cloud service to the edge network. 3) End users who need to exchange their unused computing capacity for storage in their local private cloud to lower the ownership cost. So many problems need to be solved before "Pay-as-you-go" is implemented. It is required to determine the method of determining the cost of various resources & how much of the bill goes to which party of the fog, for instance, when it comes to billing. Hence, accounting and monitoring the fog at various granularities to enforce those pricing principles is required. It's also intriguing how the use of dynamic pricing for cloud computing services increases income and utilization, just like old industrialized industries do with hotels, car rentals, and airline tickets [25, 54].

- **RESOURCE MANAGEMENT:**

  In Applications-aware provisioning, metrics like bandwidth, storage, compute, and latency will change dynamically, and the mobility of the end node presents issues. For instance, in a linked vehicle situation, people could monitor a working ambulance, program smart traffic lights to ensure a green light and alert all the close cars to clear the road. Provisioning must be performed getting the resources ready to deliver service mobility to satisfy QoS requirements like latency. According to Work presented in [37], a method for deploying and migrating both fog and cloud services is known as MigCEP. Prioritizing operator migration assures end-to-end latency constraints and lowers network utilization. Fog computing, which offers application-aware provisioning, is what it is believed will let mobile crowdsourcing and sensing apps succeed with the IoT. Resource sharing and discovery are essential for application performance in the fog. The proposed work in [28] suggests a strategy for dynamically choosing centralized and flooding solutions to conserve energy in heterogeneous networks, but fog computing has additional limitations due to latency and mobility. A methodology to share heterogeneous resources in fog computing is

proposed. Takayuki et al. [34] by representing heterogeneous resources like CPUs, bandwidth to time resources. It is possible to frame resource-sharing optimization problems to maximize either the sum or product of utility functions that are service-oriented. The utility function, which can be expanded to incorporate variables like service availability, energy usage, or even revenue, focuses only on service delay. To ensure QoS and minimize energy waste, IoT systems need resource management policies. To foster innovation and development in fog computing and enable real-time analytics, an assessment environment is required to investigate various resource management and scheduling solutions [34]. Real-time IoT applications frequently cost too much and do not provide a testable and controllable environment. The cloud will function in a fog-aided cloud framework with the assistance of Fog nodes.

- **SECURITY AND PRIVACY:**

  When operating in more delicate circumstances, fog frequently goes to the desired location of the user and can offer the finest response to the client's needs. The ability to tackle new security challenges is made possible by the proximity of fog nodes and end users [35]. Fog systems are capable of managing local security monitoring, threat detection, and threat protection on behalf of endpoints. Additionally, fog nodes can assist with managing and updating end-user security credentials, removing the requirement for all endpoints to communicate with a remote cloud for these purposes.
  Privacy Nowadays, users are worried about the possibility of their personal information (data, location, or usage) being leaked online. Many scenarios, such as the cloud [48, 4], smart grid [40], online social network [35], and wireless network [39] have called for the use of privacy-preserving approaches. Since processing and storage are ample on both ends of the fog network, privacy-preserving methods can be used there instead of at the end devices where they are typically resource-constrained.

1.4 RESOURCE MANAGEMENT

In Fog computing, resource management services comprise application placement, resource provisioning, allocation, and scheduling. Resource allocation has been a major task in the Fog computing environment due to the increase in data generated by smart applications [35]. In

the Fog computing environment, several researchers have applied various techniques like PSO, GA, and Fuzzy logic [36] to allocate resources to the applications. However, these evolutionary algorithms have a high convergence rate and result in poor performance of the smart applications. In Fog environment resource allocation is a complex problem that lies under the category of MCDM problems. MCDM methods are useful in grid and cloud computing [37-38]. In the computing environment, the integrated paradigm is complex and dynamic [39] which results in being a challenging task for the providers. Although approaches based on prediction to examine the resources have been considered as well [40]. This approach uses the relinquish probability system. Other authors have presented that for future prediction of various loading environments, the probability of stored resources could be used [41]. In this situation, the output will be evaluated when the load is acquired accurately. Also [42] worked in the hybrid system of cloud and Fog with an allocation approach for diffusing the workload. The factors such as power consumption and delay have resulted in the evaluation of resource distribution. Whereas, for distinct Fog and cloud architecture, connected resource allocation models may be used [43].

It is clear from the proposed models that these papers only predict the outcome for the different approaches used for assigning the Fog computing applications. Major contributions in the improvement of required resources and the Fog computing applications could be used for better analysis. Moreover, the allocated resources which are dealing with the certain needs of the Fog computing would be trained to deal with the whole Fog computing environment [44]. However, by dealing with the system requirements such as its RAM and storage requirements and resource capabilities the issue of low latency and network delay can be improved.

The scheduling method is used to shorten the time a work takes to complete overall. A scheduling method aids in efficient resource allocation and work scheduling. Scheduling is a novel notion in fog computing technology, and there have been very few studies conducted in this field. Scheduling mechanisms are essential to the fog computing process. Similar to cloud computing, fog computing uses resource allocation to distribute available resources to clients online. Due to the dynamically changing resource availability in fog, scheduling is a difficult task. A scheduling policy aids in the effective and efficient use of virtualization machines. For quick processing, storing confidential data, and retrieving knowledge over the internet, fog computing uses a pool of virtualized computing resources close to the user's end.

Effective scheduling is necessary to maximize the use of all resources and deliver advantages to the fog providers. To improve performance, many scheduling strategies have been adopted. Reduced latency, increased energy efficiency, an active network, and geo-distribution are further benefits of these strategies.

1.5 Quality of Service (QoS)

Error rates, bandwidth, and latency are key performance indicators for measuring QoS, which is a word frequently used to describe how a network operates. ITU has delivered the Quality of Service (QoS) definition [10]. Utilizing both subjective and objective measures of customer satisfaction, quality of experience (QoE) measures the overall system performance. It is distinct from quality of service (QoS), which estimates how well hardware and software services are performing when provided by a vendor following the terms of a contract.

1.5.1 QoS Parameters

Many elements, which may be further divided into two groups like technical and human factors, have an impact on QoS in packet-switched networks. Customer information, wait times, and service quality stability are all examples of human factors. On the other hand, technological concerns including scalability, dependability, network congestion, and maintainability [12] are important. Here are a few factors that affect the quality of service extremely effectively.

- **Latency:**

  A packet may take a more indirect route to avoid congestion or take substantially longer to reach its destination as a result of being held up in lengthy queues, or other circumstances, excessive latency may be detrimental to an application like VoIP or online gaming [11]. It is possible to refer to the total amount of time taken for a signal to get from one location to another, often from a transmitter across a network to a receiver. The fact that the data packet spends so much more time in the queue due to network congestion is also upsetting.

- **Packet Loss:**

  Packet loss is the act of getting rid of data packets when a network device, like a router or switch, is overloaded and unable to take any incoming data at a specific time[14]. TCP/IP and other higher-level transport protocols, however, guarantee that the data transmitted during transmission is correctly received at the other end.

- **Bandwidth:**

  A network connection can move a larger volume of data quickly from one goal to another. By controlling bandwidth and giving priority to apps that use a lot more resources than others, QoS maximizes the network's potential.

- **Capacity:**

  The two main factors that must be taken into account when determining capacity are network bandwidth and storage capacity. Knowing where the data are situated inside the fog network is crucial for achieving these two features, hence data localization is crucial. This becomes a significant obstacle for computing paradigms [13].To conserve network capacity and minimize latency, the cache should also be reconfigured to make use of temporary locations and larger coverage.

1.5.2 Quality of Experience (QoE)

QoE has been explained by organizations like the European Telecommunications Standard Institute (ETSI) along with the International Telecommunication Union (ITU-T). These QoE definitions have been embraced by several application fields, including mobile applications, video conferencing, and multimedia IoT. The specific QoEdefinitionsare also provided by the ITU [15], and ETSI [18].

The ITU-T definition includes a subjective evaluation by the end user of the application or service used. In the diversity of the ITU-T definition, the ETSI definition of QoE incorporates both a subjective and an objective evaluation to complete the QoE definition's blueprint. Subjective human factors include things like usefulness, need, availability, needs, emotions, happiness, expectations, wishes, brand image, contentment, etc. However, quantifiable human characteristics include reaction time. The best way to determine a user's viewpoint is through subjective evaluation. It is a difficult method, though, and it falls short of real-time applications [23]. However, all of the current definitions of QoE put the end-users (humans') comprehension and measurement of the quality of apps and services at the forefront, frequently from the standpoint of multimedia applications. However, the QoE is assessed using the arbitrary input given by people. For instance, in [30], writers explain the

IoT network as an amalgamation of three factors, namely system, content, and service, using the ITU-concept T's of QoE. IoT content is the data that it transmits. They do not, however, directly assess the effectiveness of the system, service, or content in their work.

1.5.3 QoE Parameters

The ITU-T definition for QoEabout IoT is inadequate. The European Telecommunications Standard Institute and Qualinet White Paper's definitions of quality of experience (QoE) cannot be combined with the IoT in its most basic form because their applications involve device-to-device communication and there may be zero human involvement as end users to provide their response on the application. Among the existing QoE definitions currently in use, not a single theory aims to assess the quality of an IoT application analytics, sensing, and activation. Additionally, these definitions focus on human feedback and exclude machine experiences, making it impossible to gauge the quality of such IoT apps. The concepts of QoS and QoE make clear how fundamentally different they are from one another. However, there are situations when users' expectations for improved QoE might aid system services in enhancing their QoS [140]. For instance, a user can anticipate minimal buffering when watching online information on a fixed-fee Internet-enabled system. The network service distributors can assign adequate bandwidth and sustain acceptable jitter to improve the user's QoE concerning system, which would greatly boost the QoS of the associated service.

- **Latency:**

  The time required to communicate player data from the application layer on the client and server respectively is known as latency. The one-way delay would be as described. The round trip time, sometimes referred to as "ping" by computer players, is obtained by taking into account both the direct and indirect paths (from server to client). When reporting ping values to users, many games (if not all) take into account their critical interest when playing online games. Typically, it is the sole network parameter that the user sees.

- **Packet Loss:**

  Another network issue that might be seen is packet loss. In an IP network, various factors can stop a packet from reaching its end goal. Although they use various retransmission mechanisms at lower levels, wireless networks have a higher packet loss rate than cable networks. Buffers are a major contributor to packet loss in wired

networks; they reject packets when they are full, but some Active Queue management policies, such as Random Early Drop, might do so based on statistical probabilities even when the buffer is not full.

- **Bandwidth:**

  Even while a lack of bandwidth doesn't always cause a network problem, it might result in significant packet loss, jitter, and latency. Additionally, background traffic, or bandwidth consumption, is a factor that must be considered because playing online games is not recommended on a severely overloaded network. Cloud games need a huge scale of bandwidth, particularly in the downlink, in case the connection service can't supply it, the video quality must be decreased, which in turn lowers the subjective quality. The alternative may be to experience packet loss, which significantly reduces the quality of the video streaming, and consequently, the game as well.

1.6 MOTIVATION

Regularly a massive volume of data is generated by IoT devices and delivered to the cloud for processing. The volume of IoT devices is increasing and the statistical analysis showed that this number would increase up to 75.44 billion by 2050 [45]. This would lead to a new generation of objects with sensors in them. Real-time data that is generated by the sensors is very essential and needs to be analyzed. Also, a proper decision needs to be taken within the required time. It has been estimated by the Cisco Global Cloud Index that the amount of data generated reached 500 Zetta bytes in 2019 by things, machines, and people [46]. Generally, because of the storage power and computation linked with the data centers of the cloud [47], the data is transferred to the cloud for processing. The issues with this technique are network latency, lack of security, and congestion. Fog computing is known as a horizontal architecture that assigns the services of networking, control storage, and resources to smart applications. Fog computing enables applications that need to be distributed and services.

Several real-time applications require latencies below a few tens of milliseconds that make the cloud unsuitable to fulfill the needs of the users. Similarly, at the user end, a lot of Fog devices are present. If a nearby Fog device executes a task, then the user may need not worry about the latency issue, security, and cost [48]. Although the same amount of storage power

and processing speed as the cloud cannot be provided by the Fog device. Therefore, there exists a mutual understanding between the cloud and Fog for allocating resources and scheduling tasks. This situation leads to a need for such a model in a Fog computing environment which aims to handle the matter scheduling and resource allocation. The issue of heterogeneity causes a challenging problem among the Fog devices which can be solved through multicriteria decision-making (MCDM) techniques.

1.7 OBJECTIVE

The main purpose of this research work is to create a suitable model for the allocation of Fog environment the smart apps based on its QoE parameters. To attain the aim of this study, MCDM methods are used. Figure 1.6 signifies the relation between all three objectives.

- To create a QoE-based resource allocation framework and scheme for the Fog computing environment.
- To design an MCDM-based technique for managing resources in the Fog computing Environment
- To design a cost-effective scheduling technique for a Fog computing environment.

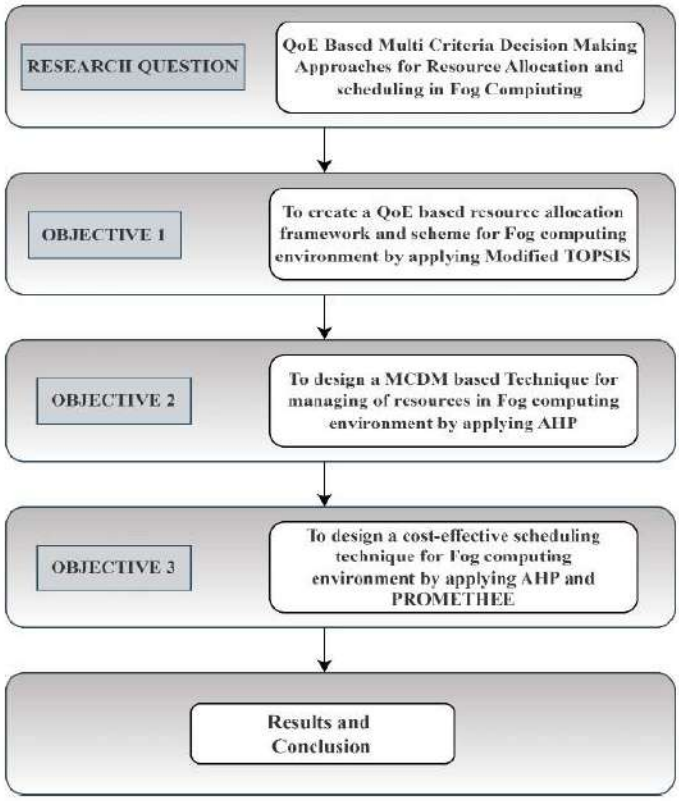


Figure 1.6 Objective Relationship Diagram

1.8 RESEARCH STRATEGY TO ACHIEVE THE OBJECTIVES

Objective 1: To create a QoE-based resource allocation framework and scheme for Fog computing environment.

1. The ranking of the fog environment was done using a synthetic dataset for the proposed framework's evaluation.
2. A modified MCDM model has been used for the evaluation of the Fog computing environment.
3. The performance is analyzed based on five metrics: resource utilization, reaction time, waiting time, available resources, and completion time.

Objective 2: To design an MCDM-based technique for managing resources in the Fog computing Environment.

1. A synthetic dataset was generated for the Fog environment and smart applications.
2. QoE parameters were taken into account when assessing the Fog computing environment.
3. The scheduling of smart applications in a fog environment is done using the AHP MCDM method.

Objective 3: To design a cost-effective scheduling technique for Fog computing environment.

1. Cost-based scheduling of the Fog environment is done using QoE parameters like uplink and downlink bandwidth and latency respectively, as well as RAM requirements and storage requirements, number of cores, time to finish, and cost.
2. For the cost-based scheduling, the Modified PROMETHEE-II technique-based framework has been used.
3. Metrics like allocation time, average application costs, average Fog environment profits, resource utilization, & average number of

applications running within a certain latency are used to measure performance.

1.9 THESIS OUTLINE

Chapter 1: Introduction

This chapter discusses the research background and motivation for pursuing this topic further for better results.

Chapter Literature Survey

This chapter gives a better understanding of the work that has been carried out by many researchers for Fog service placement using several approaches.

Chapter 3: Resource Mapping and Ranking in Fog Computing

This chapter discusses the placement of the Fog environment with the usage of the Modified TOPSIS technique in the proposed framework.

Chapter 4: AHP-based Technique for Resource Management in Fog Computing Environment

This chapter addresses the issue of managing Fog resources with the help of the AHP technique and providing those organized resources to smart applications.

Chapter 5: QoE-based Cost-Effective Scheduling in Fog Computing

This chapter discusses the proposed model using the Modified PROMETHEE-II technique. The evaluation is done by taking into account several QoE parameters. The performance analysis is also discussed in detail in the proposed model.

Chapter 6: Conclusion and Future Work

This chapter concludes the work and also highlights its applications.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

A large-scale allocation of platform and infrastructure services is established by the Fog computing environment. Networking and storage resources and on-demand need for computing are provided by the infrastructure services. Whereas programming interfaces, application runtime environments, and operating systems are assisted by platform services [49]. Administrative operations like virtualization, deployment, and monitoring of Fog nodes are symbolized by the Fog resource management that facilitates the platform services and infrastructure based on Fog [50]. In addition, the Fog resource management perceives dynamic provisioning, load balancing, and auto-scaling to assure multi-tenancy and availability of service [51].

Effective Fog management helps the IoT-enabled systems manage multiple applications at once. The characteristics of these systems vary from system to system. For instance, compared to a system that evaluates environmental characteristics, the delivery time of a healthcare system and a traffic monitoring system are both rather harsh [52]. Comparatively speaking, the systems that operate virtual reality operations control a vast amount of data to those apps that assist in locating available parking spaces.

These different features act as an important part of determining the QoS needs of the applications that are difficult to encounter via Fog resource management [53]. This understanding drives to design of some application management strategies following the priorities of the applications. Generally, a strategy for application management signifies algorithms, empirical analysis, recommendations, and mathematical models that manage the installation, execution, and implementation of applications in a computing environment. The review of the literature is split into two sections. The First section consists of four subparts to provide a better understanding of the research work and the Second section describes the literature related to MCDM techniques.

2.2 OVERVIEW OF RESOURCE MANAGEMENT

The Literature review describes the resource management issues. This section is divided into four sub-sections that are application placement, resource allocation, resource scheduling, and resource provisioning. Application placement is the first category, and it directly affects how much hardware and network resources are used [54]. For instance, improperly deploying a data-intensive application in a distributed fog environment could cause network congestion [55]. The second category is resource allocation, which is utilized to distribute the right number of resources among the cloud nodes and fog nodes' accessible devices [56]. Resource scheduling falls under the third category and assists in allocating requests to the relevant resources and services [57]. The scheduling in a Fog environment is divided into near, far, and collaborative scheduling based on the application deployment techniques, which aids in meeting IoT demands [58]. Resource provisioning, the sixth category, involves scaling up or scaling down the available resources to upgrade the fog resources in terms of energy, cost, time, etc [59].

A review of some prominent papers is done which includes resource allocation, resource provisioning, application placement, and resource scheduling in the following subsections. Subsection 2.2.1 includes significant papers related to application placement, subsection 2.2.2 consists of papers related to resource allocation, 2.2.3 includes papers that focus on resource scheduling, and 2.2.4 comprises papers that have performed resource provisioning.

2.2.1 APPLICATION PLACEMENT

The issue of task distribution in Fog computing can be resolved by placing the applications based on the upcoming Fog nodes processing commitment. Likewise, the strategic placement of applications could be an important factor in normalizing the cloud and Fog integration. However, when placing the application, the resource status is reviewed properly. This review is very helpful in dynamically updating application architecture and ensuring application maintenance. Skarlat et al. [60] presented a conceptual framework to explore and enable the usage of Fog-based computational resources. The authors formulated the problem of IoT application placement as an optimization problem that is known as FSPP. This optimization problem was solved by applying various approaches like a genetic algorithm, a greedy first fit heuristic, and an optimization method.

Later the experimental results were compared by executing the same setup in the cloud environment. The proposed architecture would be improved by fault tolerance techniques to report for mobility inFog environment. Yu et al. [61] worked on the problem of single and multiple application placements that sustain the data streams with delay and bandwidth assurances. The authors formulated the problem of single and multiple application provisioning as NP-hard. In this paper, a fully polynomial time approximation method has been proposed for single application placement. Whereas in the case of multiple application scenarios, a full polynomial time approximation scheme was proposed if the applications were parallelized among various distributed instances.

However, for non-parallelizable applications, a randomized algorithm was proposed. Simulations of the proposed schemes improved the QoS of the IoT applications. Yadav and Baranwal [62] designed a trust-aware application placement policy to place the IoT applications at best trustful nodes. The authors have measured trust by social relations and QoS through Fog nodes. The authors also focus on the fact that the node is available or not before the placement of the application. In this paper, a detailed case study has been provided that explains the proposed trust model.

Kumar et al. [63] established a computational framework by integrating AHP and TOPSIS techniques to decide the suitable cloud service. The authors used AHP for two purposes first to calculate the criteria weights and then to define the architecture to decide the best cloud service. Whereas, the TOPSIS technique was used by authors to evaluate the final rank of the cloud service based on its overall performance. Later a real-time case study has been described by comparing the presented scheme with other existing works. Lastly, the efficiency of the proposed methodology has been testified with sensitivity analysis.

Triantaphyllou and Triantaphyllou [189]gave a simple example of sensitivity analysis related to MCDM methods. For some operational research and management science models, such as linear programming, inventory models, or investment analysis, there is a significant amount of research into sensitivity analyses. Nevertheless, there is very limited research in the area of sensitivity analysis for determinative MCD models. This paper provides an overview of the relevant literature.

Erkut and Tarimcilar[190]in their paper explored the different ways to perform sensitivity analysis. The AHP is a widely used multi-objective decision analysis tool. Using sensitivity

analysis, which is an important extension of AHP and has not been studied extensively, the authors describe several ways to improve decision-making processes.

Kabir and Hasin [191] in this paper, integrated the fuzzy analytic hierarchy process (FAHP) with the Method for order preference based on similarity to optimum solution method for developing better and more adequate power substations Location assessment and selection model taking into account social, technological factors, Factors, and sub-criteria in terms of economics, the environment, and site characteristics. To discuss and explain the results, a numerical example is presented to demonstrate the applicability and performance of the proposed methodology, followed by a sensitivity analysis.

Kabir and Sumi [192] presented an easy, systematic, and logical scientific approach in the present paper to assess power substation location by integrating the Fuzzy Analytical Hierarchy Process (FAHP) with the Preference Ranking Organization Method for Enrichment Evaluations (PROMETHEE). By eliminating the limitations of the FAHP and PROMETHEE methods, the proposed integrated approach provides more realistic and reliable results and facilitates the decision-maker to make multiple conflicting decisions. The proposed model is implemented in a power substation location selection problem in Bangladesh.

Baranwal et al. [64] developed an application placement policy to improve the QoE of the applications. The application placement policy proposed in this paper executes on a Fog node. The proposed policy is developed using the TOPSIS method to determine the ranking of the applications. Based on these rankings the applications are placed. Although the computational complexity is low in contrast to the other existing algorithms in the literature. The authors considered four performance metrics to measure the proposed work performance.

Mahmud et al. [36] designed an application placement algorithm in which Fog considers the location of the Fog instances and the expectations of the user meanwhile placing the applications. The authors have used two fuzzy logic models to clarify the application mapping. The designed linear optimization problem guarantees the best convergence among the scope in the Fog environment and its user expectations. Experimental results indicate that the proposed policy has performed better than the other existing policies.

Baranwal and Vidhyarthi [65] introduced an orchestrator node for the application placement policy in the Fog computing environment. The authors divided the Fog node into two groups that are Fog gateway nodes and computational nodes. The main objective of this distribution of nodes was to select an appropriate Fog gateway node as an orchestrator node to improve

the overall IoTperformance. The proposed model has been integrated into one of the states of artwork that has improved the performance of the IoT system.

Souza et al. [66] examined the service placement strategies in IoT considering the different computing paradigms such as cloud, Fog computing, and cloud Fog environments. The objective of this paper was to explore the advantages of latency in F2C and Fog computing. Later on, the authors presented a design principle as distributed and parallel execution with high QoS. In this paper assessment of distributed service placement in F2C was performed by applying different strategies. The results in this paper show the advantages in terms of network care load and service response time, of distributed execution in F2C resources.

Taneja and Davy [67] presented the utilization of resources in the network through organizing the application modules in the Fog cloud infrastructure. The authors worked on the issue of latency in real time applications that are caused due to the heavy usage of IoT in users' day-to-day life. The complexity of the proposed module mapping algorithm outperforms brute force solutions in case of an NP-hard problem. The authors believed that the experimental results of this paper could help the researchers working on IoT and Fog applications.

Mann [68] examined the impact of coordination and decentralization in the assignment of the Fog application. In this paper, two Fog application placement algorithms were executed along with four different coordination and decentralization models. The comparison is done with the help of extensive experiments, which concluded that on easy and small problem instances centralized approach achieves good results. The result of the centralized approach is in contrast with the decentralized approach when the number of components was increased.

Mahmud et al. [69] designed a latency-aware application management approach that takes into account the latency of service delivery of the volume of data signals that must be processed per unit time for multiple apps in a Fog environment. The goal of their approach is to guarantee the application's QoS while meeting service delivery targets and to maximize resource utilization in a fog environment. In the simulated fog environment created by iFogSim, the suggested policy is modeled and assessed. The simulation studies' findings show a noticeable increase in performance compared to other latency-aware techniques. Gu et al. [70] worked on the integrated platform FCMCPS to reduce its overall cost and maintain the QoS requirement. The authors worked on the cost efficiency problem by a dispute that VM deployment, task distribution, and BS association are all critical. In this paper, the

problem is expressed as a MINLP problem and then further it is linearised into a MILP problem. Also, proposed a heuristic algorithm based on two-phase LP. The evaluation results demonstrate that the proposed algorithm has a substantial advantage over the greedy algorithm.

Yang et al. [71] created a group of algorithms to reach the goal of maintaining a trade-off between the service provider cost and the mobile user's average latency. The proposed solution uses the service access pattern and user mobility pattern to anticipate the user's upcoming requests. Based on this prediction then modify load dispatching and service placement. The experimental results achieve low latency and perform better in terms of algorithm running time and cost than other classical algorithms.

Benamer et al. [72] primarily focused on deploying modules on fog nodes and reducing the application's total latency. The authors offer both precise and approximative solutions to the module placement challenge. Both CPLEX and an iFogSim-simulated fog environment were used for the experiments. The outcomes demonstrated the potency of our chosen strategy. To guarantee a lower overall latency for the entire program, the placement of the IoT modules was evaluated. The suggested solutions seek to determine where each module's best decision should be placed. The total latency of all selected nodes is also taken into consideration in addition to an inter-node latency.

Xiong et al. [73] demonstrated that the ideal policy for a single-service MDP has an alluring threshold structure, and based on the theory of the Whittle index policy, explicitly constructs the Whittle indices for each service as a function of the number of client requests. Therefore, effective learning augmented algorithms are created that fully leverage the structure of optimum policies with minimal learning regret because request arrival and service delivery rates are typically uncertain and sometimes time-varying. The outcomes of the simulations demonstrate the good empirical performance of the suggested policies.

Naranjo et al. [54] introduced a FOCAN, a multi-tiered structure in which the applications are executing on things that collaboratively compute, route, and interact with one another through the smart city environment, as a Fog-supported smart city network architecture. FOCAN reduces latency while enhancing energy supply, service efficiency, and other things with various capabilities. Specifically, three kinds of communication—inter-primary, primary, and secondary—are specified amongst FOCAN devices to conduct applications in a way that complies with the QoS requirements for the IoT. The major benefit of the suggested

architecture is that the devices can deliver the services effectively and with less energy utilization. Results from a simulation for a particular case study show the enormous influence of the FOCAN energy-efficient solution on the communication capabilities of many sorts of devices in smart cities.

Tran et al. [74] proposed multitier fog computing architecture that assists in IoT service provisioning. In particular, a reliable service placement technique has been developed that maximizes service decentralization on a fog landscape by utilizing context-aware data on the resource consumption, location, and response time of services. The experimental outcomes demonstrate the effectiveness of the suggested strategy using simulated data and data summarizations from service deployments in real-time applications, specifically the ITS in Ho Chi Minh City that the authors had built.

Yao et al. [75] examined the most cost-effective way to install the servers without compromising the expected level of service. The authors specifically take into account the fact that the cloudlet servers that are now accessible are heterogeneous, meaning that they have varying costs and resource capacities. Integer linear programming is used to formulate the issue, and a simple heuristic algorithm is developed to solve it.

Venticinque and Amato [76] created and applied a technique to solve the Fog Service Placement Problem, which entails determining the best match between apps and compute resources. To examine the ideal deployment of IoT applications, they took advantage of and expanded a Fog model from the literature. The IoT application in the case study is for smart energy. Their latest features allowed customers to use the platform more efficiently and enable automatic learning of energy profiles, but they also created new computational resource requirements.

Souza et al. [77] considered various computing paradigms, like fog computing, cloud computing, and their combination, known as F2C Computing while analyzing service placement methods in dynamic IoT situations. The main objective was to examine the advantages of the newly suggested fog and F2C computing, which combines sharing opportunities and unique collaborative models with latency-constrained assistance in fog. Further, expanded on the placement problem and execution of service by leveraging the concept of service atomization and conceptually proposing parallel and sequential service execution solutions after reviewing the state-of-the-art models.

Selimi et al. [78] used a quick heuristic approach, which is essential to swiftly respond to changing conditions, it is suggested to use network status information to advise service placement decisions. To assess its effectiveness, their heuristic approach is compared with one that uses random placement in Guifi.net, which is the largest CN globally. They measured the advantages of the suggested heuristic on a genuine live video gaming service and showed that video chunk losses were considerably reduced, achieving a 37% reduction in the packet loss rate.

Khosroabadi et al. [79] produced a heuristic technique to work on the Service Placement Problem (SPP)referred to as the clustering of Fog devices and requirement-sensitive service first (SCATTER). To confirm the viability of their proposed algorithm by taking into account a smart home application, they also presented simulations using the iFogSim toolbox and practical assessments using real hardware. The authors compared the SCATTER with two previous studies in terms of QoS metrics: edge-ward and cloud-only approaches. The test findings have shown that the SCATTER technique performs better than the existing approaches.

Xiao et al. [80] provided a federated learning-based intelligent F-RANs framework that can effectively protect user privacy because it does not require collecting user information centrally on the server for training. In this paper, federated learning is used to forecast user demand, which is an accurate indicator of the distribution of content popularity in the network. The proposed caching method effectively deploys caches and caches content. The ILP model's scalability is poor as the size of the problem grows because of the high computational complexity of their model.

Mahmud et al. [81] advised a profit-conscious application placement policy for integrated Fog-Cloud settings. When placing applications on computer instances, it is designed using a constraint Integer Linear Programming (ILP) standard that simultaneously increases profit and assures QoS. The effectiveness of the suggested policy is assessed using iFogSim in a simulated Fog-Cloud environment, and the findings show that it performs better than other placement strategies in simultaneously raising provider profit and user QoS satisfaction rate. The above-mentioned approaches have been listed in Table 2.1.

Table 2.1: Application Placement Approaches

Authors	**Technique Used**	**Results**	**Limitations**
Skarlat et al. [60]	Proposed a GA-based technique for service placement in Fog.	Experiences a lower deployment delay with the help of cloud resources.	Many Time-consuming approaches are applied such as GA, greedy first fit algorithm, and optimization method.
Yu et al. [61]	Fully polynomial time approximation method and randomized algorithm.	Simulations improved the QoS of the IoT applications.	The authors did not fully explain how network provisioning decisions are made based on QoS requirements.
Yadav and Baranwal [62]	Designed a trust-aware application placement policy.	Removes the malicious Fog node before the placement of the application.	The authors do not address issues with reliability, diversity of trust sources, and resistance to collusion, which could affect the system's overall level of trustworthiness.
Kumar et al. [63]	AHP and TOPSIS Techniques	Sensitivity analysis is performed to verify the proposed approach.	Performance measures not considered.
Baranwal et al. [64]	Modified TOPSIS Technique	Computational complexity is much less.	It might not go over how the QoE-aware strategy works with connections, that are frequently employed in fog computing implementations.
Mahmud et al. [36]	Fuzzy Logic	The proposed policy performs better than the existing algorithms.	NRR, PTRR, RG, and average application placement time are considered.

Baranwal and Vidhyarthi [65]	Proposed a FON selection Model.	Performs better than the existing state of the artwork.	Compared proposed work with state of art work.
Souza et al. [66]	Analyzed service placement strategy in different computing paradigms.	Performs good in-service response time and network core load.	Response time and network load are considered.
Taneja and Davy [67]	Designed a Module Mapping Algorithm for efficient utilization of resources.	Can serve as a micro-benchmark instudiesrelated to IoT and Fog computing.	Worked on energy consumption,network usage, and application latency.
Mahmud et al. [69]	Proposed a Latency-aware application module management strategy for the fog environment	Improved performance in terms of other latency-aware strategies.	Worked on deadline and resource optimization of applications.
Gu et al. [70]	Mixed integer linear programming and two-phase heuristic algorithm.	Outperforms a greedy algorithm.	Performance metrics have not been considered.
Yang et al. [71]	Designed heuristic algorithm for BSPP.	Achieves low latency, better cost, and running time.	Resource utilization and waiting time are not calculated.
Benamer et al. [72]	Integer Linear Programming model.	Performs well in terms of both exact and heuristic solutions.	Resource utilization and waiting time are not calculated.
Xiong et al. [73]	Markov decision process (MDP)	Exhibits excellent empirical performance.	Performance metrics are not evaluated.
Naranjo et al. [54]	Presents a multi-tier structure FOCAN.	Enhances the performance of diverse things' communication in smart cities.	Performance metrics are not evaluated.

2.2.2 RESOURCE ALLOCATION

The resource allocation is different in the Fog computing environment as compared to the cloud. Fog computing consists of various Fog nodes distributed geographically and the cloud network comprises of cloud user and its server. Due to the emerging requirements of IoT services such as QoS, and fairness in service distribution, designing a resource allocation strategy is a challenging task in a Fog computing environment [82]. The work related to the resource allocation strategies in the Fog computing environment is described below:

Tuli et al. [52] developed a FogBus named framework which enables integration of IoT-Fog-cloud. This framework provides platform-independent interfaces and for interaction, and execution the computing instances were offered. This framework also helped users at a time to execute multiple applications. The authors also have examined the FogBus settings through the placement of IoT applications in real-time on system parameters. The simulation results show that the designed framework is comparatively responsive and lightweight.

Kim and Chung [83] proposed a participatory Fog computing architecture by the user based on incentives. With the purpose of Fog instance placement, the process should take place in such a manner that the incentives are minimized and paid to the users of the proposed architecture. Through simulations based on device power consumption and actual service workload, the proposed instance placement technique is contrasted with other approaches. This would result in maximum profit for the infrastructure operator.

Concone et al. [84] have presented a multi-device HAR system that takes advantage of the fog computing paradigm and moves intensive processing from the sensor layer to intermediary devices and finally to the cloud. This option enables the circumvention of wearable device processing and storage constraints while also consuming less bandwidth than conventional cloud-based alternatives. The goal of experimental analysis is to assess the overall platform's performance in terms of the recognition process' accuracy while also highlighting the advantages it might have in intelligent surroundings.

Avgeris et al. [70] have proposed a cluster of edge servers' two-level resource allocation and admission control method giving mobile users another option for carrying out their duties. While an optimizer deals with load balancing and application placement issues to maximize the number of offloaded requests at the upper level, a set of linear systems at the lower level model the behavior of edge servers, and linear controllers are created to meet the system's constraints and QoS metrics. The evaluation shows how well the suggested offloading

technique performs in terms of performance measures like average application response time and the best use of edge servers' processing capabilities.

Azam et al. [86] have proposed historical record-based resource estimation for cloud service customers (CSC). To estimate resources, they offer a mathematical model that takes consumers' give-up probability into account. The algorithm associates the output of the historical record ratio module with the category of resource-requesting device. Finally, based on these variables, resources are estimated. Pooranian et al. [87] have presented a new FC IoE architecture to implement the resulting IoE architecture technology platform for FoE. Then, the authors elaborate on the associated QoS specifications that the underlying fog of everything (FoE)technology platform must meet.

Ni et al. [88] have introduced Priced Timed Petri nets (PTPN)-based resource allocation strategies for Fog computing, allowing users to select suitable resources on their own from a pool of pre-allocated resources. Their approach carefully weighs the price and time required to finish a work, as well as how highly users and fog resources are regarded in terms of credibility. This paper provides the dynamic fog resource allocation mechanism in particular. Simulation findings show that in terms of job completion time and cost, the proposed algorithms can be more efficient than static allocation schemes.

Neto et al.[56] have presented the Multi-tenant Load Distribution Technique for Fog Situations (MtLDF)algorithm to optimize load balancing in fog environments while considering unique multi-tenancy requirements (delay and priority). To illustrate the applicability of the suggested method in comparison to a Delay-Driven Load Distribution (DDLD) technique, the authors next presented case studies. The analysis revealed that MtLDF is capable of more efficient load distribution than DDLD.

Naas et al. [89] have introduced a new platform such as iFogStar to reduce the latency of the proposed IoT data placement strategy. iFogStar had an advantage, from the Fog infrastructure complexity to modify the data placement based on node features. In this paper, a platform is provided based on iFogSim for analyzing the strategies.

Velasquez et al. [90] have designed architecture for the placement of services at the Fog level. The main objective of this architecture was to discover the services and move these services following the network's varying conditions. Another main aim was to locate the services based on the user's status, learning through the communication environment. The proposed architecture modules have been characterized according to the correlations among

them. The authors have discussed the actions of the service orchestrator inclusive of the implementation details using ILP.

Natesha and Geddetti [91] have designed a two-level Fog infrastructure based on docker and containers to offer the resources. The authors in this paper have formulated the problem as a multi-objective optimization problem to maintain the QoS of IoT apps. In this paper, experimentation is done on the Fog testbed using container and docker on a cluster of devices with a 1.4 GHz 64-bit quad-core processor. The proposed EGA algorithm performs better than the other existing algorithms.

Basu and Ghosh [92] have worked on the selection of the most appropriate service provider with the help of the TOPSIS methodology by analyzing its accessible features and offerings. Along with this TOPSIS method, the authors studied several other MCDA methods that exist in the literature. The authors used the TOPSIS technique to select the most suitable cloud for an organization. Whereas the thorough study helped the cloud customers choose the optimal service provider from a set of different cloud features.

Kiani and Ansari [93] have proposed a three-tier cloud architecture using an LTE advanced backup haul network. The authors explored an approach in which the computing resources are provided in a profit maximization manner and to satisfy the user's quality of service communication resources are allocated. Later, they introduced an optimization approach for resource allocation, and heuristic algorithms to solve the problem of VM pricing and VM distribution. In this paper, a centralized solution is originated for the bandwidth allocation problem.

Afrin et al. [94] have proposed a non-linear programming solution that creates a balance between profit-aware resource allocation problems and QoE. In this paper, two algorithms have been developed to maintain the minimum requirement level for the other respectively. The request based on the required QoS, data rates, and connectivity was prioritized and after that, an optimized resource scheduling was introduced. The simulation of the proposed algorithms has been done a simulation toolkit which performs better than the other works.

Mishra et al. [95] have proposed a novel A-MCDM method to obtain the ranking of alternatives with less response time and time complexity respectively for the Fog computing environment. The proposed method was found to be more reliable than other MCDM methods. The performance of the A-MCDM method was examined using precision measures, Spearman-rank correlation; mean absolute error, and response time. The proposed method

might be useful in determining the various types of virtual machines that were needed to address the requirements at the application level. The authors suggest that the proposed model must be tested in a dynamic and diverse environment.

Deng et al. [96] have analyzed the tradeoffs between transmission delay and power consumption in Fog-cloud computing. The authors have worked on the problem of workload allocation that implies the optimal workload allocation admits cloud and Fog. In this paper, a systematic framework has been developed for Fog – a cloud computing system that decomposed the PP into three SPs. Experimental results represent the Fog's correlation with the cloud and the optimization performed was in a centralised manner.

Wadhwa and Aron [97] have used a unique approach to the management and allocation of resources. To ensure resource use at the fog layer, TRAM, a technique for resource allocation and management, is suggested. Using the expectation maximization (EM) algorithm, this method tracks the level of job intensity and determines the current resource situation. Concerning Fog computing, this study offers a scheduling algorithm for the resource grading procedure. This method's effectiveness is evaluated using the iFogSim simulator, and the outcomes are contrasted with SJF, FCFS, and MPSO. The experimental findings showed that TRAM effectively reduces task execution time, network usage, energy use, and average loop delay.

Zhou et al. [98] have introduced a productive incentive system based on the theoretical modeling of contracts. The difference is specifically created for the unique characteristics of an individual type of vehicle to maximize the predicted utility of the base station. Thus, the issue of work assignment is transformed into one of matching user equipment and vehicles on both sides. The identified issue is resolved by a pricing-based stable matching approach that repeatedly runs the developed and price-rising procedures based on the dynamically updated preference lists to produce a stable matching. Finally, numerical outcomes show that the suggested system is capable of significantly improving performance.

Wang et al. [99] have created the first edge node management framework, called the Edge NOde Resource Management (ENORM) framework. There are suggested mechanisms for auto-scaling and provisioning edge node resources. Based on a use case like a Pokemon Go-like online game, the framework's viability is proven. Application latency is minimized by 20–80% when ENORM is employed while transferring data and communication frequency

between the edge node and the cloud is decreased by up to 95%. These results demonstrate how fog computing can raise customer satisfaction and service levels.

Naha et al [100] have provided a ranking of resources and provision in a hybrid& hierarchical way as resource allocation and provisioning algorithms to address the challenge of meeting deadline-based user requirements. By adding a real case of Fog environment to the CloudSim toolbox, the proposed methods are tested in a simulation setting. With an increase in application submissions, the findings show that the suggested algorithms perform better than existing methods in terms of entire network delay, data processing time, and instance cost. When compared to current solutions, the average processing cost and time are reduced by 12% and 15%indivisually.

Sood and Singh [101] have introduced a unique theory called Free Resource Fog (FRF) that is offered as a Social Network Analysis (SNA) based upon a deadlock manager that aids in breaking the deadlock by gathering free resources from all active activities. A rule-based algorithm that allows priorities to the jobs and distributes resources from fog and cloud following those priorities is proposed to maximize resource utilization and decrease response time for the submitted work. The authors concluded that future computing with optical fog systems can be made possible with the best resource usage and latency measures.

Chang et al. [102] have designed a dynamic optimization scheme with multiple mobile devices (MDs) for the IoT fog computing system, in which offloading decisions and radio and computational resource allocation could be dynamically coordinated and allocated based on the varying needs for compute and radio resources. The authors presented a joint compute offloading and radio resource allocation technique derived from Lyapunov optimization to reduce the system cost related to latency and weights of MDs.

Duo et al. [103] have presented a differential game-based approach for context-aware dispute resolution. When distributing service resources for scenarios, conflicts in resource allocation will arise due to varying priorities because distinct context awareness activities have varied priority characteristics. This study groups distinct scenario perceptions with priorities, models the allocation of service resources according to the continuously differentiable state equation of a differential game, and builds the objective functions of utility maximization by grouping. It considers the dynamic properties of resource allocation in wisdom networks. The Berman dynamic programming approach resolves the optimal outcome of the objective function.

Chen et al.[104] suggested a two-part CRACCR system. The first is known as "spectral multiplexing computation consideration," which consists of the communication resources of the system while taking resource allocation into account for calculation. The other is FN scale adjustment (FNSA), in which the allocation of communication resources affects the number of FNs in use. Furthermore, the authors designed a mechanism to sketch users' credibility to create a user-aware CRACCR scheme.

Zang et al. [105] have presented a fog computing-based VNET and conducted research on resource allocation as the associated key approach. By using the Perron-Frobenius theory and the weighted minimum mean square error (MSE) method, the presented optimization issue is first described as a mixed-integer nonlinear program and then converted into a convex problem in this paper. The presented solution can greatly reduce the transmission time with fast convergence, according to numerical data.

Abbasi et al. [106] have modified the energy model of these devices using the NSGA II algorithm to reach a compromise between power usage and delay in fog devices. According to the simulation findings, this adjustment not only lowers the power consumption of fog devices but also considerably lowers the time it takes for packets to be processed on the IoT. The two evolutionary algorithm-simulated situations involved sending the controller IoT requests at full and half capacity. As a result, this paradigm may be applied in networks where the processing resources cannot access green energy. Table 2.2 presents the techniques discussed in the above section.

Table 2.2: Resource Allocation Techniques

Authors	**Technique Used**	**Results**	**Limitations**
Tuli et al. [52]	Proposed a FobBus framework based on	The proposed framework is lightweight and responsive	The complexity of the proposed framework is high
Kim and Chung [83]	Proposed an incentive-based Fog computing architecture based on Integer non-linear programming	Focused on device power consumption and workload	
Concone et	Developed a multi-	Performance is	Focuses only on the

Authors	Technique Used	Results	Limitations
al. [84]	device HAR system that	determined in terms of the recognition process.	bandwidth of the system.
Avgeris et al. [85]	Proposed a cluster of edge servers' two-level resource allocation and admission control method.	Performs well under the considered performance metrics.	Have not considered the completion time of the proposed system.
Azam et al. [86]	proposed a historical record-based resource estimation	minimize resource underutilization, dynamic resource estimate is therefore carried out.	
Pooranian et al. [87]	Proposed a new FC IoE architecture to implement the resulting IoE architecture technology platform for FoE	The proposed framework reduces energy consumption.	
Ni et al. [88]	introduced with Priced Timed Petri nets (PTPN)-based resource allocation	in terms of job completion time and cost, the proposed algorithms can be more efficient than static allocation schemes	The task completion time of the proposed platform is not considered.

2.2.3 RESOURCE SCHEDULING

The most effective way to employ processing time and provide resources to programs is through scheduling. The resource scheduler's main responsibility is to select the next process to run out of a list of appropriate processes. Resource estimation helps in fog computing to allocate computing resources based on specific parameters, making sure that there are enough resources available for upcoming computations. In fog computing, scheduling approaches can also help manage latency, load, and duplication. Both Fog computing and cloud computing require scheduling techniques and load-balancing algorithms. Some of the resource scheduling papers are discussed in detail below.

Lin and Yang [107] have investigated the use of a cloud center, gateways, sensors, and edge devices linked with a logistics center's facilities to create an intelligent computing system. This paper sets up an integer programming model for positioning fog devices, gateways, and edge devices in their potential sites, except for the locations of the cloud center and sensors that were identified according to the factory layout. The NP-hard facility location problem is further solved in this study using a metaheuristic method that combines discrete monkey searches to find high-quality answers with a genetic search to speed up computation.

The suggested algorithm's good performance in the deployment of intelligent computing systems in moderately-sized instances of intelligent logistics centers is confirmed by simulation. Rahbari and Nickray [108] have provided a scheduling derived from knapsack that is symbiotic organisms search-optimized and is simulated in iFogSim, a common Fog computing simulator. The results reveal that their scheduling method outperforms the First Come First Served (FCFS) and knapsack algorithms in terms of energy consumption, total network usage, execution cost, execution time, and sensor lifetime.

Gupta et al. [109] introduced iFogSim to simulate and model the environments of IoT, fog, and edge computing. Particularly, with regards to diverse workloads, iFogSim offers examination and comparison of managing resource strategies based upon QoS standards like latency. The authors presented two test cases and showed the usefulness of iFogSim for assessing managed resource strategies, such as placing a cloud-only application module placement and a strategy that pushes applications onto edge devices when sufficient resources are available. The outcomes of the experiment showed that iFogSim can handle simulations at the scale anticipated in the context of IoT.

Zeng et al. [110] have considered a software-defined embedded system enabled by Fog computing, where task pictures are stored on a storage server and calculations can be done on either a computation server or an embedded device. To enhance the user experience, it is important to develop an effective strategy for task scheduling and resource management with reduced job completion time. Through a combined analysis of task picture placement and task scheduling, the authors looked into the FC-SDES problem of minimizing the maximum job completion time.

Murtaza et al. [111] have proposed an LRFC technique for task scheduling in a Fog computing environment. The main objective of the proposed algorithm was to enhance energy consumption and QoS. In this paper performance bottlenecks have been reduced by the proposed deployment model that was scalable. The authors also introduced a new layer between the nodes and the IoT devices that would be expanded for implementing several other learning policies. Extensive simulation has been conducted for the evaluation of the proposed approach regarding QoS and energy efficiency.

Subbaraj and Thiyanranjan [46] have proposed the first MCDM-based scheduling algorithm based on their literature survey. The nature of module mapping of applications to Fog devices has been improved with the help of MCDM techniques in this paper. Some outcomes have been made from the analysis of the MCDM-based proposed method and experimental tests. Although cloud data centers are proper for applications with MIPS, Storage, and RAM and Fog environment is suitable for low cost, high security, and less power consumption. In this paper two distinguished evaluation methods have been proposed for selecting the best device.

Pham et al. [112] have provided a cost-aware scheduling algorithm to maintain the cost of cloud resource usage and the application execution performance. Also, to improve the output of the proposed algorithm, a reassignment strategy based on the critical path of the direct acyclic graph was proposed. The authors suggest that to obtain an advantage from this platform, the computing tasks should be allocated at every processing node strategically. Several simulations have been performed and the results depict that the proposed algorithm exceeds other existing algorithms.

Fan et al. [113] have proposed a deadline-aware task scheduling system in a hierarchical IoT infrastructure for a Fog computing environment. In this proposed system the link between their Fog nodes is utilized by the service providers. The authors have prepared the task scheduling problem as a 0-1 knapsack problem in such a cloud-Fog environment which is

NP-hard. They also proposed an ant colony optimization-based algorithmic solution for this problem. The main aim of this paper is to improve the profits of Fog service meanwhile meeting the needs of task deadline constraints. As compared to the existing heuristics the proposed optimization enhances the system performance.

Yadav et al. [114] have explained a modified version of the fireworks algorithm, combining principles from differential evolution with opposition-based learning. To avoid local optima, the differential evolution operator has been implemented, and opposition-based learning has been used to develop a population with a diverse set of solutions. The suggested approach minimizes costs and lead times while increasing resource efficiency. The trials were run on a variety of workloads, and the results were contrasted with some current, well-liked metaheuristic methods. The comparison has confirmed the value of the suggested strategy.

Yin et al. [115] have designed a fresh task-scheduling framework by taking into account containers' functions. Then, to guarantee task completion on time and optimize the number of concurrent jobs for the Fog node, a task scheduling algorithm is built. Finally, following the properties of the containers, they suggest a reallocation strategy to reduce task delays. The outcomes demonstrated that our suggested task scheduling technique and reallocation scheme may successfully decrease task delays and increase the number of processes running concurrently in Fog nodes. Tychalas and

Karatza [48] has proposed a Bag-of-Tasks workload paradigm to examine the viability of utilizing every resource that is available to lower overall costs. The authors presented a Fog computing system method that, by making use of all available resources like monthly fees for virtual machines. To further cut expenses and speed up execution, and further expanded their research to look into the use of containers rather than virtual machines. The findings of the simulation demonstrate that by integrating all available resources, costs can be saved while maintaining a relatively constant mean response time.

Jamil et al [116] have presented a state-of-the-art Fog computing scheduler that streamlines network consumption and delays while facilitating service delivery for the Internet of Everything (IoE). The authors give a case study to show how to effectively schedule requests from IoE devices and respond to their resource demands on each Fog device. They compare the proposed scheduling algorithm to existing methods using iFogSim, taking into account latency and energy usage as performance measures. According to the results, the proposed

scheduler's delay and network use are reduced by 32% and 16%, respectively, when compared to the FCFS technique.

Fizza et al. [117] have proposed a privacy aware scheduling in a heterogeneous fog environment (PASHE), an algorithm that schedules real-time jobs with privacy restrictions on heterogeneous cloud data center (cdcs) and micro data center (mdcs). Private, semi-private, and public responsibilities have all been taken into account. Users' local mdcs are used to carry out private jobs with strict timeframes. The cdc is given public assignments with ambiguous timelines to complete. In a Fog computing environment, simulation findings demonstrate that PASHE outperforms conventional scheduling algorithms by taking into consideration mdc heterogeneity, user mobility, and application security.

Ding et al. [118] have proposed a cost-effective multi-workflow-scheduling technique under time restrictions to solve such an issue. First, they establish models for resource usage and workflow execution time in fog computing. Following that, to analyze the cost of executing workflows within predetermined timeframes, a fitness function is used in a novel multi-workflow scheduling technique based on PSO (Particle Swarm Optimisation). A heart rate monitoring app is used as a motivating example, and thorough experimental findings demonstrate that, compared to existing solutions, our suggested strategy can dramatically lower the execution cost of various workflows within set timeframes.

Benblidia et al. [119] have presented a ranking-based task scheduling system that incorporates user preferences and fog node features using linguistic and fuzzy quantified propositions. The system ranks fog nodes from the most to the least pleasant. Additionally, to separate the similarities, they employed two criteria termed least satisfied percentage (LSP) and greatest satisfactory proportion (GSP). The outcomes of the simulation demonstrated that their plan successfully schedules tasks while satisfying user preferences. Additionally, it offers a compromise between average-user pleasure, execution time, and energy consumption.

Yang et al. [120] have established an analytical model for precisely assessing the energy efficiency (EE) in homogeneous fog networks, that takes into account circuit, computing, and offloading energy consumptions. This model allows us to devise the EE optimization problem for future intelligent IoT apps with realistic constraints in available computing resources at helper nodes and unused spectrum in nearby environments. The model investigates the tradeoff relationship between performance gains and energy costs in

collaborative task offloading. To choose the best scheduling option for a task node and numerous surrounding helper nodes under practical modulation schemes and time allocations, a technique called maximal energy-efficient task scheduling(MEETS) has been devised. This algorithm is based on rigorous mathematical analysis.

Wang et al. [121] proposed a task scheduling approach (I-FASC) for the features of tasks and resources in this study, and also an improved genetic algorithm (I-FA) is provided by introducing the explosion radius detecting mechanism of fireworks. Then, by simulating a cloud-fog system, they may quickly improve load and execution times by running two sets of trials in this system. As fog computing becomes more widely used, more people will be drawn to study this objective discussed in this paper. The study of task scheduling in fog computing is covered in this paper, and some research findings were made.

Tuli et al. [122] have suggested a real-time scheduler for stochastic Edge-Cloud systems that is A3C-based and enables concurrent decentralized learning across many agents. The authors collect numerous host and task factors using the R2N2 architecture, along with temporal patterns, to produce effective scheduling decisions. The selection of hyper-parameters is explained via sensitivity analysis in this paper. When compared to state-of-the-art algorithms, the studies on real-world data sets demonstrate considerable progress in terms of energy consumption, reaction time, SLA, and operation cost by 14.4, 7.74, 31.9, and 4.64 percent, respectively.

Vijayalakshmi et al. [123] have presented a unique task-scheduling technique that focuses on minimizing the makespan and maximizing resource consumption in the fog environment. Based on the median Suffrage value, this algorithm classifies the work. The recommended technique significantly reduces makespan while maximizing resource consumption. The test results show that our approach outperforms other well-known algorithms in terms of resource usage rate and makespan when compared to several live scheduling methods for performance analysis.

Zhang et al. [124] have proposed a unique delay-optimal task scheduling (DOTS) technique to achieve the offloading solution following the claimed capabilities of the VNs. Numerous models in a fog network are run, and the numerical results show that the DOTS algorithm could efficiently provide the best combination of subtask sizes, helper nodes, and TN transmission power to reduce the overall task-processing delay. Additionally, the voluntary mode offloading delivers a more balanced unloading and a higher level of justice among the

FNs as compared to the command-mode offloading. According to the presented voluntary capability report ratio in this paper, balanced energy consumption and a higher level of fairness among the FNs have been accomplished in the fog network's voluntary mode.

Sun et al. [125] have designed a two-level resource scheduling approach. They have used a theory of the enhanced non-dominated sorting genetic algorithm II (NSGA-II), which takes into account the diversity of various devices; they have created a resource scheduling system for fog nodes inside the same fog cluster. The outcomes of our scheme's MATLAB simulation demonstrate its effectiveness in reducing service latency and enhancing task execution stability. The cost of the resource requesters should be taken into account in addition to their service quality.

Shetty and H [126] suggested the structure using machine learning techniques to maximize the use of cloud computing resources. Task scheduling can take into account a variety of factors, including Makespan, QoS, energy usage, execution time, and load balancing. Instead of assigning the scheduling algorithm at random, they suggest using a machine learning technique to classify the best acceptable algorithm for each task request as it comes in. Techniques for supervised machine learning can be applied here. The suggested research's findings enable the optimum task scheduling method to be chosen for the input task (request).

Rahbari and Nickray [127] have analyzed scheduling tactics and parameters and offer the greedy knapsack-based scheduling (GKS) algorithm for properly distributing resources to modules in a fog network. As a common FC simulator, iFogsim was used to mimic our suggested technique. GKS outperforms FCFS, concurrent, and delay-priority algorithms in terms of total execution cost, energy usage, and application loop time in DCNS and VRGame.

Pham and Huh [128] have examined how a cloud provider can make use of the cooperation between its fog nodes and the rented cloud nodes to effectively execute users' large-scale offloading applications by scheduling tasks in a cloud-fog computing system. Before proposing a heuristic-based approach, the authors first describe the problem of task scheduling in this cloud-fog environment. This program's main goal is to strike stability between the makespan and the financial cost of cloud resources. The numerical outcomes demonstrate that our suggested approach outperforms other current algorithms in terms of tradeoff value.

Nguyen et al. [129] have provided a novel method for solving the task scheduling problem for Bag-of-Tasks applications running in a Cloud-Fog environment. On 11 datasets with different sizes, the suggested algorithm known as TCaS was evaluated. While attaining a balance between finishing time and operating cost, the experimental findings demonstrate improvements of 15.11 percent when compared to the Bee Life Algorithm (BLA) and 11.0 percent when compared with Modified Particle Swarm Optimization (MPSO).

Mukherjee et al. [130] have suggested a scheduling method with the main objective will maximizing the amount of activities that are finished within their allotted deadlines while maintaining good network stability. According to the findings of our simulations, the suggested technique works better than the baseline schemes, particularly when the jobs have clear delay deadlines. The above-discussed algorithms have been summarised in Table 2.3.

Table 2.3: Resource Scheduling Algorithms

Authors	**Technique Used**	**Results**	**Limitations**
Lin and Yang [107]	Proposed an integer programming model for deploying gateways, fog devices, and edge devices	Performs well in the deployment of an intelligent computing system	Data gathering techniques are not strict, and the validity and dependability of the findings may be affected. .
Rahbari and Nickray [108]	Proposed a knapsack-based scheduling	Enhances energy consumption, network utilization, and lifetime of sensors.	Time taking technique
Gupta et al. [109]	Designed iFogSim simulator to simulate and model the environments of the fog, and edge-computing.	iFogSim can handle simulations at the scale anticipated in the context of IoT.	The toolkit might be dependent on notions of the characteristics and behavior of fog computing environments which might not accurately reflect the complexity of real-world situations.

Zeng et al. [110]	Proposed an effective task scheduling and resource management strategy.	Reduced completion time of the jobs.	Focuses on completion time only.
Murtaza et al. [111]	Proposed a LRFC technique for task scheduling	Extensive simulation has been done for the evaluation of the designed approach regarding QoS and energy efficiency.	Employs restricted metrics to assess how well QoS-aware provisioning strategies work.
Subbaraj and Thiyanranjan [46]	proposed the first MCDM-based scheduling algorithm	the result demonstrates the productivity of the proposed algorithm over the other three considered algorithms.	Due to Limited Performance metrics not covering a full range of challenges.
Pham et al. [112]	Designed a cost-aware scheduling algorithm	Performs better than existing algorithms.	Focused on maintaining the cost of the proposed framework only.
Fan et al. [113]	proposed a deadline task scheduling system in a hierarchical IoT infrastructure for a Fog computing environment	Improved the profits of Fog service provider while meeting the needs of task deadline constraint	Employs a limited set of criteria to assess the scheduling framework's performance, which may not fully account for all scheduling needs and user satisfaction.
Yadav et al [114]	Presented a modified version of the fireworks algorithm	The suggested approach minimizes costs and lead times while increasing resource efficiency.	Metrics might not fully account for user satisfaction and performance needs in fog computing.

Yin et al. [115]	Proposed a fresh task scheduling framework by taking into account containers' function	Suggested task scheduling technique and reallocation scheme may successfully decrease task delays and increase the number of processes running	Research work is not able to distribute and manage resources throughout this diverse infrastructure taking into account variables like resource availability and closeness to data sources.

2.2.4 RESOURCE PROVISIONING

The IoT environment is dynamic; there are workload changes for IoT services, which can lead to over or under provisioning problems. The resources in an over provisioning situation have a greater ability to handle the necessary demand, therefore the SLA is not broken. However, when there is a problem with under-provisioning, the resources do not have adequate capacity to handle the necessary workload, which violates SLA. To manage resource provisioning and reduce system costs, resource provisioning must be done dynamically in a Fog environment. Some of the resources provisioning papers are discussed in detail below.

Avasalcai and Dustdar [131] provided a cutting-edge distributed resource allocation technique to facilitate the deployment and integration of various applications in an IoT environment. The authors proposed a distributed latency-aware resource provisioning technique for IoT application deployment that ensures SLA fulfilment.

Yao et al. [132] have provided the solution for the joint optimization problem, which is represented as a mixed integer nonlinear programming problem, to reduce the system cost (VM rents) while maintaining QoS criteria. It is crucial to concurrently optimize Fog resource provisioning (decisions on the amount of VMs to rent) and power control because both fog processing and wireless transmission have an impact on QoS (i.e., task completion time). The solution to the issue is then suggested as an approximation algorithm. Simulation results show how well the suggested approach performs.

Santos et al. [133] have proposed an energy-efficient Deep Reinforcement Learning (DRL) method for SFCA in fog computing is suggested. Here, the agent learns from a previously presented mixed-integer linear programming (MILP) formulation the best resource allocation choices with a cost-reduction focus. Results reveal that, in dynamic use cases, our agent

achieves performance comparable to cutting-edge MILP formulations, accepting 95% of requests.

Donassolo et al. [134] have proposed a FITOR orchestration system in the Fog environment for IoT applications. The proposed system provides an efficient orchestration mechanism and a realistic Fog environment. FITOR system depends on O-FSP which is an optimized Fog service provisioning strategy that aims at the low cost of IoT applications. The experimental results showed that the O-FSP optimizes the IoT application placement and performs better than other strategies in terms of resource usage, provisioning cost, and acceptance rate.

Agarwal et al. [135] have proposed an efficient resource allocation architecture and algorithm (ERA) to evaluate the effectiveness of the suggested method in the fog environment and implemented using the cloud analyst tool. In terms of overall expected response time and cost, this document compares the proposed algorithm with their current resource allocation approach.

Ogundoyin and Kamil [136] have utilized the fuzzy AHP technique for the criteria of trust formation in Fog computing services. The evaluations showed that the QoS was a high priority and QoSec was a low priority. In this paper, social relationships were at the top rank, and past reputations were considered to be the last priority. In this paper, the authors have utilized the hybrid fuzzy AHP method, whereas other existing works could be integrated with fuzzy AHP to form a better combination.

Maruf et al. [137] have discovered the required number of Fog nodes for over-the-air (OTA) updates. The authors employed k means clustering for Fog node distribution and located the traffic load. A case study has been demonstrated in this paper to prove the efficiency of the proposed algorithm. To predict the communication delay among vehicles and Fog devices a machine-learning approach has been employed. In this paper it was observed that the net Fog resources increased by an average of 26.57% and OTA time have been reduced.

Dehnavi et al. [138] have developed a resource provisioning plan that takes dependability and real-time needs into account when dividing a given workload among these several computer levels. Workload partitioning could assist in making important design decisions, such as how many computing resources are needed to construct a local private Cloud that works with Fog nodes, how much communication bandwidth is required between the Fog and the public Cloud data center, to meet the application's reliability requirements.

Etemadi et al. [139] have described a method for resource provisioning that is effective. To decide how to adjust the scaling of Fog resources to handle the workload from IoT services in the fog environment, this method is inspired by an autonomous computing model that uses Bayesian learning. Finally, testing is done to check the performance of this system using three different workload traces. The results show that, in comparison to other ways, the suggested solution decreases the overall cost and delay violation and boosts fog node use.

Azam and Huh [140] have included these elements in the proposed technique for resource estimate and management, formulating resource management based on the customer's shifting relinquish likelihood, service kind, service pricing, and variance of the relinquish probability. The discussion and findings demonstrate that these variables can assist service providers in estimating the appropriate resource level for each type of service user.

Zhao et al. [141] have offered a multilevel and multidimensional (MM) model-based platform that is capable of accessing vast heterogeneous resources and exposing their features to lightweight services. The platform offers the fundamental infrastructure for the IoT application pattern: inner-domain autonomy and inter-domain coordination. It also allows applications that share and reuse resources.

Fard et al. [142] have designed a container scheduler with application awareness that aids in orchestrating heterogeneous resources for edge and fog systems. The suggested scheduling mechanism chooses the most suitable host to achieve the lowest response time for a certain IoT service by taking into account the available capacity, and also dynamic system status. Furthermore, the proposed method performs better than Docker Swarm's scheduling tools.

Chandak and Ray [143] have proposed a resource provisioning mechanism based on several agents that allocate resources following load. The authors considered three sorts of agents for resource provisioning. The simulation findings show that the suggested strategy decreases makespan, average execution time, and flowtime as compared to the already used technique, which they examined to assess the effectiveness of the proposed agent-based resource provisioning technique.

Nguyen and Minh [144] have mainly focused on the optimization challenge that reduces an application's overall response time while considering network and server usage. Tenth, they have tested two service deployment approaches, dubbed the cloudy and the foggy approaches. To demonstrate the effectiveness of the suggested foggy service deployment

technique, the authors statistically examine the overall response time, network consumption, and server usage of those two strategies.

Yao et al. [145] have explored how to deploy the servers in a Fog computing environment in a cost-effective method without desecrating the QoS. The authors have considered heterogeneous cloudlet servers which consist of varying resource capacities and cost. The problem was developed into an integer linear programming problem and to solve this low complexity heuristic algorithm was proposed. Simulation results validate the efficiency of their proposed algorithm as it performs much closer to the optimal solution.

Liu et al. [146] have presented a randomized approximation algorithm to solve the NP-hard problem of minimizing total cost. The proposed algorithm is well known as the partial rounding algorithm (PRA) that assures approximation. Later the authors also proposed an improved version of the Chernoff bound which was applied to the PRA algorithm. The experimental results represent that the improved version of the proposed algorithm helps in achieving better results close to the optimal one.

Abouaomar et al. [147] have offered a resource representation strategy that would enable each ED to expose its resource information to the edge node supervisor using the mobile EC application programming interfaces suggested by the European Telecommunications Standards Institute. Each time a resource allocation is necessary, the edge node supervisor is made aware of information on the ED resource. According to the simulations, our suggested solution performs better than previous benchmark approaches and offers minimal latency and efficient resource use.

Santos et al. [148] have proposed a Mixed Integer Linear Programming (MILP) formulation that takes into consideration SFC ideas, various LPWAN technologies, and numerous optimizations for the IoT service allocation problem. Their research advances from the state-of-the-art in Fog cloud systems by offering full end-to-end (E2E) resource provisioning while taking cloud and wireless network requirements into account. A thorough evaluation of the proposed MILP formulation for use cases in Smart Cities has been conducted. The outcomes demonstrate trade-offs between the various provisioning techniques.

Son and Buyya [149] have provided a resource provisioning algorithm for VNFs to use edge and cloud resources. The system intelligently distributes resources for VNFs across the edge and cloud, according to dynamic changes in network volumes. The authors have compared their approach with the cutting-edge baseline algorithm and assessed it in a simulation

environment with heavy web application demands. The outcome demonstrates that the suggested method processes 77.9% more packets in the edge nodes than the application non-aware algorithm, which decreases the end-to-end response time.

Yigitoglu et al. [150] have developed a framework called Foggy that makes it easier for automated application deployment and dynamic resource provisioning in Fog Computing architectures. The authors examined several applications and pinpointed the needs that must be taken into account when designing the Foggy framework. The authors developed a Foggy prototype and distributed it on a group of Raspberry Pi devices as a proof of concept.

Ostberg et al. [151] have outlined the plan for toolkits for continuous data collecting, modeling of application performance, auto-scaling and remediation of application and components, and deployment optimization. It also presents the REliableCApacity Provisioning and enhanced remediation for distributed cloud applications (RECAP) vision for integrated edge-cloud architecture. To demonstrate the developments of RECAP, this paper also provides four use cases from complementary domains. The research outputs from the RECAP project, which is currently underway and will last until 2019.

Mehmandar et al. [152] have suggested a distributed computing system for autonomous resource management. Then, through the control MAPE-k loop, they offer a customized version of an IoT service provisioning system. In the analysis stage, the system makes decisions using support vector regression, and in the planning stage, it makes decisions using reinforcement learning. Finally, they run a series of simulation-based tests to evaluate the effectiveness of the system. When compared to current solutions, the average delay, cost, and delay violation are reduced by 1.95 percent, 11 percent, and 5.1 percent, respectively.

Shahidinejad and Arani [153] have offloaded the dynamic workloads into either cloud or edge servers using decision-making with learning automata. Additionally, the authors provided a method for providing edge servers that makes use of a long short-term memory model to predict future workloads and a reinforcement learning strategy to determine the proper scaling. The simulation results obtained under actual and simulated workloads show that, when compared to alternative algorithms, the suggested solution boosts CPU usage while decreasing execution time and energy consumption.

Wang et al. [154] have addressed the major issue by creating DYVERSE, a technique on edge environments to support multi-tenancy. The objective is to raise the QoS of workloads that are being executed on a multi-tenant Edge node. To ensure that SLOs are not broken, the

suggested scaling method develops post-deployment plans for workloads while they are being executed. The static and dynamic priority management strategies support the mechanism. Three suggested dynamic priorities take workload, community, and system considerations into account.

Mulinti and Nagendra [155] have developed a mobile edge computing structure with a latency-sensitive resource provisioning method. The framework initially receives requests from SFC for virtual network functions (VNFs) to utilize edge and cloud resources. The asset provisioning method Adaptive Grey Wolf Optimisation (AGWO) is then used to efficiently schedule the SFC requests to the cloud-assisted edge network. The results of the exploratory investigations show that, in terms of system cost, arrival rate, and average reaction time, the proposed methodology is superior to the existing one. The above approaches have been outlined in Table 2.4.

Table 2.4: Resource Provisioning Approaches

Authors	**Technique Used**	**Results**	**Limitations**
Avasalcai and Dustdar [131]	Proposed a cutting-edge distributed resource allocation technique	The proposed approach potential to reduce the latency of various IoT applications.	Response time is not considered.
o et al. [132]	Handles the joint optimization problem	Results show how well the suggested approach performs	Response time is not considered.
Santos et al. [133]	Proposed an energy-efficient Deep Reinforcement Learning (DRL) method	The proposed system performs better than the MILP formulation	Performance metrics are not considered
Donassolo et al. [134]	proposed a FITOR orchestration system	results showed that the O-FSP optimizes the IoT application placement and performs better than other strategies	RAM and storage, response time, and resource utilization are not considered.

Agarwal et al. [135]	Designed an algorithm for effective resource allocation (ERA)	overall expected response time and cost, this paper compares the proposed algorithm with the current resource allocation approach.	Resource utilization is not considered.
Ogundoyin and Kamil [136]	The fuzzy AHP technique is used for identifying trust in Fog computing services.	The study elaborates on the validity of the Fuzzy AHP method in MCDM in a Fog environment and might have been a crucial tool in IoT.	
Maruf et al. [137]	K means clustering is used for Fog node distribution	It was observed that the net Fog resources increased by an average of 26.57% and OTA time has been reduced	Performance metrics are not considered
Dehnavi et al. [138]	Proposed a resource provisioning plan	Workload partitioning can help us make important design decisions	Performance metrics are not considered
Etemadi et al. [139]	Proposed a technique based on Bayesian learning	The suggested solution decreases the overall cost and delay violation and boosts fog node use.	Resource utilization is not considered
Azam and Huh [140]	Proposed a technique for resource estimation and management	Variables used in the proposed technique can assist service providers in estimating the appropriate resource level for each type of service user	Have not evaluated their proposed technique for resource utilization, resource estimation
Zhao et al. [141]	Proposed a multilevel-multidimensional model-based service provisioning framework	offers a consistent message space to make it easier for scattered IoT environments to distribute and share sensor data on demand	The authors have not considered any performance metric
Fard et al. [142]	Proposed a container scheduler with application awareness	The proposed Fog edge platform performs better and less cost is used.	Completion time is not included.

Chandak and Ray [143]	Proposed a resource provisioning mechanism	The proposed technique performs better	Response time is not included
Nguyen and Minh [144]	Proposed foggy service deployment technique	The trade-off between reducing network congestion, reducing server utilization in the cloud layer, and reducing application response time.	Completion time is not considered

2.3 OVERVIEW OF MCDM TECHNIQUES

MCDM is a quantitative and qualitative strategy that may be applied to problems with multiple solutions to find the best one and reach the best decision possible. This chapter provides information on all MCDM techniques and discusses the literature-supported applications of MCDM to a wide range of fields. To comprehend the nature of MCDM for various situations, it is important to grasp MCDM methodologies and their applications. The application of MCDM to other fields provides a concept for a new area of study.

The research assists in determining the issues that various researchers have investigated and the solutions they have proposed utilizing MCDM techniques like PROMETHEE, TOPSIS, and, AHP. The most popular techniques were AHP, and TOPSIS, although hybrid or integrated techniques offer solutions for a variety of issues, including location, finances, bankruptcies, bridge construction, wastewater, and many more. The new era in MCDM history is brought about by this combination. To address various types of challenges, numerous multicriteria approaches have been created. Although each strategy has advantages and disadvantages and may be more or less effective depending on the circumstance, only a few approaches have been put out to help in the selection of a technique that is appropriate for the particular context. The current literature is organized with a framework that directs the examination of each selection method following both its qualities and the characteristics of the MCDM techniques that it aids in choosing.

There are several issues in the world, and MCDM is the tool to find the appropriate answer. To make the methods and their process easier to understand, numerous books on MCDM [156–160] have been written. To find the best option for mobile interface selection, [161] employed TOPSIS. Many researchers use the MCDM tool in the decision-making process to ensure the best possible option. MCDM is a promising technique for complicated problem

analysis by weighing many alternatives, such as policies, scenarios, strategies, and weightings, according to various criteria and then picking the optimal alternative using mathematical calculations. In the literature, there are numerous MCDM approach kinds. Each method's properties vary and can be categorized as deterministic, stochastic, and a variety of other ways. There are several MCDM approaches, including [157]. There are many different MCDM techniques, as mentioned in the previous article. The crucial techniques that have gained a lot of attraction in recent years are discussed below.

2.3.1 The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS):

One of the methods that is frequently used to address MCDM issues is the TOPSIS technique [172]. Hwang and Yoon were the ones who first developed this technique [173]. The TOPSIS method depends on the assumption that the chosen alternative is the one with the shortest geometric distance to the ideal solution which is positive and the one with the largest geometric distance to the ideal solution which is negative [174].

Following are definitions of ideal and unfavorable ideal solutions:

- Ideal solution: A perfect solution would maximize benefits and minimize drawbacks while having the best attribute values.
- Negative ideal solution: This ideal solution ought to improve the disadvantage criterion and minimize the benefit criteria, and it ought to have the poorest attribute values.

The benefits of TOPSIS are seen in its straightforward usage and simple programming. No matter how many attributes are used, the TOPSIS process always follows the same steps. However, TOPSIS does not take into account the correlation between the qualities because it uses Euclidean distance. Furthermore, TOPSIS makes it challenging to maintain a consistent judgment, and this method completely ignores the ambiguity of weightings. The TOPSIS method is shown in Figure 2.1 for better understanding.

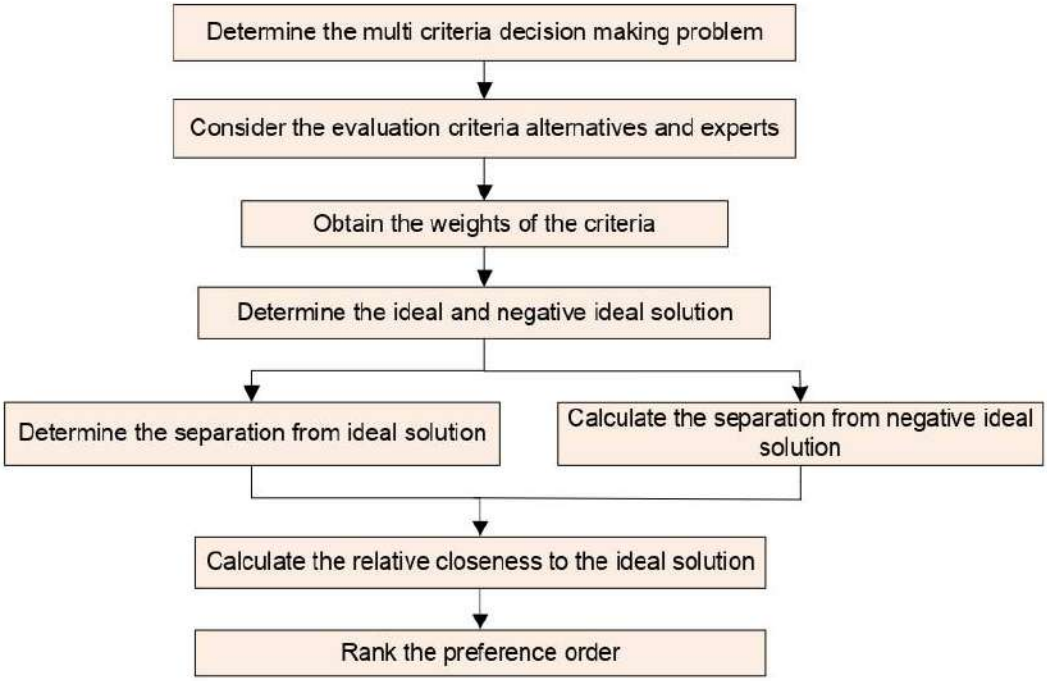


Figure 2.1: Workflow of TOPSIS Technique

The numerical steps of the TOPSIS technique are interpreted in detail below.

i. Matrix creation:

In this first step, a matrix is created with the help of alternatives and criteria that are considered for the evaluation of any specific problem. In the proposed framework the Fog environment and the QoE parameters are considered as the alternatives criteria respectively. Whereas the matrix's row and column signify the alternative and criteria respectively. FSP_{ij}, i = 1, 2,…..m and j = 1, 2,…..n. The matrix formed is defined as FSP_{ij}.

$$FSP_{ij} = \begin{bmatrix} X_1Y_1 & . & . & X_1Y_n \\ . & . & . & . \\ . & . & . & . \\ X_mY_1 & . & . & X_mY_n \end{bmatrix}$$

ii. Normalized decision matrix:

This matrix is created by dividing by the square of its column sum and overall square root.

$$NFSP_{ij} = FSP_{ij} / \left[\sum_{j=1}^{m} FSP_{ij}^2\right] 1/2$$

$$i = 1,2,....,m \ \& \ j = 1,2,....,n$$

iii. Pair-wise matrix:

In this step, a pairwise matrix is created using the AHP technique to identify the weights of the criteria. In this matrix creation, the value of the importance of n QoE parameters is used. Every QoE parameter for its i^{th} parameter with its j^{th}parameter forms a square matrix in the form of B_{nxn}.

iv. Geometric Mean:

Generally, the weights of every criterion that is considered in a complex problem are evaluated by the old AHP traditional technique [165]. However, due to an issue in the inconsistency of the weights, the geometric mean method was applied in the AHP technique [46]. Firstly, the geometric mean is evaluated and then the normalization of geometric mean values is calculated.

$$GM_j = \left[\prod_{j=1}^{n} B_{ij}\right] 1/n$$

v. Normalized matrix:

After the normalization of the weights is performed then the normalized weight matrix

is evaluated by the following formula.

$$W_j = GM\ j \Big/ \sum_{j=1}^{n} GM_j$$

vi. Relative normalized matrix:

In this step, the relative normalized matrix is calculated by applying this equation which is as follows:

$$NB_{nx1} = B_{nxn} * W_{nx1}$$

$$RNB_{nx1} = NQ_{nx1} / W_{nx1}.$$

vii. Consistency Index:

In this step, the maximum eigenvalue is obtained by taking the average of RNB_{nx1}. With the help of this matrix consistency index is evaluated as follows.

$$CI = (\lambda_{max} - n) / (n - 1)$$

viii. Consistency Ratio:

In this step, the ratio is the division of the consistency index upon random index. A random index is predefined by the various number of criteria as mentioned in Table 2.5.

Table 2.5: Value importance for a number of criteria [160]

Values	Meaning of values
1	Equally important
3	Moderately important
5	Strongly important
7	Very Strongly important
9	Absolutely important
2,4,6,8	Between the above-provided values

The ratio is given by the equation below.

CR =CI/RI

ix. Weighted Normalised matrix:

This step signifies the weighted normalized matrix by the equation below

$T = w_j NFSP_{ij}$

$i = 1, 2, \ldots, m \;\&\; j = 1, 2, \ldots, n$

where

$w_j = W_j \Big/ \sum_{j=1}^{n} W_j$; W_j are the initial weights given to the QoE parameters

x. Identify the optimal solutions:

This is the final step to compute the positive (T_j^+) and negative ideal (T_j^-)solution for the specified problem.

$T_j^+ = \{ \max (T/j) , \min (T/j) / i = 1, 2, \ldots N\}$

$= \{T_1^+, T_2^+, T_3^+ \ldots\ldots T_M^+\}$

$T_j^- = \{\min (T/j \in J), \min (T/j) / i = 1, 2, \ldots N\}$

$= \{T_1^-, T_2^-, T_3^- \ldots\ldots T_m^-\}$

where,

$T_M^+ = (j=1,2, \ldots, M) \Big/$ j associated with positive alternative value and

$T_M^- = (j=1,2, \ldots, M) \Big/$ j associated with negative alternative value

xi. Euclidean distance:

This phase involves calculating the Euclidean distance from both the ideal solutions, which is given as follows:

$S_i^+ = \left\{\sum_{j=1}^{n} (T_{ij} - T_j^+)^2\right\}^{0.5}$

$i = 1, 2, \ldots, N \;\&\; j = 1, 2, \ldots, N$

$S_i^- = \{\sum_{j=1}^{n}(T_{ij} - T_j^-)^2\}^{0.5}$

i = 1, 2, ...,N& j = 1, 2, ... ,N

xii. Relative closeness:

This is the final step of this technique to evaluate the total distance which will help determine how the alternatives should be ranked.

(Relative Closeness)$_i$= $S_i^-/(S_i^+ - S_i^-)$

xiii. Ranking:

In this step, the ranking is provided on a priority basis among the relative closeness of the alternatives. If one alternative has having highest value then it will be given first rank and respectively other ranks will be provided to other alternatives.

TOPSIS additionally offers ideal and non-ideal solutions in addition to asserting the distance of selection alternatives from positive and negative ideal solutions [175]. Due to the following factors, TOPSIS is mostly employed in many contexts of multi-criteria group decision-making:

- It depends on the assumption that it gives the optimal outcome, regardless of how close it comes to the perfect solution being a positive ideal or a negative ideal individually.
- It is easy to understand, comprehensible, and empirical.
- Comparable to other techniques, it provides certain benefits. One of these benefits, the performance, is influenced in part by the quantity of options and driven by the expanding alternatives and ranking criteria.
- When a non-optimal alternative is entered, the rank of alternatives may also change.

TOPSIS also assists organizations in prioritizing and selecting projects by considering factors like financial feasibility, strategic fit, and resource availability. TOPSIS is a useful tool for decision-making because of its many benefits, which include its objectivity, simplicity, comprehensiveness, adaptability, ease of interpretation, and efficiency.

2.3.2 Preference Ranking Organisation Method for Enrichment Evaluation (PROMETHEE):

Preference ranking organization method for enrichment evaluation (PROMETHEE), often referred to as GAIA, is a theory that stems from comments on multi-criteria problems that

argue that this kind of problem cannot be resolved by including information on the decision-makers priorities and preferences [176]. The analysts and decision-makers can typically easily specify the relevant data from GAIA and PROMETHEE. It includes weights that explain each criterion's relative importance as well as a preference function associated with each one. The final alternatives' rankings, which call for a wide variety of criteria, can be achieved with the help of this technique. Additionally, an upgraded PROMETHEE II technique has been deployed, expanding its potential for use in solving this kind of decision-making problem with a variety of competing choices and criteria.

This method uses PROMETHEE I for a biased ranking of the activities based on positive and negative flows and PROMETHEE II for a comprehensive ranking of the acts [177]. Figure 2.2 depicts the PPROMETHEE technique's workflow. Similarly, like ELECTRE, PROMETHEE comes in a variety of forms, such as PROMETHEE I and II for partial and complete ranking issues, respectively. Later, PROMETHEE III and IV were created based on interval and continuous situations, respectively, by J.P. Brans and B. Mareschal [178]. Following years of investigation, other PROMETHEE extensions were found, including PROMETHEE V in 1992 and PROMETHEE VI in 1994, both of which were also proposed by J.P. Brans and B. Mareschal [178].

While PROMETHEE VI tackles issues with human brain representation, PROMETHEE V is specifically designed for issues with segmentation limits. In their comparison of the stability of ELECTRE III and PROMETHEE, J. P. Brans, P. Vincke, and B. Mareschal [179] discovered that PROMETHEE is more stable due to discontinuities in the preference functions and derivatives of ELECTRE III. PROMETHEE utilizes both qualitative and quantitative standards, much like ELECTRE.

PROMETHEE, on the other hand, has the advantage of expressing these requirements in its units and requiring fewer inputs, which reduces complexity and makes it easier. Thus, the PROMETHEE results could be challenging for the DM to assess if there are numerous criteria and options offered. The rank reversal issue may also affect PROMETHEE anytime a fresh option is presented. The following are the steps used in PROMETHEE, as summarised in [180]:

- Track down the evaluation matrix and compare the two items in pairs.
- Provide a preference function that takes values between 0 and 1 based on how different two pairs are from one another.

- Determine the rank by computing the global matrix.

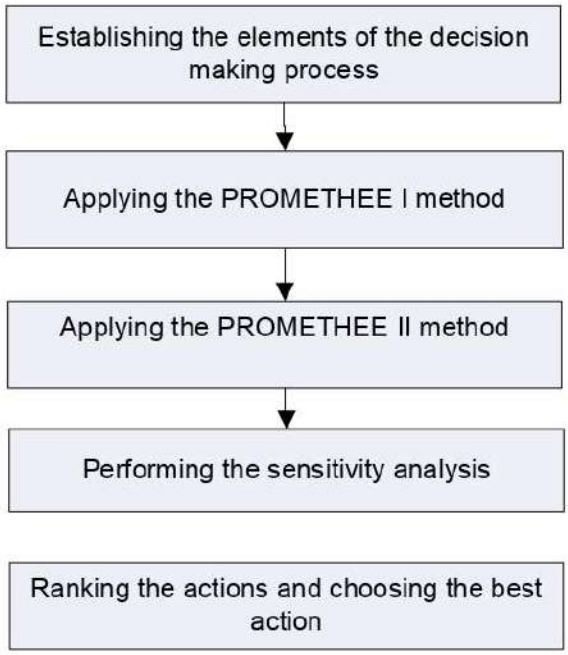


Figure 2.2:PROMETHEE technique workflow

Due to the thoroughness of its ranking, PROMETHEE is regarded as a simple outranking technique that has been successfully applied to real-world planning issues [181]. The most commonly used MCDM technique among various versions of PROMETHEE tends to be PROMETHEE-II as shown in Figure 2.3. This approach, which was introduced by [182], solves problems using pairwise comparison-based outranking. This method's qualities are comparable to ELECTRE's because both focus on outranking techniques. Here, pairwise comparisons are transformed into uni-criterion comparisons so that they can be computed against one another. Versions of PROMETHEE like ELECTRE are available.

The PROMETHEE I method is utilized for partial ranking of alternatives, the PROMETHEE II method for complete ranking, the PROMETHEE III method for ranking based on interval, the PROMETHEE IV method for total ranking, the PROMETHEE V method to address problems with segmentation constraints, and the PROMETHEE VI method to depict the human brain [183].

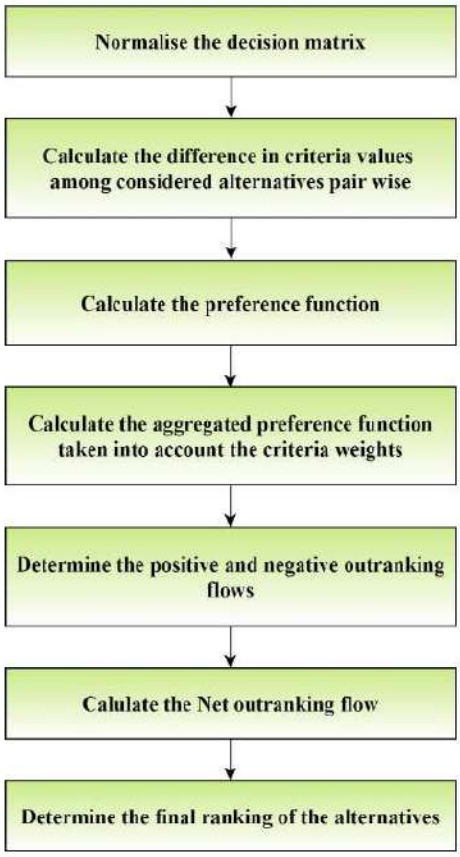


Figure 2.3: PROMETHEE-II workflow Diagram

PROMETHEE can assist in project management by ranking project goals, tasks, and resource distributions according to predefined criteria. PROMETHEE is dependent on a large amount of data, which can be time- and resource-intensive, including preference data and pairwise comparisons between alternatives.

2.4 SUMMARY

This chapter covers the review of the literature that was done about resource management in Fog computing and MCDM techniques. The review of the literature is split into two sections and the first section consists of four subparts to provide a better understanding of the research work. Whereas the second section covers the research work done using MCDM techniques. Most of the research work has employed QoE instead of incorporating its parameters in their evaluation. This leads to an increase in the importance of the QoE requirement for the smart applications deployed on the Fog environment. QoE also plays a crucial role in the efficient scheduling of the Fog computing environment. Shortly, with the rapid advancement of technology, the MCDM technique could be highly helpful for making complex judgments.

Earlier, the hybrid MCDM approaches have been applied specifically for a few issues. However, combining two or more strategies could potentially result in more effective decision-making when it comes to Resource management issues. This chapter elaborates on some of the key MCDM techniques that researchers have used in cloud and Fog computing. There is also a need for decision-making approaches for ranking the Fog computing environment using QoE parameters. The upcoming chapters will introduce practical methods designed to address these issues effectively.

CHAPTER 3

RESOURCE MAPPING AND RANKING IN FOG COMPUTING ENVIRONMENT

3.1 INTRODUCTION

According to the previous chapter demand for real-time device service providers has increased which requires immediate response for better QoE and performance of the Fog environment. This problem has been identified as a complex problem by various authors [36, 69,76] and has been solved to address this issue by assigning the desired smart application to the Fog environment. Several techniques for resource mapping have been adapted in the fog computing environment like linear programming, and improved game theory allocates the resources in the Fog computing environment. These methods have a high convergence rate and a significant method with less convergence rate is very useful. MCDM techniques have been of huge importance in solving such complex problems. This chapter proposes an integrated MCDM technique that can effectively perform two operations mapping and ranking of resources in a Fog environment. TOPSIS and AHP MCDM techniques have been used to improve the overall QoE and its performance.

3.2 PROPOSED FRAMEWORK

The work plan of the proposed framework is addressed in this section which works on the issue of assigning the desired smart application to a specific Fog environment resulting in better QoE. The assignment of the various smart applications with minimum latency to desired FSP matching its requirements is considered to be of much importance. Also, with the rise in the number of Fog nodes a reliable component out of many smart applications is required to perform the Fog computing environment mapping. The proposed framework provides better latency and efficient service with respective of all the performance metrics used. However, the quality of the smart applications requires the best resource for the execution of services. Thus, there is a need for such a framework that offers better mapping of the Fog environment based on the requirements of smart applications resulting in enhanced

performance. In this section, I have proposed two algorithms for the allocation and deallocation of the Fog environment to the smart application respectively.

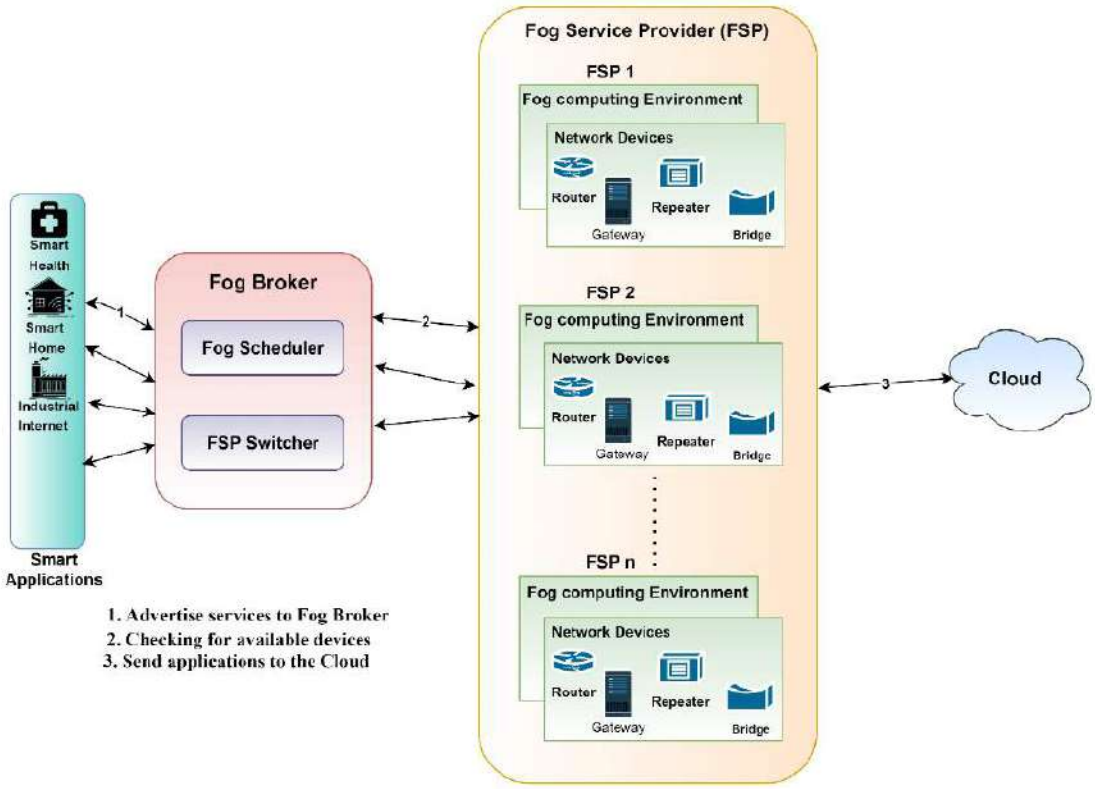


Figure 3.1: Proposed Framework

Figure 3.1 represents the proposed framework which is designed for choosing the appropriate Fog resource relative to a specific smart application. The proposed framework comprises three parts such as IoT devices, Fog brokers& Fog service providers (FSPs). The overall entities used in this framework are discussed below. Further, the section discusses the proposed framework entities in detail.

i. **Smart Applications:** for instance, smart watches, smart health monitoring systems, smart agriculture, traffic monitoring, etc. demand low latency services to provide good QoE to its users. These smart applications use the Fog computing environment resources for its low latency feature. However, the smart application generates requests that are forwarded to the Fog broker.

ii. **Fog broker:** is a mediator between the IoT devices and the Fog computing environment. This component relates the smart application with the FSP and consists of two sub-components which are the Fog scheduler and FSP switcher. The main aim of this component is to offer ranking to the Fog environment based on the information about Fog node availability. This data is refined by the improved TOPSIS technique.

iii. **FSP:** It contains the Fog computing environment which saves their data on the Fog nodes. Fog nodes perform all the processing of the Fog computing environment. The requested data from the smart applications is also provided by the Fog nodes. Fog computing other services like scalability, low latency, and reduced operation cost offer instant response to the smart applications. This component affirms the information exchange of the devices and based on application requests offers responses to them respectively.

iv. **Network Devices:** acts as an important component of the proposed framework. To avoid high latency, the computing resources are handled at the network edge which results in high efficiency as well. Some network devices such as gateway, router, and bridge establish the Fog environment and are elements in FSP.

3.2.1 Resource Ranking Approach

The resource ranking of the Fog environment is required for providing better services to the smart applications. In this section, the proposed resource mapping algorithm is explained in detail. A combination of TOPSIS and AHP methods has been employed for Fog environment ranking and allocation. The first MCDM technique has been used to evaluate the precise Euclidean distance which is the TOPSIS technique from positive and negative optimal solutions individually. The next technique used is for calculating the weights of the criteria which is the AHP technique.

3.2.2 Resource Mapping Algorithm

This section describes the best suitable Fog environment resource mapping for the smart application. Based on these ranks the allocation will be done to the smart applications as per their requests. The working steps of Algorithm 3.1 are described below. Initially, the Fog environments are stored in FN[ ] from the FSPs. For storing the resources associated with the

Fog environment, a new matrix is used as R[ ]. In this experiment, smart application parameters such as number of cores, time to finish, and network bandwidth are selected. After this Improved TOPSIS technique is applied and the resulting ranks are stored in RFN [ ]. The chosen smart application is given the highest rank available from the RFN[ ] of the Fog environment.

Algorithm 3.1: Resource Mapping Algorithm

Input:
-List of Fog Environments (FN)
- List of Resources associated with each Fog Environment (R)
- List of Smart Applications and their parameters (SA, SAP)
Output: RFN[ ]
Allocated resources for each Smart Application

BEGIN
Step 1: Store all the Fog Environments in FN[]
Step 2: Store all the resources associated with each Fog Environment in R[]
Step 3: for each Smart Application SA in SA:
a. Select SA based on Smart Application Parameters (SAP)
b. Apply Improved TOPSIS technique on Fog Environment and store ranks in RFN[]
c. Select the first rank of Fog Environment from RFN[] and assign it to the selected SA
d. Update the resources of the Fog Environment in R[]
e. If the Smart Application has utilized the allocated resource, release the resource
f. Update the released allocated resource in R[]
Step 4: Repeat steps 3a-3f for each Smart Application SA in SA until all resources are allocated
END

After this allocation of resources to the smart applications the remaining resources are modified in the matrix R[ ]. Now the smart applications will check the assigned resources and if there is any requirement of new resources. This check for new resources will be updated in R[ ]. The same process is updated for the smart application that needs instant response.

3.3 EXPERIMENTAL SETUP AND RESULTS

This section provides a brief explanation of the experimental testbed formed for the execution of the proposed framework and its results. This experiment has been carried out based on three criteria which are network bandwidth, no. of cores, and average latency. Improved TOPSIS technique is executed with the help of Python framework 3.7. This execution is made possible because of the designed testbed which is shown in Figure 3.2. A detailed study is provided in this section about the three components to better understand the proposed framework like Fog broker, synthetic data generator, and list of no. of cores used by Fog environment component.

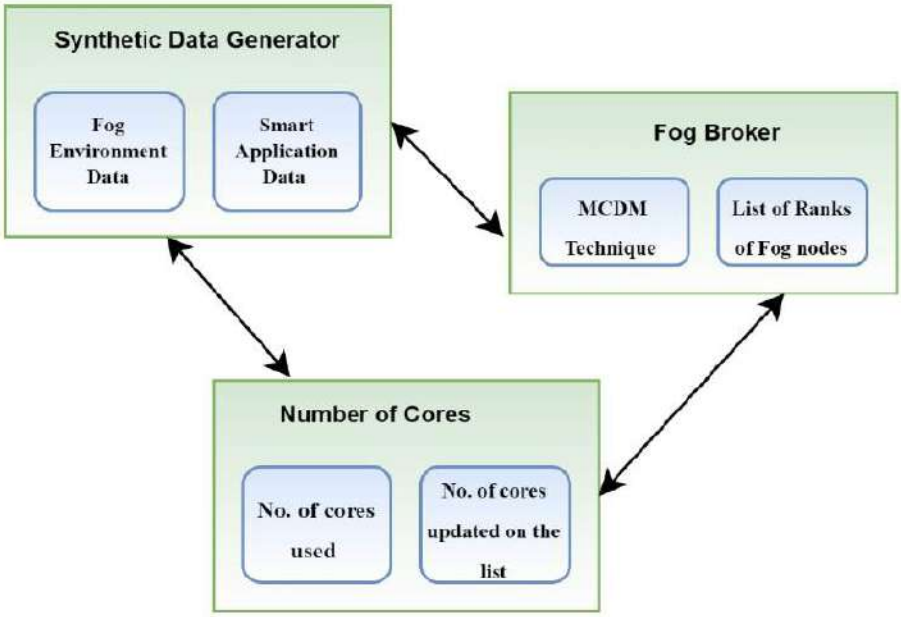


Figure 3.2: Proposed framework testbed

3.3.1 Synthetic Data Generation

The proposed framework requires a data set consisting of multiple Fog environments and smart application attributes. I have searched the internet for a dataset that imitates the required QoE parameters to examine the proposed framework. Several sites such as Data World. The UCI data repository has been thoroughly searched and no such data set has been discovered to meet the needs of this experiment. Therefore, a dataset has been designed using random processes for the Fog environment as shown in Table 3.2 [46, 136]. This newly

generated Fog environment dataset consists of QoE parameters such as average latency, number of cores, and network bandwidth are listed in Table 3.1. These three parameters are evaluated based on the correlation analysis [38]. By applying Pearson Correlation the correlation between average latency and number of cores is 0.143, indicating a very weak positive linear relationship. The correlation between average latency and network bandwidth is 0.145, indicating a very weak positive linear relationship. The correlation between number of cores and network bandwidth is 0.150, indicating a very weak positive linear relationship.

Spearman Correlation the correlation between average latency and number of cores is 0.091, indicating a very weak positive monotonic relationship. The correlation between average latency and network bandwidth is 0.030, indicating a very weak positive monotonic relationship. The correlation between number of cores and network bandwidth is 0.115, indicating a very weak positive monotonic relationship. The correlation coefficients are all close to zero, suggesting that average latency, number of cores, and network bandwidth are largely independent of each other in this dataset. This supports the claim that these factors are independent. For a more robust analysis, you would ideally want to use a larger and more diverse dataset, but these results indicate minimal correlations, aligning with the hypothesis of independence. The dataset generated is in CSV format and is provided as an input to the Python script and the output is stored in another CSV file.

Table 3.1 Simulation Parameters

Simulation Parameters	**Fog Environment**	**Smart Applications**
Network Bandwidth	1Mbps -20Mbps	2 Mbps- 5Mbps
Average Latency/ Time to finish	50μs – 80μs	10μs – 100μs
Number of cores required	10 CPU cores - 80 CPU cores	1 CPU cores – 25 CPU cores

Table 3.2 Dataset generated for Fog node

Fog Environment	**Network Bandwidth (Mbps)**	**Average Latency (μs)**	**No. of cores**

Fog Environment 1	19	25	17
Fog Environment 2	20	35	21
Fog Environment 3	1	26	61
Fog Environment 4	3	26	33
Fog Environment 5	4	55	20
Fog Environment 6	10	18	52
Fog Environment 7	2	37	73
Fog Environment 8	3	41	31
Fog Environment 9	14	55	20
Fog Environment 10	7	48	65
Fog Environment 11	7	42	67
Fog Environment 12	6	23	10
Fog Environment 13	11	47	71
Fog Environment 14	10	42	13
Fog Environment 15	12	16	63
Fog Environment 16	18	22	50
Fog Environment 17	2	30	38
Fog Environment 18	11	26	75
Fog Environment 19	14	42	16
Fog Environment 20	12	19	11
Fog Environment 21	12	42	40
Fog Environment 22	8	30	64
Fog Environment 23	10	26	20
Fog Environment 24	5	57	29
Fog Environment 25	10	17	21

Similarly, another dataset has been designed for measuring the requirements of smart applications to advertise the requirements of the smart application to the Fog environment. I also searched for a suitable smart application dataset on the internet with the required parameters. Thus, a smart application dataset was generated in CSV format as listed in Table 3.3. The QoE parameters considered are total core requirement, time to finish, and bandwidth requirement. The smart application requirement is sent to the Fog computing environment for allocation of Fog resources.

Table 3.3 Dataset generated for smart application

Smart Applications	**Minimum Network Bandwidth required (Mbps)**	**Time to Finish (μs)**	**Total core requires**
Smart Applications 1	3	95	1
Smart Applications 2	4	95	6
Smart Applications 3	2	73	4
Smart Applications 4	3	47	10
Smart Applications 5	2	89	1
Smart Applications 6	2	46	2
Smart Applications 7	5	25	5
Smart Applications 8	4	10	3
Smart Applications 9	5	82	6
Smart Applications 10	3	65	8
Smart Applications 11	3	13	5
Smart Applications 12	4	99	8
Smart Applications 13	4	57	2
Smart Applications 14	5	84	1

Smart Applications 15	2	72	19
Smart Applications 16	2	62	2
Smart Applications 17	4	41	11
Smart Applications 18	5	13	3
Smart Applications 19	3	74	2
Smart Applications 20	4	41	1
Smart Applications 21	3	100	5
Smart Applications 22	5	44	5
Smart Applications 23	2	64	25
Smart Applications 24	3	75	2

As listed in Table 3.3 the no. of cores required by the smart application 1 is 1, but suppose the allocated Fog environment has 17 cores as listed in Table 3.2. Now the Fog environment is left with 16 cores. The remaining number of cores is saved in the list of Python scripts. This list is revised when smart application 1 transmits for another Fog environment in some interval of time. The management of several cores is managed by the list whenever there is an unusual requirement by the smart application.

3.3.2 Fog Broker

In this section, the working of the Fog broker is addressed which acts as a mediator between the Fog node and the smart applications. Fog broker helps in ranking the Fog environments with the help of the MCDM technique. The processing of the Fog broker is done in the Python framework using QoE parameters of the Fog environment as shown in Figure 3.2. The evaluated weights of the parameters for each of them are as follows network bandwidth with 0.68601, average latency is with 0.17853 and number of cores is with 0.13544. These parameter weights are used as input to the Python script and output the rank of the Fog environment. The corresponding output is presented in Table 3.4.

Table 3.4 Ranks of Fog environment using MCDM technique

Sr. No.	Alternative	Relative Closeness	Rank

1	Fog Environment 1	0.72841	23
2	Fog Environment 2	0.30870	74
3	Fog Environment 3	0.45548	52
4	Fog Environment 4	0.28746	75
5	Fog Environment 5	0.43528	57
6	Fog Environment 6	0.20939	95
7	Fog Environment 7	0.32850	70
8	Fog Environment 8	0.72109	24
9	Fog Environment 9	0.43464	58
10	Fog Environment 10	0.78194	7
11	Fog Environment 11	0.20548	96
12	Fog Environment 12	0.49203	49
13	Fog Environment 13	0.24605	84
14	Fog Environment 14	0.25725	79
15	Fog Environment 15	0.56436	42
16	Fog Environment 16	0.62128	33
17	Fog Environment 17	0.17085	99
18	Fog Environment 18	0.63870	32
19	Fog Environment 19	0.36296	67
20	Fog Environment 20	0.61296	35
21	Fog Environment 21	0.26725	77
22	Fog Environment 22	0.38381	61
23	Fog Environment 23	0.74601	18
24	Fog Environment 24	0.77942	8

Table 3.5 Pair-wise comparison matrix

Alternatives	**Network Bandwidth**	**Average Latency**	**Number of cores**
Network Bandwidth	1	3	7
Average Latency	0.33	1	1
Number of cores	0.14	1	1

Table 3.5 represents the pairwise comparison matrix [38] for Fog environment parameters. With the help of this table, the relative importance of the criteria is calculated. The QoE requirements of the smart applications inspire the smart applications parameters and are beneficial in the processing of the Fog network.

3.3.3 Experimental Results

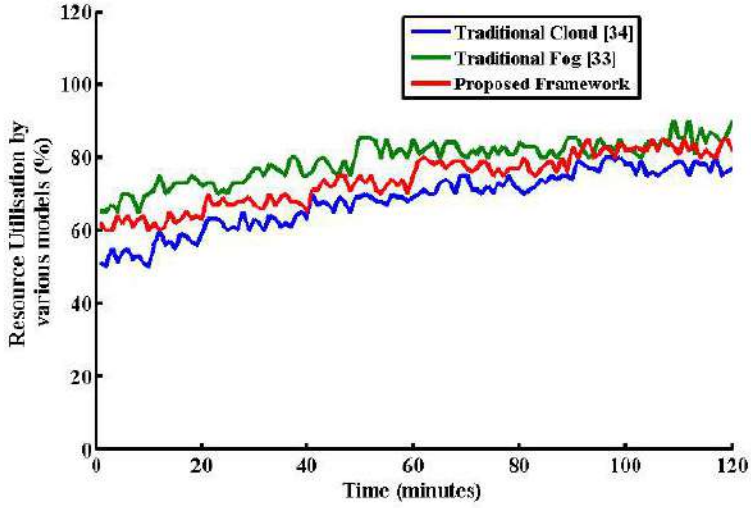
This section provides a brief understanding of the performance of the proposed framework which is measured using five metrics. These five metrics are response time, availability of resources, resource utilization, completion time, and waiting time. The proposed framework is compared with three computing models which are explained in detail below.

- Traditional cloud computing model: all the smart application service requests are directly dealing with the cloud environment for processing [187].
- Traditional Fog computing model: The entire smart application requests are sent to the Fog environment for resources. And the allocation of resources is done based on the available resources [119].
- Proposed Framework: the entire smart application request which is requesting resources from the Fog environment is provided on a priority basis. In this step, the Fog broker works as a mediator between these platforms.

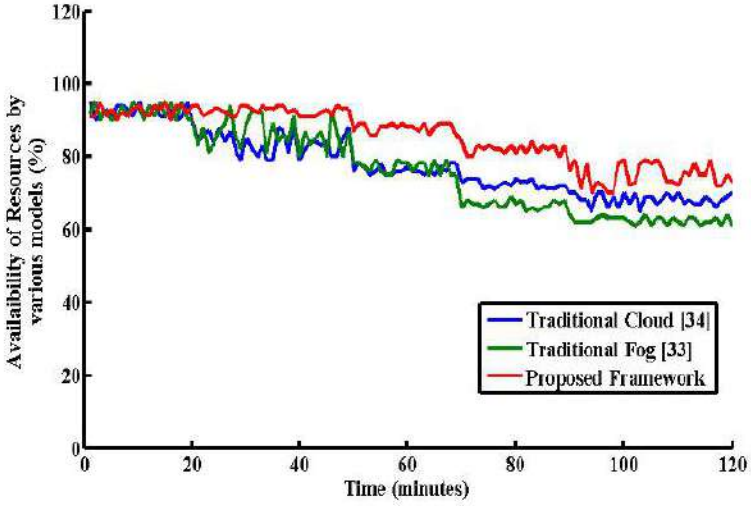
Figure 3.3 (a) part shows the resource utilization in all the computing models. The proposed model has 90% utilized the resources as compared to other models. The other models have a little difference and vary by 82% and 80% accordingly. However, high resource utilization has a serious impact on smart application performance. It was discovered that the utilization of resources must not exceed 75% [187]. Because after 75% there will be more heat produced by the system and a chance of damaging hardware would also be there [188]. Therefore, I have calculated the resource utilization after every hour of the day. The experiment continued for 5 days to better observe of results. Although the traditional Fog model had been not much available by the request of the smart applications.

Figure 3.3 (b) part shows the resources available for the models including the proposed model that is utilized for the experimental study. The number of available resources was high at the beginning of the allocation of resources but as the number of smart application requests increased the availability of the resources decreased. Due to the systematic scheduling in the proposed model, it has more available resources than the other two models. Figure 3.3 (c) represents the response time of the models. The proposed framework has a response time of around 15 seconds as compared to other models. Figure 3.3 (d) shows the waiting time of all three models. Due to minimum network latency traditional Fog computing model has less waiting time. However, after some time, the waiting time increased due to a small number of resources. Whereas the proposed model performs better with its efficient usage of resources. both the waiting time and the response time had an important impact on the completion time of the application. This signifies that the measured completion time is less in comparison with the other two models. Figure 3.3 (e) shows the completion time of the proposed model in comparison with the other two models. Since the proposed model is based on an improved

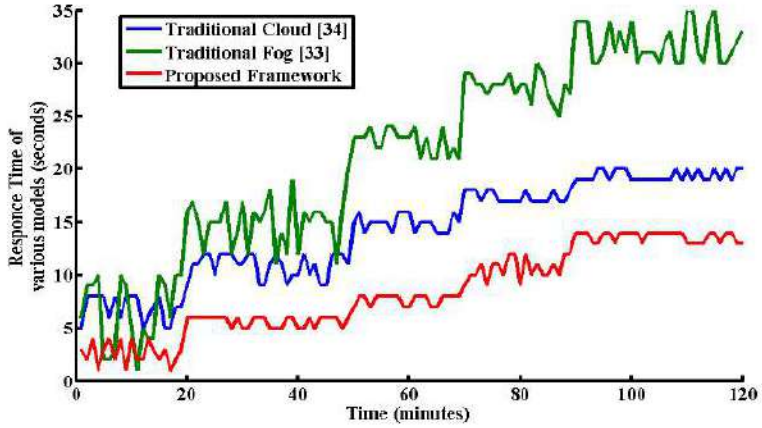
version of the Fog computing environment with less latency, so it's excellent smart application completion time.



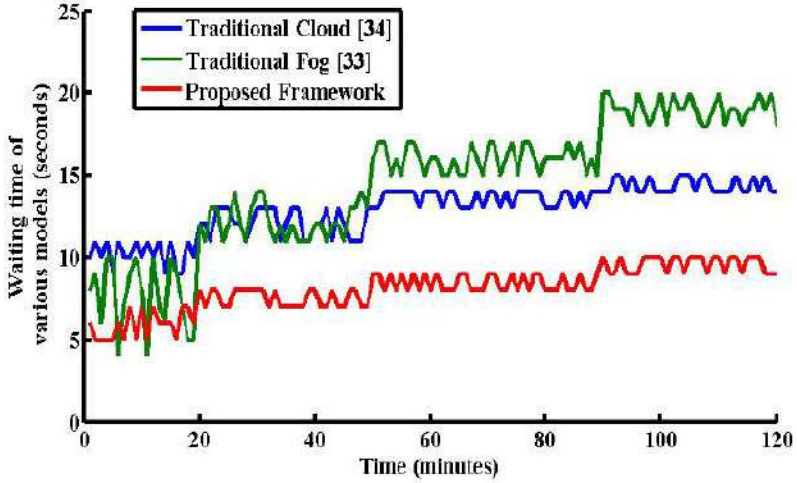
(a)



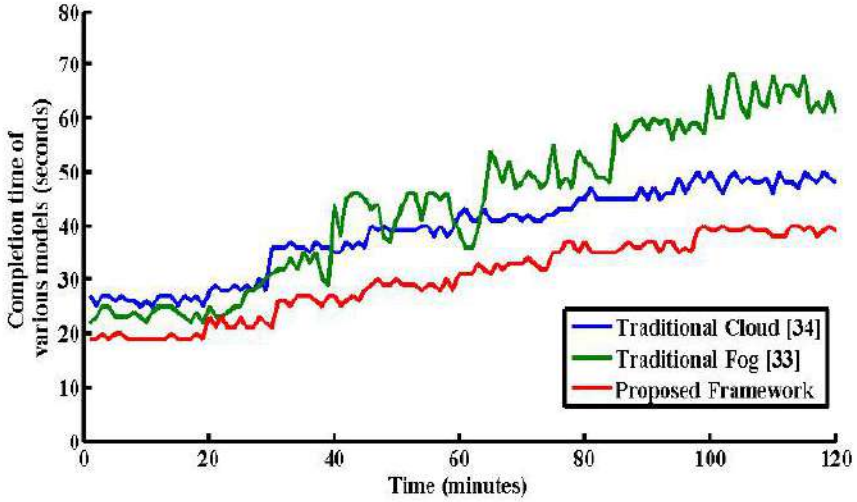
(b)



(c)



(d)



(e)

Figure 3.3: (a)Resource Utilisation of the proposed framework (b) Resource availability for all the models(c) Response Time for the considered models (d) Waiting of the resources in the three models (e) Completion time of the proposed framework

3.4 Conclusion

This chapter demonstrated the integrated MCDM approach for the allocation of smart applications to the Fog environment. The datasets used for the evaluation of the proposed framework are generated synthetically using random processes. These datasets are made for Fog environment and smart applications with three QoE parameters each. The Fog environment parameters are network bandwidth, no. of cores, and latency. Whereas, the smart application parameters are minimum network bandwidth, latency, and total number of cores. An improved TOPSIS technique-based framework is proposed to evaluate the datasets. The results demonstrate that the proposed framework performs better than the other two traditional cloud and Fog models respectively.

CHAPTER 4

AHP-BASED TECHNIQUE FOR RESOURCE MANAGEMENT IN FOG COMPUTING ENVIRONMENT

4.1 INTRODUCTION

With the recent development in IoT, it is growing to be the most commonly used technologies that enable new dynamics to improve the quality of services. To process the data of IoT applications, the Fog computing paradigm has appeared as a promising solution. The processing of the IoT applications is performed by the Fog computing nodes along with physical servers which are located in cloud data centers. Whereas, because of its resource limitations, dynamic nature, and uncertainty issues resource management has become one of the complex problems in the Fog environment.

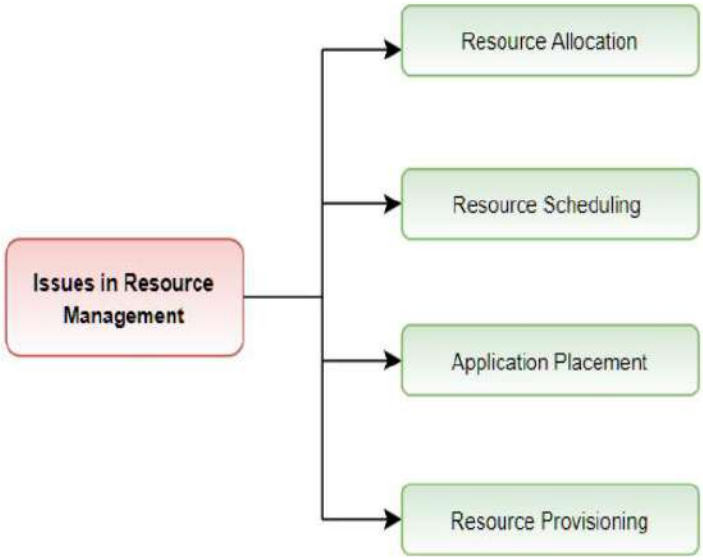


Figure 4.1 Taxonomy of Resource Management Issues

Although the Fog environment offers services to the smart applications for their customer based on their perception. Due to this, the QoE of the user is very poor leading to unsatisfied customers. Fog resource management is therefore necessary, although if any Fog computing infrastructure fails, the connections are moved to the nearby Fog environment. Data is sent from the previous Fog node to the new node [1] as a result of enabling a new connection. The

Fog infrastructure is more susceptible to trust problems as a result of failure in the Fog computing environment. To help the network resources closer to the user, Fog devices are employed at the network edge. Fog computing transfer's resources, such as the apps for billions of connected devices, at the central layer. This characteristic of Fog computing aids in the provision of edge-level storage, computation, and network infrastructure [2]. According to [3], the Fog computing environment also supports mobility, real-time interactions, interface heterogeneity, and scalability to latency-sensitive applications. The various resource management-related problems in Fog computing are depicted in Figure 4.1. In [4], these concerns are discussed in length. The four main problems are resource supply, application placement, resource scheduling, and resource allocation, as shown in Figure 4.1. These problems consequently make resource management in Fog computing of utmost importance. For the Fog computing environment, a significant amount of quality- and security-directed, trust-based resource management is needed.

4.2 PROPOSED FRAMEWORK

To meet the needs of various smart applications in the Fog computing environment, a resource management method has been devised. The suggested method enabled the smart applications to function correctly. The network topology, which is continually changing as new devices enter and leave the network, requires a dynamic computing environment for fog technology.

Figure 4.2: Process involved in Proposed Framework

The suggested method enables the processing of smart applications using several Fog resources. With the use of the MCDM technique, the primary goal of this work is to provide a better understanding of Fog resource allocation. The most popular methods for ranking priorities and choosing the best possible solution are MCDM techniques. One of the well-known MCDM techniques, AHP, is utilized to determine the rankings and scores of the options [5].

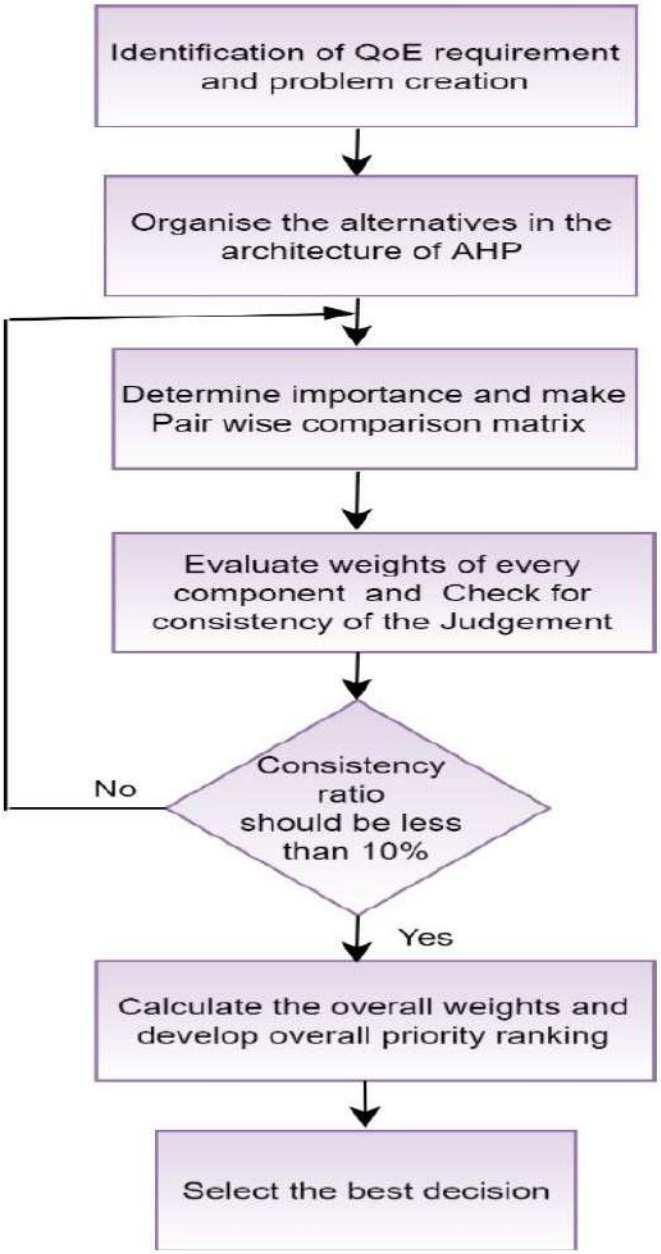


Figure 4.3: AHP Technique Flow diagram

Numerous smart applications mostly in a Fog environment exchange a vast amount of data. This increases the use of the processing, storage, and network resources in the Fog

environment. For study reasons, some QoE metrics have been taken into consideration, including network bandwidth, average latency, storage, and processing time. The evaluation of the priority ranking of the Fog resources is supported by these QoE metrics as well. The suggested framework uses a well-known MCDM technique to present a resource management strategy which is shown in Figure 4.2. One well-known MCDM technique is the AHP technique. According to the score values, the Analytical Hierarchy Process (AHP) determines the ranks of the parameters. Problem formulation comes first in the AHP technique, after which the alternatives are arranged in a hierarchy. The pairwise comparison matrix is then determined using the judgment values as a reference. The consistency of the judgment values is tested following this stage. The consistency ratio (CR) must be lower than 10% if its weights are determined; otherwise, the pair-wise matrix must be re-evaluated. Network bandwidth (NB), average latency (AL), storage (S), and processing time (PT) are some of the QoE parameters used during the experimental study. Figure 4.3 provides a detailed explanation of the AHP approach. In this section, a quick experiment is conducted to better figure out how to manage Fog resources using AHP.

4.2.1 Dataset Formation

Fog computing is a distributed computing platform that, as was already noted, offers on-site data processing and storage capabilities to carry out IoT services rather than sending them remotely to the cloud. It offers storage, computation, and networking resources, much like cloud computing. The dynamic, heterogeneous, and uncertain nature of the fog environment makes a resource management system necessary to realize the potential of fog computing. It is necessary to have a dataset with many different Fog Environment attributes to test the suggested framework. The dataset used to evaluate the proposed framework was obtained after a thorough search on the internet. As a result, several websites utilize the data from the UCI repository. The World, among other places, has been searched, but no dataset that satisfies the criteria of the Fog computing environment has been discovered. As a result, a dataset for the fog environment is produced utilizing random processes, as seen in Table 4.1 [26, 27].

Table 4.1 Dataset values for Fog environment

Fog Environment (F.E)	Network Bandwidth (Mbps)	Average Latency (μs)	Storage (TB)	Processing Time (μs)
FR1	3	16	95	5
FR2	4	24	95	9
FR3	2	24	73	6
FR4	3	21	47	7
FR5	2	20	89	5
FR6	2	19	46	10
FR7	5	17	25	8
FR8	4	12	10	10
FR9	5	18	82	8
FR10	3	18	65	5

The resource management issue falls within the MCDM category since it requires the highest priority of the Fog resource, which is afterward monitored for smart applications. For priority ranking, MCDM employs several strategies. The AHP approach is used in this work to determine rankings. In several fields, the priority ranking method known as AHP is used. Based on the significance of the QoE characteristics, a pair-wise comparison matrix is created. This matrix aids in evaluating the consistency of the conclusions reached during the process. The rankings of the Fog resources that are produced as a result of this approach further aid in choosing the superior Fog resource.

Table 4.2 Pairwise comparison matrix

Criteria	NB	AL	S	PT
NB	1	2	3	4
AL	0.5	1	4	5
S	0.33	0.14	1	3
PT	0.25	0.33	0.2	1

Table 4.3 List of Fog Resources ranks

S.No.	Alternative	Score	Rank
1.	FR1	0.20401	27
2.	FR2	0.23952	9
3.	FR3	0.19815	31
4.	FR4	0.15689	53
5.	FR5	0.20541	24
6.	FR6	0.14656	61
7.	FR7	0.12199	86
8.	FR8	0.08104	96
9.	FR9	0.20411	26
10.	FR10	0.16963	44

The pairwise comparison matrix of the QoE parameters created with the aid of the AHP technique is shown in Table 4.2. Additionally, a set of values including these four parameters—network bandwidth, average latency, storage space, and processing time are evaluated. The random distribution is used in Excel to evaluate the set of values made up of 99 Fog resources. Then, to better comprehend the experiment, the AHP approach is applied to this collection of values. This pairwise matrix is diagonal, and the values are determined using a rating scale that is based on the Saaty 2005 Theorem [6]. Table 4.3 displays the rankings of the Fog resources along with their scores. Once again, determining the priority ordering of the Fog resources is made much easier by using the AHP technique. Among various Fog resources, Fog resource 2 is ranked at position 9. When controlling the services for the smart application, this FR will be given priority over other alternatives.

4.3 PERFORMANCE ANALYSIS

This section describes the performance analysis of the proposed framework. Based on alternatives, a resource management strategy based on AHP is provided. Priority is first determined using QoE characteristics, as stated in Table 4.3. Second, four elements of network bandwidth, typical latency, storage, and processing time have been taken into account while evaluating the Fog resources. The score and ranks of the Fog resources were determined using these characteristics. The study of the score and rankings of the resources about the alternatives is shown in Figure 4.3.

Figure 4.4: Performance analysis of the proposed framework

Based on the values of the QoE parameters, 10 Fog resources are evaluated. The priority ordering of the Fog resources about the AHP approach would also vary if the decision values of the QoE parameters changed. The FR 2 has a low rank of 9 and a score of 0.23952 out of the 10 resources, as illustrated in Figure 4.4. The smart application user will initially monitor FR 2 in terms of other Fog resources. Examining the alternatives' wide range of flexibility can let the applications be scaled up and down more effectively. The AHP MCDM technique is used in this research to conduct a small experiment on the management of fog resources.

4.4 SENSITIVITY ANALYSIS

This method examines how changes in inputs impact the outputs of a model. Decision-makers can more easily perceive and manage uncertainties by determining which input has the most important impact on the outcome [192]. If the decision maker does not have any preponderant weightings in mind, sensitivity analysis is especially valuable [189]. The sensitivity analysis is used to determine how much variation in input values for a given variable affects the results of the proposed model. Regarding the weights of criteria, sensitivity analysis is used to assess the reliability and stability of the ranking. To analyze the AHP methodology presented in this study, a sensitivity analysis has been carried out [190]. Therefore, the first two alternative score values obtained from AHP are used to evaluate the

sensitivity constant for one criterion or factor, while the rest remain constant. In the First sensitivity analysis, FR1, and FR2 score values are used and checked with NB criteria [189]. By considering the first two alternatives as shown in Table 4.3 with their score values the value of δ comes out to be -0.1641. Therefore, when comparing the values of FR1 and FR2 the value of δ with criteria weights comes out to be feasible. These values come out to be a -ve value, which indicates the weights need to be increased by 37% for ranks to be reversed between FR 1 and FR 2. But in the case of Storage criteria, the values are not feasible. That means the sensitivity coefficient is set to zero if an alternative rank cannot be changed with any weight change. However, in the third case of sensitivity analysis for NB criteria for alternatives FR 1 and FR 4, the sensitivity coefficient value is 33.62 which is a +ve value. This +ve value indicates the weights should be decreased by 33% for the ranks to be reversed between FR 1 and FR 4. The AHP values for sensitivity analysis and score values of the alternatives are listed in Table 4.4 for evaluation.

Table 4.4 AHP Values for Sensitivity Analysis

Sensitivity Analysis No.	**Alternatives**	**Weights**	**Alternative weights**			
			NB 0.4326	**AL 0.3502**	**S 0.1378**	**PT 0.0791**
			Sensitivity coefficient			
1	**FR 1**	**0.4326**	-37.93	-28.92	Not Feasible	Not Feasible
	FR 2	0.3502				
2	**FR 1**	**0.4326**	-569.3	-313.02	-1880.2	Not Feasible
	FR 3	0.1378				
3	**FR 1**	**0.4326**	33.62	23.04	88.75	Not Feasible
	FR 4	0.0791				
4	**FR 2**	**0.3502**	Not Feasible	39.24	72.64	Not Feasible
	FR 3	0.1378				
5	**FR 2**	**0.3502**	Not Feasible	Not Feasible	Not Feasible	Not Feasible
	FR 4	0.0791				
6	**FR 3**	**0.1378**	Not Feasible	-177.32	Not Feasible	Not Feasible
	FR 4	0.0791				

5.5 CONCLUSION

Based on the values of the QoE parameters, 10 Fog resources are evaluated. The priority ordering of the Fog resources about the AHP approach would also vary if the decision values of the QoE parameters changed. The FR 2 has a low rank of 9 and a score of 0.23952 out of the 10 resources, as illustrated in Figure 4.4. The smart application user will initially monitor FR 2 in terms of other Fog resources. Examining the alternatives' wide range of flexibility can let the applications be scaled up and down more effectively. The AHP MCDM technique is used in this research to conduct a small experiment on the management of fog resources. Sensitivity analysis is done to check the robustness of the presented model. In this model, some values of weights need to be increased and other values are required to be decreased for the rank of two alternatives to be changed. This sensitivity analysis shows that the model with a small % change in criteria weights can reverse the ranking of alternatives.

CHAPTER 5
QoE BASED COST EFFECTIVE SCHEDULING IN FOG COMPUTING

5.1 INTRODUCTION

With the growing usage of real-time IoT devices, their price has been increasing rapidly. This growing demand gives rise to several problems like increasing cost of service providers, overloading of servers, scheduling issues of resources, and smart applications. The cost-of-service providers have become a major issue in placing the smart applications in the Fog environment. However, due to the absence of profit maximization and pricing methods, it becomes even more complicated for the service providers to gain from the Fog environment. To avoid such issues a cost-effective model is required for allocation of smart applications. Such type of model would help in assigning the smart application their desired Fog resource. Also, this model would result in better profit and QoE of the Fog environment.

This chapter addresses a cost-effective scheduling model in a Fog computing environment that would reduce the smart application cost from the user end and provide better profit by the Fog environment. To solve such a problem an MCDM-based model has been proposed for better efficiency and profit of the environment. To test the proposed framework a test bed containing three analysis phases is applied. This evaluation is done based on five metrics

- Average Allocation Time
- Average Profit by Fog Environment
- Average Cost of Smart Applications
- Resource Utilisation
- Number of Applications run within a given latency

5.2 PROPOSED FRAMEWORK

This section discusses the work plan of the proposed framework which evaluates the cost-effective scheduling framework. The proposed framework is designed with the help of the

MCDM technique. The proposed framework consists of three phases that perform the allocation of the Fog environment in the provided manner which are as follows:

- Resource Mapping Stage
- Latency Mapping Stage
- Cost Mapping Stage

These three stages are interconnected and one stage's output works as an input to the other. To evaluate this a dataset is required consisting of the QoE parameters such as bandwidth, latency, RAM and storage, number of cores, time to finish, and cost. I have synthetically generated a dataset to analyze the proposed framework. Experimentation of the proposed framework is done by comparing it with the other two models using five metrics. This experiment is performed on Python framework 3.7. The proposed model stages are described in the below sub-sections as shown in Figure 5.1.

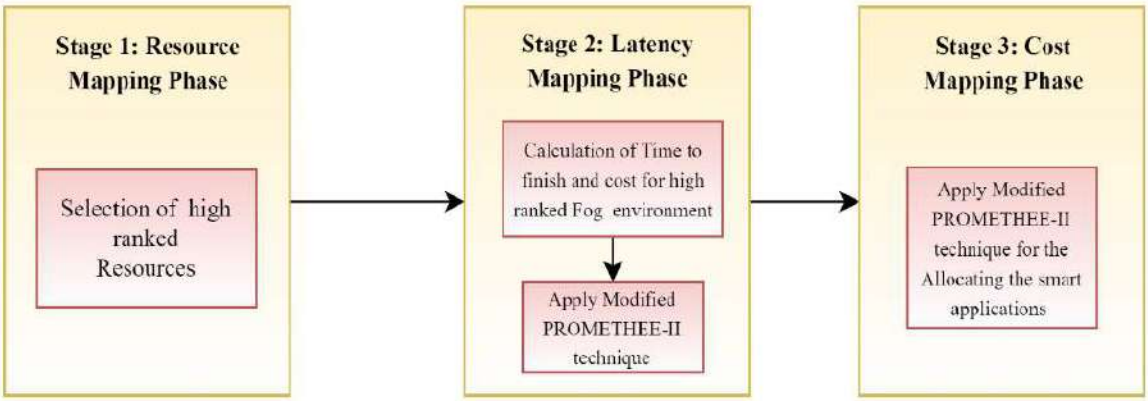


Fig. 5.1. Proposed framework architecture

5.2.1 Data Generation

This section explains a detailed insight into the process of generating a dataset for the evaluation of the proposed framework. I have searched for the Fog computing dataset that comprises parameters such as storage requirements and RAM capacity, uplink and downlink latency, no. of CPU cores required, etc. on sites like Kaggle, Data World, UCI repository, GitHub, etc. I did not find any dataset meeting the requirements. This situation led to the creation of Fog computing and smart application datasets with the required QoE parameters. I created this dataset with the help of random processes. For this experimental study, the Fog computing dataset includes QoE parameters which are RAM and storage requirements, uplink and downlink latency, time to finish, number of cores, cost, uplink, and downlink

bandwidth parameter values that are listed in Table 5.2. This dataset is generated with the help of some simulation parameters which are shown in Table 5.1.

Table 5.1: Simulation parameters used for Dataset

Simulations Parameters	Fog Environment	Smart Application
Uplink Bandwidth (Mbps)	1 – 100	1 – 100
Downlink Bandwidth (Gbps)	1 – 100	1 – 100
RAM requirement (GB)	2 – 10	2 – 10
Storage requirement (TB)	2 -16	2 -16
Uplink Latency (μs)	50– 100	50 – 100
Downlink Latency (μs)	50– 100	50 – 100
Number of Cores required (CPU cores)	10 – 80	10 – 80
Time to finish (μs)	60 – 100	60 – 100
Cost ($)	1 – 20	1 – 20

Table 5.2: Dataset for the Fog computing environment

Fog Environment (F.E)	RAM requirement (GB)	Storage requirement (TB)	No. of cores	uplink latency (μs)	downlink latency (μs)	uplink bandwidth (Mbps)	down link bandwidth (Gbps)	Time to Finish (μs)	Cost ($)
F.E 1	7	8	50	555	2677	27	74	81	3
F.E 2	8	10	43	335	2426	85	56	75	19
F.E 3	7	10	61	717	3504	97	78	11	28
F.E 4	6	5	17	501	3306	56	59	43	10
F.E 5	7	6	59	451	3123	29	35	76	16
F.E 6	7	2	43	688	4020	77	40	30	30
F.E 7	2	10	41	741	3297	25	74	43	12
F.E 8	2	9	68	506	3003	39	98	95	35
F.E 9	5	7	24	392	3439	75	36	27	25
F.E 10	5	10	21	659	3622	22	81	20	21
F.E 11	4	2	46	438	2896	86	69	35	16
F.E 12	6	10	45	400	2779	100	22	85	21
F.E 13	3	6	19	526	4048	76	6	40	17
F.E 14	6	2	63	643	2667	15	32	43	8
F.E 15	3	2	57	468	2807	9	38	80	34
F.E 16	8	3	28	496	2459	55	6	63	17
F.E 17	5	2	61	366	3636	72	79	84	29
F.E 18	6	8	24	624	3450	13	17	31	4
F.E 19	3	3	57	512	2783	66	58	83	14
F.E 20	6	2	43	305	3666	44	35	67	3

For the allocation of the Fog environment to the smart application, a smart application dataset with its parameter values is also required for its specification of services. Thus, a smart application dataset is required is required for the processing of the proposed framework. I have generated a smart applications dataset containing QoE parameters such as the average

runtime of the task, data to transfer, total core required, total time in which to complete, RAM and its storage requirements which are listed in Table 5.3.The smart application dataset serves as a major factor in demanding the resources from the Fog environment.

Table 5.3: Dataset for smart applications

Smart Application (S.A)	Data To Transfer (MB)	Avg. Runtime of Task (sec)	Total time in which to complete (sec)	Total Task	Total Core require	RAM (GB)	Storage (TB)
S.A 1	40	9	265	46	7	4	5
S.A 2	25	5	533	15	5	6	6
S.A 3	90	7	237	34	11	2	5
S.A 4	99	6	55	6	5	5	9
S.A 5	50	8	74	34	5	7	8
S.A 6	30	6	393	51	4	2	7
S.A 7	32	10	362	68	11	3	9
S.A 8	67	7	181	47	1	6	3
S.A 9	93	8	35	10	3	3	6
S.A 10	53	8	100	10	3	8	3
S.A 11	43	8	448	42	1	3	10
S.A 12	78	5	376	57	6	7	5
S.A 13	96	6	93	49	5	6	3
S.A 14	29	5	289	61	4	7	10
S.A 15	53	7	304	51	6	8	4
S.A 16	40	7	310	35	10	4	2
S.A 17	32	7	565	63	7	7	6
S.A 18	91	10	383	22	1	4	3
S.A 19	72	10	375	65	6	4	8
S.A 20	82	6	76	41	8	6	8

5.2.2 Resource Mapping Stage

In this section, the first stage of the proposed framework is elaborated in brief. In this stage, all the Fog resources are put together in a resource pool for filtering based on the demands of smart applications. The QoE parameters that are used for filtering the Fog resources are storage and RAM requirements, and the number of CPU cores. These two attributes are examined for allocating best-suited resources to the smart application. The first attribute is RAM and storage which informs about the size required for storing the data and the second attribute is total core requirements which helps in selecting only those CPU cores that are required for processing the data. For filtering out the Fog resources Algorithm 5.1 is applied and the steps are discussed in detail below:

Firstly, all the Fog environments values that are listed in the F.E [ ] note that all resources are meeting the requirements of S.A [ ]. These requirements are verified by the earlier two selected QoE parameters. In the next step the filtered F.E [ ] values are then stored in the Resr_Mapp [ ] buffer to proceed with the process of allocating the Fog environment. Whereas the left-out Fog environment values that do not meet the smart application requirements are excluded. This algorithm will be executed for all the smart applications to allocate them their desired Fog environment.

Algorithm 5.1: Resource Mapping Algorithm

Input:F.E[ ], SA[ ]
Output: Resr_Mapp[ ]
Begin
For values of F.E[ ]
    If resources of FE[ ] satisfy the requirement of SA[ ]
        Store the FE[ ] values in Resr_Mapp[ ]
    Else
        Discard the FE[ ] values which are not satisfying the requirements of SA[ ]
End For
End

5.2.3 Latency Mapping Stage

This section defines the working of the second stage of the proposed framework which is known as the Latency mapping stage. In this stage Algorithm 5.2 is applied for the further selection of Fog environment values for their allocation. For better allocation of resources scheduling time, data transfer time (T_d) by Eq.(5.1), and boot time are calculated. The scheduling time (T_s) here signifies the time needed for allocating applications with the processes. Boot time (T_b) is defined as the server time to handle the new application request. Boot time would be minimal if the server is running and handles the upcoming request. On the other hand, if the case server is free then it will take more time to process the requests. In this section time to finish (λ) is defined by the total time taken by the allocation process. λ is evaluated by the using Eq.(5.2) for Resr_Mapp[ ].

T_d= (Time in which data is sent * upward bandwidth)+ (Time in which data is received * downward bandwidth) (5.1)

$$\lambda = (\eta_r * (\eta / m)) + T_s + T_b + T_d \quad (5.2)$$

Algorithm 5.2: Latency Mapping Algorithm

Input: Resr_Mapp[ ], ToF [ ]
Output: Latency_Mapp[ ]
Begin
For all values in Resr_Mapp[ ]
    Calculate $\lambda = (\eta_r * (\eta / m)) + T_s + T_b + D_t$
    Store λ in ToF[ ]
For all values in ToF[ ]
Apply Algorithm 5.3 on ToF[ ]
Store the result values in RFE[ ]
For all values in RFE[ ]
Select the top rank values by n% in RFE[ ]
Store the selected top-rank values in Latency_Mapp[ ]
    Discard the rest of the RFE[ ] values
End For
End For
End For
End

The step-by-step description of Algorithm 5.2 is described as follows. In the first step input values are stored in Resr_Mapp[ ]along with this time to finish is evaluated with Eq. (5.2). All the calculated values are now stored in ToF[ ] (Time to Finish). The next step is to apply Algorithm 5.3 on ToF[ ] and the result is saved in RFE[ ]. This new buffer RFE[ ] consists of a list of Fog environment values in their respective rank order. To perform optimal placement of the Fog environment only a confined portion should be considered. This process would save a lot of time when compared with other processes which allocate all the Fog environment values to each smart application. Therefore, through this algorithm a small n% percent of Fog environment. I have selected 10% criteria for choosing the Fog environment values in this algorithm. The final batch of selected Fog environment values is saved in the Latency_Mapp[ ] buffer for advanced processing. Also, Algorithm 5.3 is explained in detail under section 5.2.3.1.

5.2.3.1 Modified PROMETHEE-II Algorithm

In this section, the detailed working of the Modified PROMETHEE-II technique is stated. This technique is applied to getting a rank-wise list of Fog environments. The QoE parameters considered for the analysis are the number of cores, RAM requirement, time to finish, storage requirement, uplink, and downlink latency, and bandwidth respectively. Although the PROMETHEE MCDM technique was first introduced by Bransvincke and Mareschal in 1986 [179] it relates to a type of outranking method. Algorithm 5.3 is improved

by integrating the PROMETHEE-II technique with the AHP technique. AHP MCDM method was first applied by [184] to calculate the weights of the criteria. This section further explains the steps involved in this technique that are mentioned in Algorithm 5.3. The overall complexity of this algorithm will be $O(m^2 n^2)$. Since the most time-consuming steps are Step 2, Step 3 and Step 5.

Step 1:

Let $FE = \{FE_1, FE_2, \ldots. FE_m\}$ be the set of n Fog environments and $P = \{P_1, P_2, \ldots. P_n\}$. The data containing all the alternatives and the criteria are better described in the form of a table with n*m evaluations. Every row defines the alternatives which are the Fog Environment and the criteria which are the QoE parameters.

$$FE_{m*n} = \begin{matrix} FE_1 \\ . \\ . \\ FE_m \end{matrix} \begin{bmatrix} P_1 & . & . & P_n \\ P_{11} & . & . & P_{1n} \\ . & . & . & . \\ P_{m1} & . & . & P_{mn} \end{bmatrix}_{m*n}$$

Step 2:

Now the preference value is calculated. Let $P_j(a)$ be the value of a criteria j for FE_a. The difference in the value of a criteria j for two decisions a and b is noted as $d(FE_a, FE_b)$.

$$d\ (FE_a, FE_b) = P_j\ (FE_a) - P_j\ (FE_b)$$

$P_j\ (FE_a, FE_b)$ represents the preference value of a criteria j for two decisions $\boldsymbol{FE_a}$ and $\boldsymbol{FE_b}$. The preference functions used to compute these preference values are described as

$$p_j\ (FE_a, FE_b) = F(d\ (FE_a, FE_b) \text{ with } \forall x\ \varepsilon\ ) -\infty,\infty\ [0 \leq F(x) \leq 1]$$

Step3:

In the PROMETHEE II, no method is described for the weights evaluation [177]. In this paper AHP method is used for weight evaluation.

Step4:

In this step, a global function is calculated with the help of criteria weights. Let Cr is a set of criteria and $W_j$is the weight associated with the criteria j. The global preference index for decisions a and b is defined as follows:

$$\prod(FE_{a,}FE_b) = \sum_{j \in P} W_j * P_j(FE_{a,}FE_b)$$

Step5:

Now for every decision a, the positive outranking flow $\emptyset^+(FE_a)$ and negative outranking flow $\emptyset^-(FE_a)$ is computed. Let A be the set of possible decisions and m is defined as the number of decisions. The positive outranking flow is evaluated by the following equation for decision a:

$$\emptyset^+(FE_a) = \sum_{x \in A} \pi(FE_a, x)$$

The negative outranking flow is evaluated by the following equation for decision a:

$$\emptyset^{-}(FE_a\mathrm{a}) = \sum_{x\in A}\pi(FE_a, x)$$

Step6:

To determine the trustworthiness a selection index is evaluated between the FE. The ranking is done based on outranking flows which is considered as a trustworthy index. The total outranking flow $\emptyset(FE_a)$ for a possible decision is computed as follows:

$$\emptyset(FE_a) = \emptyset^{+}(FE_a) - \emptyset^{-}(FE_a)$$

The higher the value of the decision the more trustworthy the decision is considered. In the trustworthy FE selection area where the ranks are calculated for best allocation, I will choose the maximum outranking flow of any particular decision.

Algorithm 5.3: Modified POMETHEE-II Algorithm

Input: FE, Attributes
Output: Best possible alternative
Begin
Step 1: Create a matrix between the FE and the attributes(m*n).
Step 2: The preference value is calculated (FE_a, FE_b).
Step 3: Weight evaluation is done by the AHP method.
Step 4: A Global function and global preference index are evaluated.
Step 5: For every decision positive $\emptyset^{+}(FE_a)$and negative$\emptyset^{-}(FE_a)$ outranking flow is evaluated
Step 6: Total outranking flow is evaluated to evaluate the best possible attribute.
End

5.2.4 Cost Mapping Stage

This section gives a detailed understanding of the last stage of the proposed framework. Algorithm 5.4 is applied to perform cost mapping of the proposed framework. The detailed working of Algorithm 5.4 is explained in detail below:

Initially, the output of Algorithm 5.2 that was stored in Latency_Mapp[ ] is given as an input to Algorithm 5.4. In the next step, cost is calculated for all the values of this buffer using Eq.(5.3) and is kept in the Cost_Mapp[ ] buffer. Now Algorithm 5.3 is used to evaluate the ranks of Cost_Mapp[ ] buffer for assigning the FE to the smart applications with its cost requirements. Later on, the values are filed in FE4 [ ] and after this top n% Fog environment values in FE4 [ ] are adopted, and the rest remaining values are discarded. If FE4 [ ] contains the first value of Fog environment as 1 then it will be allocated to the demanding smart application. This process will be repeated for all the values in FE4 [ ]. Now the allocated Fog environment is stored in the new buffer Allo_FE[ ]. Whereas, among this list, some Fog

environment values must be there that will not be allocated to smart applications. Thus, a new buffer WL [ ] will be maintained for these remaining values.

Table 5.4: Symbols used for Equations

Symbol	Representation
ϱ	Charge per minute for F.S.P
t_a	Application run time for F.S.P
η	Number of Tasks
η_r	Average runtime of tasks
ρ	Operational Cost
λ	Time to finish
m	Number of cores
a	Applications
ρ_c	Cost per core

$$\rho = \varrho * m * t_a \quad (5.3)$$
$$\rho_c = (\varrho * \eta * \eta_r) / m \quad (5.4)$$
$$t_a = (\eta * \eta_r) / m \quad (5.5)$$
$$\text{Using Eq (11) } m = (\eta * \eta_r) / t_a \quad (5.6)$$
$$\text{Cost} = \varrho * \eta * \eta_r \quad (5.7)$$
$$\text{Where } t_a = \eta * \eta_r \quad (5.8)$$
$$\text{Operational cost} = \varrho * t_a \quad (5.9)$$

Algorithm 5.4: Cost Mapp Algorithm

Input: Latency_Mapp[ ]
Output: WL [ ], FE4 [ ], Alloc_FE [ ]
Begin
For all values in Latency_Mapp[ ]
    Calculate $\rho = \varrho * m * t_a$
    Store ρ values in Cost_Mapp[]
For all values in Cost_Mapp[ ]
    Apply Algorithm 5.3 on the Cost_Mapp[ ]values
    Store the ranks of the values in FE4 [ ]
    Select the top 10% of high-rank values
Discard the rest of the values
If FE4 [ ] values =1
Allocate the FE with rank 1 to the SA
Store the allocated FE in Alloc_FE[ ]
Else
Store them in the waiting list WL [ ]
End For
End For
End For
End

5.3 EXPERIMENTAL RESULTS

In this section results of the proposed framework are provided based on the QoE parameters that are considered for Fog environment and smart applications. The experiment results of the proposed framework are evaluated in three parts. The first phase is Resource mapping analysis, the second is Latency mapping analysis and the third is Cost mapping analysis. The objective of this experiment is to provide smart applications their desired Fog resources so that it can provide better performance of their services. The whole experiment is performed on a 64-bit OS device, with processor Intel (R) Core (TM) i7 and 164GB RAM.

5.3.1 Resource Mapping Analysis

In this sub-section, the results of the Resource mapping stage of the proposed framework are presented. Suppose the first smart application (SA 1) in need of resources is with these specifications like minimum bandwidth requirements of 3MB, average runtime of task is 9 microsec, data to transfer is 40MB, total time in which to complete is 265microsec and total cores needed are 7. By applying Algorithm 5.1 as mentioned in section 5.2.2 for SA 1 on the Fog environment dataset as mentioned in Table 5.2 the filtered Fog environment values are listed in Table 5.5. This analysis phase is an important part of the allocation process because it saves time from allocating all the unnecessary Fog environment values.

Table 5.5: Filtered Fog Environment in Stage 1

Fog Environment (F.E)	RAM (GB)	Storage (TB)	No. of cores
F.E 1	7	8	50
F.E 2	8	10	43
F.E 12	6	10	45
F.E 13	3	6	19
F.E 14	6	2	63
F.E 20	6	2	43
F.E 24	6	8	57
F.E 31	8	2	41
F.E 34	2	10	58
F.E 35	7	7	37
F.E 42	4	5	24
F.E 43	2	5	71
F.E 44	6	7	78
F.E 53	6	9	32
F.E 54	3	6	42
F.E 55	5	5	50

5.3.2 Latency Mapping Analysis

This section presents the analysis results of the second stage of the proposed framework. The analysis results of the Resource mapping stage are provided as a starting point for Algorithm 5.2. This section states the results acquired from Algorithm 5.2 in the form of Table 5.6 and Table 5.7. Here, Table 5.6 represents the evaluated time Fog environment that is required to finish a process. Table 5.7 represents the output of the filtered Fog environment by applying Algorithm 5.2.

Table 5.6: Fog Environment parameter

Fog Environment (F.E)	RAM (GB)	Storage (TB)	No. of cores	uplink latency (μs)	downlink latency (μs)	uplink bandwidth (Mbps)	down link bandwidth (Gbps)	Time to finish (μs)
F.E 1	7	8	50	555	2677	27	74	81
F.E 2	8	10	43	335	2426	85	56	75
F.E 12	6	10	45	400	2779	100	22	85
F.E 13	3	6	19	526	4048	76	6	40
F.E 14	6	2	63	643	2667	15	32	43
F.E 20	6	2	43	305	3666	44	35	67
F.E 24	6	8	57	347	3495	73	65	26
F.E 31	8	2	41	647	2694	79	93	42
F.E 34	2	10	58	535	2498	22	61	97
F.E 35	7	7	37	269	3289	86	82	41
F.E 42	4	5	24	558	3686	55	30	30
F.E 43	2	5	71	749	2357	55	77	68
F.E 44	6	7	78	338	2682	59	18	44
F.E 53	6	9	32	742	3692	50	94	61
F.E 54	3	6	42	286	2855	42	82	58
F.E 55	5	5	50	707	2672	96	16	62

Table 5.7: Fog environment Ranks

Sr.	Alternative	Positive Flow	Negative Flow	Total Flow	Rank
1	F.E 1	0.13183	0.29055	-0.15871	78
2	F.E 2	0.04467	0.35292	-0.30825	98
3	F.E 12	0.10246	0.32437	-0.22191	90
4	F.E 13	0.27077	0.07840	0.19238	13
5	F.E 14	0.17965	0.20824	-0.02860	57
6	F.E 20	0.20754	0.08619	0.12135	25
7	F.E 24	0.06132	0.26759	-0.20627	85
8	F.E 31	0.11540	0.15714	-0.04173	61
9	F.E 34	0.33888	0.05655	0.28232	4
10	F.E 35	0.31139	0.04957	0.26182	5
11	F.E 42	0.35772	0.02084	0.33688	1
12	F.E 43	0.23029	0.11702	0.11328	26
13	F.E 44	0.11938	0.32167	-0.20230	84
14	F.E 53	0.15720	0.13859	0.01861	48
15	F.E 54	0.35174	0.05286	0.29887	3
16	F.E 55	0.17417	0.12769	0.04649	43

5.3.3 Cost Mapping Analysis

This section shows the results obtained from the Cost Mapping stage of the proposed framework. In this section Table 5.9 lists the results that are attained by applying Algorithm 5.4. Here, Table 5.8 shows the attribute values with their calculated cost for the Fog environment values that are listed in Table 5.6. Table 5.9 represents the values after employing Algorithm 5.3 in the Fog environment. The whole experimental analysis shows that FE1 is allocated to SA1 through the proposed framework.

Table 5.8: Fog environment new parameter values

Fog Environment	RAM (GB)	Storage (TB)	No. of cores	uplink latency (μs)	downlink latency (μs)	uplink bandwidth (Mbps)	down link bandwidth (Gbps)	Time to finish ((μs))	Cost ($)
F.E 1	7	8	50	555	2677	27	74	81	3
F.E 2	8	10	43	335	2426	85	56	75	19
F.E 12	6	10	45	400	2779	100	22	85	21
F.E 13	3	6	19	526	4048	76	6	40	17
F.E 14	6	2	63	643	2667	15	32	43	8
F.E 20	6	2	43	305	3666	44	35	67	3
F.E 24	6	8	57	347	3495	73	65	26	18
F.E 31	8	2	41	647	2694	79	93	42	21
F.E 34	2	10	58	535	2498	22	61	97	32
F.E 35	7	7	37	269	3289	86	82	41	11
F.E 42	4	5	24	558	3686	55	30	30	20
F.E 43	2	5	71	749	2357	55	77	68	29
F.E 44	6	7	78	338	2682	59	18	44	6
F.E 53	6	9	32	742	3692	50	94	61	23
F.E 54	3	6	42	286	2855	42	82	58	4
F.E 55	5	5	50	707	2672	96	16	62	10

Table 5.9: Final Fog environment Rank values

Sr.	Alternative	Positive Flow	Negative Flow	Total Flow	Rank
1	F.E 1	0.13183	0.29055	-0.15871	78
2	F.E 2	0.04467	0.35292	-0.30825	98
3	F.E 12	0.10246	0.32437	-0.22191	90
4	F.E 13	0.27077	0.07840	0.19238	13
5	F.E 14	0.17965	0.20824	-0.02860	57
6	F.E 20	0.20754	0.08619	0.12135	25
7	F.E 24	0.06132	0.26759	-0.20627	85
8	F.E 31	0.11540	0.15714	-0.04173	61
9	F.E 34	0.33888	0.05655	0.28232	4
10	F.E 35	0.31139	0.04957	0.26182	5
11	F.E 42	0.35772	0.02084	0.33688	1
12	F.E 43	0.23029	0.11702	0.11328	26
13	F.E 44	0.11938	0.32167	-0.20230	84
14	F.E 53	0.15720	0.13859	0.01861	48
15	F.E 54	0.35174	0.05286	0.29887	3
16	F.E 55	0.17417	0.12769	0.04649	43

5.4 PERFORMANCE ANALYSIS

In this section, I have discussed the testing of the proposed framework based on some performance metrics. These performance measures are allocation time, average profit by the Fog environment, average cost of application, number of applications run within a given latency, and resource utilization. The QoE parameter values for both the attributes that is Fog environment and smart application are developed using random distribution. The simulation run time for every individual application and resource is 120 minutes. Each metric results are combined in a 2D graph for better interpretation. Figure 5.2 displays the graphs by all the performance metrics. The further explanation of all the graphs of performance measures is described in detail in this section.

To allocate the smart application, the time occupied by the Fog environment is known as allocation time. Figure 5.2(a) shows that the allocation time of the proposed model is minimal in comparison with the other models. The allocation time is less in the proposed model because it is divided into three parts. Every stage of the model discards some amount of Fog environment which makes the allocation time less. Whereas the traditional Fog model shows high allocation time which is due to its random allocation of resources to the smart applications. At first, the traditional Fog model follows a first come basis to allocate its resources. But once the smart application requests are made in huge amounts, the system gets loaded leaving the Fog resource underutilized or maybe sometimes overutilized. This situation gives rise to the increase in allocation time of the Fog environment. However, the proposed model is useful in filtering the Fog environments which are an optimal choice for smart applications. The average allocation time of the proposed model is 2 seconds when compared with the traditional Fog model which has 7 seconds and the Enhanced Fog model has 4 seconds.

The response time of the Fog environment defines the average cost of the applications. The response time of the Fog environment specifies the cost of smart applications of the proposed model. However, in the traditional Fog model, the cost of smart applications varies with their performance with the allocated resources. Eq(12) is used for calculating the average cost of applications. Figure 5.2(b) shows the change in smart application cost during the experiment. Also, the waiting time for resources signifies the profit attained by the Fog environment. The waiting time means the Fog environment cost for smart applications. Where the Fog environment needs to assess the smart application in terms of cost. If the smart application cannot be executed by the fog environment, the waiting time will grow and the fog

environment won't make as much money. Some smart apps that are latency-sensitive run under their requested time in a typical Fog environment. However, some intelligent programs are unable to provide users with the needed services within a given time frame. All previously allotted resources are wasted as a result of this circumstance. On the other hand, the suggested model allocates better Fog resources in less time, which prevents resource underuse and overuse. The fog environment in Figure 5.2(c) is used to determine the greater profit.

Utilization of fog resources indicates that there are resources available for allocating to the fog environment. The suggested model uses 80% of the resources in Figure 5.2(d), whereas the classic and improved Fog models use 70% and 84% of the resources, respectively. Based on the available resources in the computer models being used, this chart shows the resources that have been consumed. The quantity of QoE-satisfied applications is declining as a result of the rising demand for application requests. These are determined by how long the applications' waiting times are. The number of applications seeking the customer service deadline by a user is depicted in Figure 5.2(e). Comparing the suggested model to the conventional and improved Fog models, the graph demonstrates that the greatest number of applications are meeting their timeframe in the proposed model.

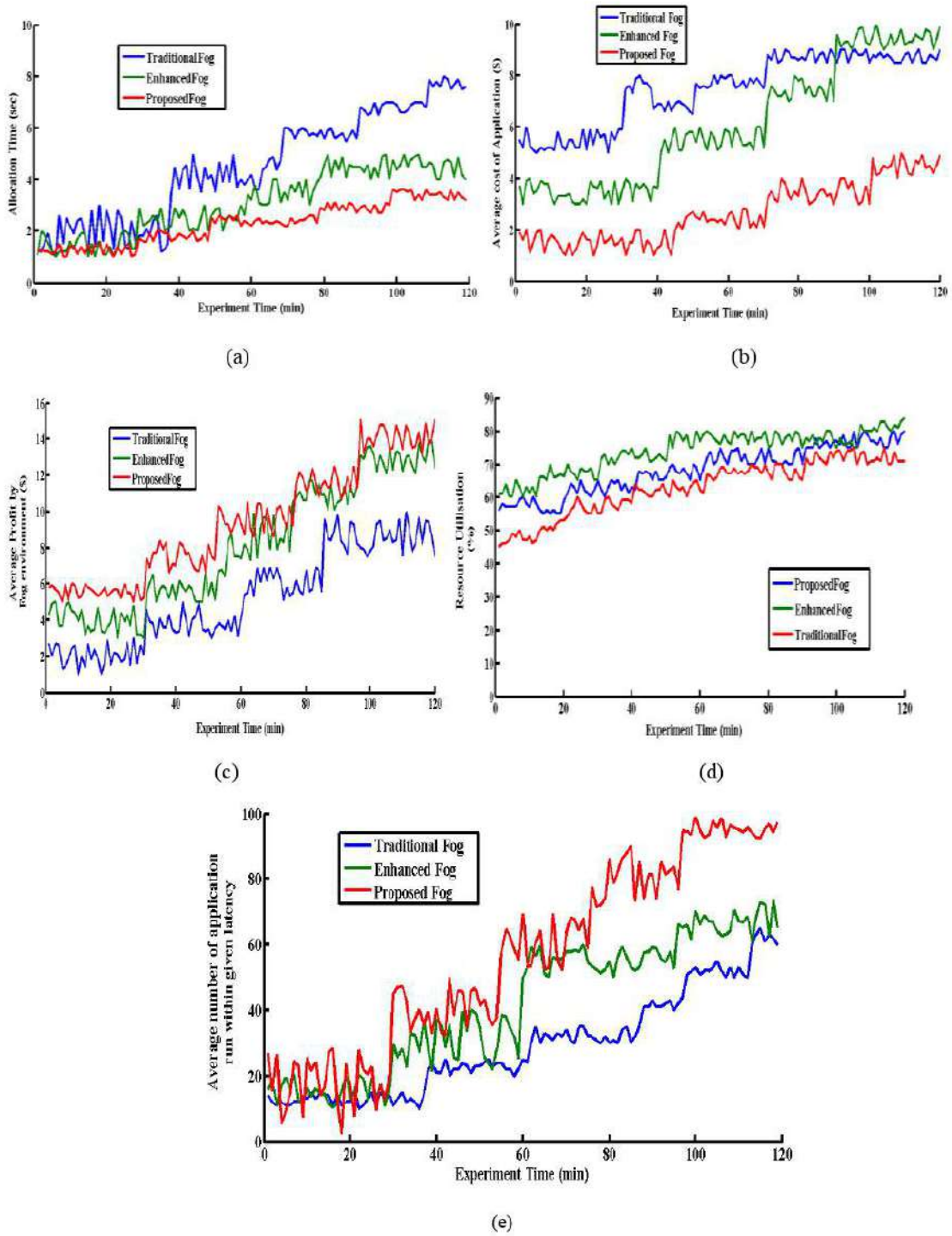


Figure 5.2: (a) Average Allocation time of Fog environment, (b) Average Cost of Applications, (c) Average Profit by Fog environment, (d) Resource Utilisation by Fog environment, (e) Average number of Application run within given latency

5.5 CONCLUSION

This section illustrates the use of the Multiple criteria decision technique to schedule Fog environments affordably. For the evaluation of the Fog environment, many QoE factors are taken into account, such as the amount of RAM and storage needed, the uplink and downlink latency, the uplink and downlink bandwidth, the number of cores necessary, the completion time, and the cost. To more effectively allocate the fog environment to the smart application, the suggested framework is split into three sections. These are utilizing Algorithms 5.1, 5.2, 5.3, and 5.4, respectively, to choose Fog resources at each stage. A scheduling combination of two MCDM approaches has been utilized to improve execution. By developing a four-phase experimental setup, the proposed framework is verified. The construction of a synthetic dataset helps in the more accurate calculation of QoE attribute values. Later, the two additional Fog models—traditional [] and enhanced—are used to compare the proposed framework. With the aid of several indicators, such as average allocation time, average cost of applications, average profit by the Fog environment, resource usage, and number of applications executed within the stipulated latency, it is possible to compare the suggested framework with the other two. The experimental finding demonstrates that the suggested structure performs better across the board for all five measures.

CHAPTER 6
CONCLUSION

This chapter concludes work done in this thesis concerning the Resource management and scheduling issues in Fog computing environment using different MCDM techniques. The next section of this chapter is a discussion of the main research contributions and potential future study areas.

6.1 CONTRIBUTIONS OF THE THESIS

The main contributions of this research work are listed below.

- Fog computing is thoroughly examined, along with its use, important problems, architecture, and research difficulties. By offering services to end devices, Fog computing reduces the load on the cloud. At the Fog layer, several other problems must be managed with effective methods, including resource management and scheduling. An analysis of the literature for various existing techniques has been done. A framework based on MCDM techniques has been developed to address the issue of Resource management in a Fog computing environment.
- Through the use of the AHP technique and an evaluation of the values of a few chosen QoE parameters in the Fog computing environment, developed a resource management approach that controls the various Fog resources. The specified experiment's performance is assessed based on the chosen QoE characteristics, such as network bandwidth, average latency, storage capacity, and processor speed. The suggested method applies the AHP technique to rank and score Fog resource data. The Fog computing environment can allocate its resources to intelligent apps following their needs thanks to this experimental investigation.
- Discussed an MCDM-TOPSIS-based framework for evaluating trust that takes into account the rankings of the QoE parameters and gives evaluated trust based on their values. The suggested architecture is built on the smart application's ability to request resources from the FSP while its performance is being tracked. The quality of experience provided by the FSP is encompassed by the numerous QoE criteria. The suggested approach fosters confidence in intelligent applications. The trust value is one of the factors taken into account when rating smart apps; the lower the rank, the higher

the trust value. These smart applications' trust values will be useful in further service communication to the Fog computing environment for processing. Research on updating smart application trust for performance improvement can be done in the future.

- An MCDM-based framework is presented uses a minimal amount of resources and offers an efficient method for resource ranking and resource mapping in the Fog computing environment. The modified TOPSIS algorithm distributes resources to smart apps that require them in real time based on QoE metrics including network bandwidth, average latency, and core count.
- A Cost-effective scheduling framework has been provided for arranging the resources of Fog computing environment by offering benefits to the users. To evaluate the Fog environment, several parameters are taken into account, including the amount of RAM and storage needed, up-link and down-link latency, up-link and down-link bandwidth, the number of required cores, the amount of time needed to complete the evaluation, and the cost. For better Fog environment allocation to the smart application, the suggested framework is split into three sections. Two MCDM strategies have been integrated into the scheduling process for better execution.

6.2 FUTURE WORK

Future research has several avenues opened up by this study. The cloud computing technique's shortcomings are addressed by the development of the fog computing methodology. The employment of artificial intelligence in the optimization approach will be the main component of the future strategy for sustainable directions. The use of artificial intelligence will aid in resolving the clogged issue with fog nodes and assist all businesses. The advantages of AI at the edge level are extended by the potential of fog computing in 5G enabled environments. Fog infrastructure can be either public or private. Fog infrastructure that is accessible to the general public is quite vulnerable to security risks. However, there is a lack of transparency in the operation of privately held Fog infrastructure. To assure cooperation and dependability amongst various forms of Fog computing infrastructure in this situation, a trusted service orchestration strategy is necessary. Fog computing is designed to run a variety of complicated IoT applications from many domains, such as smart cities, smart healthcare, smart agriculture, and smart industrial. These IoT applications need specialist support and have particular requirements. To manage them in Fog, application-specific management policies can be useful.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, Sep. 2013, doi: https://doi.org/10.1016/j.future.2013.01.010.

[2] J. An et al., "Toward Global IoT-Enabled Smart Cities Interworking Using Adaptive Semantic Adapter," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5753–5765, Jun. 2019, doi: https://doi.org/10.1109/jiot.2019.2905275.

[3] F. Cirillo, D. Gomez, L. Diez, I. Elicegui Maestro, T. B. J. Gilbert, and R. Akhavan, "Smart City IoT Services Creation Through Large-Scale Collaboration," IEEE Internet of Things Journal, vol. 7, no. 6, pp. 5267–5275, Jun. 2020, doi: https://doi.org/10.1109/jiot.2020.2978770.

[4] W. Li, T. Logenthiran, V.-T. Phan, and W. L. Woo, "A Novel Smart Energy Theft System (SETS) for IoT based Smart Home," IEEE Internet of Things Journal, pp. 1–1, 2019, doi: https://doi.org/10.1109/jiot.2019.2903281.

[5] D. Shin, K. Yun, J. Kim, P. V. Astillo, J.-N. Kim, and I. You, "A Security Protocol for Route Optimization in DMM-Based Smart Home IoT Networks," IEEE Access, vol. 7, pp. 142531–142550, 2019, doi: https://doi.org/10.1109/access.2019.2943929.

[6] V. K. Quy, V. H. Nam, D. M. Linh, N. T. Ban, and N. D. Han, "Communication Solutions for Vehicle Ad-hoc Network in Smart Cities Environment: A Comprehensive Survey," Wireless Personal Communications, Aug. 2021, doi: https://doi.org/10.1007/s11277-021-09030-w.

[7] F. Kiani et al., "Adaptive Metaheuristic-Based Methods for Autonomous Robot Path Planning: Sustainable Agricultural Applications," Applied Sciences, vol. 12, no. 3, p. 943, Jan. 2022, doi: https://doi.org/10.3390/app12030943.

[8] Muhammad Ammad et al., "A Novel Fog-Based Multi-Level Energy-Efficient Framework for IoT-Enabled Smart Environments," vol. 8, pp. 150010–150026, Jul. 2020, doi: https://doi.org/10.1109/access.2020.3010157.

[9] C. K. Metallidou, K. E. Psannis, and E. A. Egyptiadou, "Energy Efficiency in Smart Buildings: IoT Approaches," IEEE Access, vol. 8, pp. 63679–63699, 2020, doi: https://doi.org/10.1109/access.2020.2984461.

[10] S. Rani, S. H. Ahmed, and S. C. Shah, "Smart Health: A Novel Paradigm to Control the Chickungunya Virus," IEEE Internet of Things Journal, pp. 1–1, 2018, doi: https://doi.org/10.1109/jiot.2018.2802898.

[11] Z. Zhou, H. Yu, and H. Shi, "Human Activity Recognition Based on Improved Bayesian Convolution Network to Analyze Health Care Data Using Wearable IoT Device," IEEE Access, vol. 8, pp. 86411–86418, 2020, doi: https://doi.org/10.1109/access.2020.2992584.

[12] A. Chowdhury, G. Karmakar, and J. Kamruzzaman, "The co-evolution of cloud and iot applications: Recent and future trends," in Handbook of Research on the IoT, Cloud Computing, and Wireless Network Optimization. IGI Global, 2019, pp. 213–234.

[13] A. Rezaeipanah, M. Mojarad, and A. Fakhari, "Providing a new approach to increase fault tolerance in cloud computing using fuzzy logic," International Journal of Computers and Applications, vol. 44, no. 2, pp. 139–147, Jan. 2020, doi: https://doi.org/10.1080/1206212x.2019.1709288.

[14] M. Ghobaei-Arani and A. Shahidinejad, "An efficient resource provisioning approach for analyzing cloud workloads: a metaheuristic-based clustering approach," The Journal of Supercomputing, Apr. 2020, doi: https://doi.org/10.1007/s11227-020-03296-w.

[15] P.Mell and T.Grance, "The NIST Definition of Cloud Computing. National Institute of Standards and Technology", NIST Special Publication,pp.800-145, 2011.

[16] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds," ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, p. 50, 2008, doi: https://doi.org/10.1145/1496091.1496100.

[17] M. Yannuzzi, R. Milito, R. Serral-Gracia, D. Montero, and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing," 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Dec. 2014, doi: https://doi.org/10.1109/camad.2014.7033259.

[18] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599–616, Jun. 2009, doi: https://doi.org/10.1016/j.future.2008.12.001.

[19] H. Madsen, Grigore Albeanu, B. Burtschy, and Florin Popentiu-Vladicescu, "Reliability in the utility computing era: Towards reliable Fog computing," International Conference on Systems, Signals and Image Processing, Jul. 2013, doi: https://doi.org/10.1109/iwssip.2013.6623445.

[20] O. S. Albahri et al., "Systematic Review of Real-time Remote Health Monitoring System in Triage and Priority-Based Sensor Technology: Taxonomy, Open Challenges, Motivation and Recommendations," Journal of Medical Systems, vol. 42, no. 5, Mar. 2018, doi: https://doi.org/10.1007/s10916-018-0943-4.

[21] A. Vahid Dastjerdi et al. "Fog computing: Principles, architectures, and applications". In: Internet of things. Elsevier, 2016, pp. 61–75.

[22] H. Wadhwa and R. Aron, "Fog Computing with the Integration of Internet of Things: Architecture, Applications and Future Directions," 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), Dec. 2018, doi: https://doi.org/10.1109/bdcloud.2018.00144.

[23] M. IANSITI and K. B. CLARK, "Integration and Dynamic Capability: Evidence from Product Development in Automobiles and Mainframe Computers," Industrial and Corporate Change, vol. 3, no. 3, pp. 557–605, 1994, doi: https://doi.org/10.1093/icc/3.3.557.

[24] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," Software: Practice and Experience, vol. 32, no. 2, pp. 135–164, 2002, doi: https://doi.org/10.1002/spe.432.

[25] C.Ernemann, V.Hamscher, U.Schwiegelsholn, R.Yahyapour, and A. Streit, "On Advantages of Grid Computing for Parallel Job Scheduling," June 2003, doi: https://doi.org/10.1109/ccgrid.2002.1017110.

[26] M. Abu Sharkh, M. Jammal, A. Shami, and A. Ouda, "Resource allocation in a network-based cloud computing environment: design challenges," IEEE Communications

Magazine, vol. 51, no. 11, pp. 46–52, Nov. 2013, doi: https://doi.org/10.1109/mcom.2013.6658651.

[27] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-Aware Application Scheduling in Fog Computing," IEEE Cloud Computing, vol. 4, no. 2, pp. 26–35, Mar. 2017, doi: https://doi.org/10.1109/mcc.2017.27.

[28] A. Yousefpour et al., "All one needs to know about fog computing and related edge computing paradigms: A complete survey," Journal of Systems Architecture, Feb. 2019, doi: https://doi.org/10.1016/j.sysarc.2019.02.009.

[29] R. Mahmud, F.LuizKoch, and R. Buyya. "Cloud-fog interoperability in IoT-enabled healthcare solutions". In: Proceedings of the 19th international conference on distributed computing and networking, pp. 1–10, 2018.

[30] Y. Jie, C. Guo, K.-K. R. Choo, C. Z. Liu, and M. Li, "Game-Theoretic Resource Allocation for Fog-based Industrial Internet of Things Environment," IEEE Internet of Things Journal, pp. 1–1, 2020, doi: https://doi.org/10.1109/jiot.2020.2964590.

[31] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog Orchestration for Internet of Things Services," IEEE Internet Computing, vol. 21, no. 2, pp. 16–24, Mar. 2017, doi: https://doi.org/10.1109/mic.2017.36.

[32] O. Skarlat,V. Karagiannis, T. P. Rausch, K. Bachmann, and S. Schulte-Merker. "A framework for optimization, service placement, and runtime operation in the fog." In 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC), pp. 164-173. IEEE, 2018.

[33] I. Fister et al., "Novelty search for global optimization," Applied Mathematics and Computation, vol. 347, pp. 865–881, Apr. 2019, doi: https://doi.org/10.1016/j.amc.2018.11.052..

[34] M. Ayoubi, M. Ramezanpour, and R. Khorsand, "An autonomous IoT service placement methodology in fog computing," Software: Practice and Experience, vol. 51, no. 5, pp. 1097–1120, Dec. 2020, doi: https://doi.org/10.1002/spe.2939..

[35] I. Martinez, A. S. Hafid and A. Jarray, "Design, Resource Management, and Evaluation of Fog Computing Systems: A Survey," in IEEE Internet of Things Journal, vol. 8, no. 4, pp. 2494-2516, 15 Feb.15, 2021, doi: 10.1109/JIOT.2020.3022699

[36] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of Experience (QoE)-aware placement of applications in Fog computing environments," Journal of Parallel and Distributed Computing, vol. 132, pp. 190–203, Oct. 2019.

[37] J. Araujo, P. Maciel, E. Andrade, G. Callou, V. Alves, and P. Cunha, "Decision making in cloud environments: an approach based on multiple-criteria decision analysis and stochastic models," Journal of Cloud Computing, vol. 7, no. 1, Mar. 2018.

[38] 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, no. 2, pp. 197–218, Apr. 2017.

[39] F. Bonomi, R. Milito, Jiang Zhu, and S. Addepalli. "Fog computing and its role in the internet of things." In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp. 13-16. 2012.

[40] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter." In 2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops), pp. 105-110. IEEE, 2015.

[41] M. Aazam, M. St-Hilaire, C.-H. Lung, I. Lambadaris, and E.-N. Huh. "IoT resource estimation challenges and modeling in fog." In Fog Computing in the Internet of Things, pp. 17-31. Springer, Cham, 2018.

[42] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing." In 2015 IEEE International Conference on Communications (ICC), pp. 3909-3914. IEEE, 2015.

[43] C. T DO, N. H. Tran, C. Pham, Md. G. R. Alam, J. S. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing." In 2015 International Conference on Information Networking (ICOIN), pp. 324-329. IEEE, 2015.

[44] K. D. Ahmed, and S. R. M.Zeebaree. "Resource allocation in fog computing: A review." International Journal of Science and Business 5, no. 2 (2021): 54-63.

[45] T. Alam, "A reliable communication framework and its use in internet of things (IoT)." CSEIT1835111| Received 10,pp.450-456, 2018.

[46] S. Saroja and T. Revathi, "Performance Oriented Task- Resource Mapping and Scheduling in Fog Computing Environment," Cognitive Systems Research, Jul. 2021, doi: https://doi.org/10.1016/j.cogsys.2021.07.004.

[47] 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 124 (2017): 1-21.

[48] D. Tychalas and H. Karatza, "A Scheduling Algorithm for a Fog Computing System with Bag-of-Tasks Jobs: Simulation and Performance Evaluation," Simulation Modelling Practice and Theory, vol. 98, p. 101982, Jan. 2020, doi: https://doi.org/10.1016/j.simpat.2019.101982.

[49] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya. "Fog computing: Principles, architectures, and applications" In Internet of Things: Principles and Paradigms. Morgan Kaufmann, pp. 61–75,2016. DOI:https://doi.org/10.1016/ B978-0-12-805395-9.00004-6

[50] S. Sarkar, S. Chatterjee, and S. Misra. 2015. Assessment of the suitability of fog computing in the context of Internet of Things. IEEE Transactions on Cloud Computing PP, 99 (2015), 1.

[51] C.-H. Hong and B. Varghese, "Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms," vol. 52, no. 5, p. 97, Sep. 2019.

[52] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "FogBus: A Blockchain-based Lightweight Framework for Edge and Fog Computing," Journal of Systems and Software, vol. 154, pp. 22–36, Aug. 2019, doi: https://doi.org/10.1016/j.jss.2019.04.050.

[53] R. Mahmud, K. Ramamohanarao, and R. Buyya. "Edge affinity-based management of applications in fog computing environments" In Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (UCC'19), pp. 1–10, ACM, New York, NY,2019.

[54] P. G. V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, and R. Buyya,"Focan: A fog-supported smart city network architecture for management of applications in the internet of everything environments", Journal of Parallel and Distributed Computing 132, pp. 274–283, 2019.

[55] R. Mahmud and R. Buyya, "Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit," Fog and Edge Computing, pp. 433–465, Jan. 2019, doi: https://doi.org/10.1002/9781119525080.ch17.

[56] E. C. Pinto Neto, G. Callou, and F. Aires,"An algorithm to optimise the load distribution of fog environments",In 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)', IEEE, pp. 1292–1297, 2017.

[57] V. Cardellini, V. Grassi, Francesco Lo Presti, and M. Nardelli, "On qos-aware scheduling of data stream applications over fog computing infrastructures", In '2015 IEEE Symposium on Computers and Communication (ISCC)', IEEE, pp. 271–276, 2015.

[58] M. Ghobaei-Arani, R. Khorsand, and M. Ramezanpour, "An autonomous resource provisioning framework for massively multiplayer online games in cloud environment," Journal of Network and Computer Applications, vol. 142, pp. 76–97, Sep. 2019, doi: https://doi.org/10.1016/j.jnca.2019.06.002..

[59] S. El Kafhali and K. Salah, "Efficient and dynamic scaling of fog nodes for IoT devices," The Journal of Supercomputing, vol. 73, no. 12, pp. 5261–5284, Jun. 2017, doi: https://doi.org/10.1007/s11227-017-2083-x.

[60] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized IoT service placement in the fog," Service Oriented Computing and Applications, vol. 11, no. 4, pp. 427–443, Oct. 2017, doi: https://doi.org/10.1007/s11761-017-0219-8.

[61] R. Yu, G. Xue, and X. Zhang, "Application provisioning in fog computing-enabled internet-of-things: A network perspective", In IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 783-791, IEEE, 2018.

[62] R. Yadav, and G. Baranwal, "Trust-aware framework for application placement in fog computing", In 2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp. 1-6, IEEE, 2019.

[63] R. R. Kumar, S. Mishra, and C. Kumar, "A Novel Framework for Cloud Service Evaluation and Selection Using Hybrid MCDM Methods," Arabian Journal for Science and Engineering, vol. 43, no. 12, pp. 7015–7030, Nov. 2017, doi: https://doi.org/10.1007/s13369-017-2975-3.

[64] G. Baranwal, R. Yadav, and D. P. Vidyarthi, "QoE Aware IoT Application Placement in Fog Computing Using Modified-TOPSIS," Mobile Networks and Applications, Aug. 2020, doi: https://doi.org/10.1007/s11036-020-01563-x.

[65] G. Baranwal and D. P. Vidyarthi, "FONS: a fog orchestrator node selection model to improve application placement in fog computing," The Journal of Supercomputing, vol. 77, no. 9, pp. 10562–10589, Mar. 2021, doi: https://doi.org/10.1007/s11227-021-03702-x.

[66] V. B. Souza et al., "Towards a proper service placement in combined Fog-to-Cloud (F2C) architectures," Future Generation Computer Systems, vol. 87, pp. 1–15, Oct. 2018, doi: https://doi.org/10.1016/j.future.2018.04.042.

[67] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), May 2017, doi: https://doi.org/10.23919/inm.2017.7987464.

[68] Z. A. Mann, "Decentralized application placement in fog computing," IEEE Transactions on Parallel and Distributed Systems, pp. 1–1, 2022, doi: https://doi.org/10.1109/TPDS.2022.3148985.

[69] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-Aware Application Module Management for Fog Computing Environments," ACM Transactions on Internet Technology, vol. 19, no. 1, pp. 1–21, Mar. 2019, doi: https://doi.org/10.1145/3186592.

[70] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost Efficient Resource Management in Fog Computing Supported Medical Cyber-Physical System," IEEE Transactions on Emerging Topics in Computing, vol. 5, no. 1, pp. 108–119, Jan. 2017, doi: https://doi.org/10.1109/TETC.2015.2508382.

[71] L. Yang, J. Cao, G. Liang, and X. Han, "Cost Aware Service Placement and Load Dispatching in Mobile Cloud Systems," IEEE Transactions on Computers, vol. 65, no. 5, pp. 1440–1452, May 2016, doi: https://doi.org/10.1109/tc.2015.2435781.

[72] A.R.Benamer, H.Teyeb, and N. B. H. Alouane, "Latency-aware placement heuristic in fog computing environment," InOTM Confederated International Conferences On the Move to Meaningful Internet Systems, pp. 241–257, Springer, 2018.

[73] G.-J. Xiong, R. Singh, and J. Li, "Learning Augmented Index Policy for Optimal Service Placement at the Network Edge.," Jan. 2021.

[74] M.-Q. Tran, D. T. Nguyen, V. A. Le, D. H. Nguyen, and T. V. Pham, "Task Placement on Fog Computing Made Efficient for IoT Application Provision," Wireless Communications and Mobile Computing, vol. 2019, pp. 1–17, Jan. 2019, doi: https://doi.org/10.1155/2019/6215454.

[75] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, "Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing," vol. 29, no. 16, pp. e3975–e3975, Oct. 2016, doi: https://doi.org/10.1002/cpe.3975.

[76] S. Venticinque and A. Amato,"A methodology for deployment of IoT application in fog,"Journal of Ambient Intelligence and Humanized Computing, vol 10 ,pp1955–1976, 2019. https:// doi. org/ 10. 1007/ s12652- 018- 0785 4

[77] V. B. Souza et al., "Towards a proper service placement in combined Fog-to-Cloud (F2C) architectures," Future Generation Computer Systems, vol. 87, pp. 1–15, Oct. 2018, doi: https://doi.org/10.1016/j.future.2018.04.042.

[78] M. Selimi, L. Cerdà-Alabern, F. Freitag, L. Veiga, A. Sathiaseelan, and J. Crowcroft, "A Lightweight Service Placement Approach for Community Network Micro-Clouds," Journal of Grid Computing, vol. 17, no. 1, pp. 169–189, Feb. 2018, doi: https://doi.org/10.1007/s10723-018-9437-3.

[79] F. Khosroabadi, F. F. Ghazvini, H. Fotouhi,"SCATTER: service placement in realtime fog-assisted IoT Networks," J Sens Actuator Netw, vol. 10, 2021. https:// doi. org/ 10. 3390/ jsan1 00200 26

[80] T. Xiao, T. Cui, S. M. R. Islam, and Q. Chen, "Joint Content placement and storage allocation based on federated learning in F-RANs," Sensors, vol. 21, pp. 215, 2020. https:// doi. org/ 10. 3390/ s2101 0215.

[81] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Profit-aware application placement for integrated Fog–Cloud computing environments," Journal of Parallel and Distributed Computing, vol. 135, pp. 177–190, Jan. 2020, doi: https://doi.org/10.1016/j.jpdc.2019.10.001.

[82] C.-H. Hong and B. Varghese, "Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms," vol. 52, no. 5, p. 97, Sep. 2019.

[83] W.-S. Kim and S.-H. Chung, "User incentive model and its optimization scheme in user-participatory fog computing environment," Computer Networks, vol. 145, pp. 76–88, Nov. 2018, doi: https://doi.org/10.1016/j.comnet.2018.08.011.

[84] F. Concone, G. L. Re, and M. Morana, "A Fog-Based Application for Human Activity Recognition Using Personal Smart Devices," ACM Transactions on Internet Technology, vol. 19, no. 2, pp. 1–20, Apr. 2019, doi: https://doi.org/10.1145/3266142.

[85] M. Avgeris, D. Dechouniotis, N. Athanasopoulos, and S. Papavassiliou, "Adaptive resource allocation for computation offloading: A control-theoretic approach,"ACM Transactions on Internet Technology,vol.19, no. 2, pp.1-20, 2019.

[86] M. Aazam, M. St-Hilaire, C. H. Lung, I. Lambadaris, and E.-H. Huh, "Iot resource estimation challenges and modeling in fog," in Fog Computing in the Internet of Things, Springer, pp. 17–31, 2018.

[87] Z. Pooranian, M. Shojafar, P. G. V. Naranjo, L. Chiaraviglio, and M. Conti, "A Novel Distributed Fog-Based Networked Architecture to Preserve Energy in Fog Data Centers," 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Oct. 2017, doi: https://doi.org/10.1109/mass.2017.33.

[88] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1216–1228, Oct. 2017, doi: https://doi.org/10.1109/jiot.2017.2709814.

[89] M. Naas, P. R. Parvédy, J. Boukhobza, and L. Lemarchand, "iFogStor: An IoT Data Placement Strategy for Fog Infrastructure," May 2017, doi: https://doi.org/10.1109/icfec.2017.15.

[90] K. Velasquez, David Perez Abreu, Marilia Curado, and E. Monteiro, "Service placement for latency reduction in the internet of things," vol. 72, no. 1–2, pp. 105–115, Jun. 2016, doi: https://doi.org/10.1007/s12243-016-0524-9.

[91] B. V. Natesha and R. M. R. Guddeti, "Adopting elitism-based Genetic Algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment," Journal of Network and Computer Applications, p. 102972, Jan. 2021, doi: https://doi.org/10.1016/j.jnca.2020.102972.

[92] A. Basu and S. Ghosh, "Implementing Fuzzy TOPSIS in Cloud Type and Service Provider Selection," Advances in Fuzzy Systems, vol. 2018, pp. 1–12, Nov. 2018, doi: https://doi.org/10.1155/2018/2503895.

[93] A. Kiani and N. Ansari, "Toward Hierarchical Mobile Edge Computing: An Auction-Based Profit Maximization Approach," IEEE Internet of Things Journal, vol. 4, no. 6, pp. 2082–2091, Dec. 2017, doi: https://doi.org/10.1109/jiot.2017.2750030.

[94] M. Afrin, Md. Razzaque, I. Anjum, M. Hassan, and A. Alamri, "Tradeoff between User Quality-Of-Experience and Service Provider Profit in 5G Cloud Radio Access Network," Sustainability, vol. 9, no. 11, p. 2127, Nov. 2017, doi: https://doi.org/10.3390/su9112127.

[95] 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, Jul. 2019, doi: https://doi.org/10.1016/j.compeleceng.2019.05.010.

[96] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption," IEEE Internet of Things Journal, pp. 1–1, 2016, doi: https://doi.org/10.1109/jiot.2016.2565516.

[97] H. Wadhwa and R. Aron, "TRAM: Technique for resource allocation and management in fog computing environment," The Journal of Supercomputing, vol. 78, no. 1, pp. 667–690, May 2021, doi: https://doi.org/10.1007/s11227-021-03885-3.

[98] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation Resource Allocation and Task Assignment Optimization in Vehicular Fog Computing: A Contract-Matching Approach," IEEE Transactions on Vehicular Technology, vol. 68, no. 4, pp. 3113–3125, Apr. 2019, doi: https://doi.org/10.1109/tvt.2019.2894851.

[99] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos,"ENORM: A framework for edge node resource management," In: IEEE transactions on services computing, pp. 1–1, 2019.

[100] R. K. Naha, S. Garg, A. Chan, and S. K. Battula, "Deadline-based dynamic resource allocation and provisioning algorithms in Fog-Cloud environment," Future Generation Computer Systems, vol. 104, pp. 131–141, Mar. 2020, doi: https://doi.org/10.1016/j.future.2019.10.018.

[101] S. K. Sood and K. D. Singh, "SNA Based Resource Optimization in Optical Network using Fog and Cloud Computing," Optical Switching and Networking, vol. 33, pp. 114–121, Jul. 2019, doi: https://doi.org/10.1016/j.osn.2017.12.007.

[102] Z. Chang, L. Liu, X. Guo, and Q. Sheng, "Dynamic Resource Allocation and Computation Offloading for IoT Fog Computing System," IEEE Transactions on Industrial Informatics, pp. 1–1, 2020, doi: https://doi.org/10.1109/tii.2020.2978946.

[103] L. Duo, Q. Li, H. Xu, and Y. Zhou, "Dynamic Priority-Based Service Resource Allocation for Context-Aware Conflict Resolution in Wisdom Network with Fog Computing," Wireless Communications and Mobile Computing, vol. 2020, pp. 1–7, Aug. 2020, doi: https://doi.org/10.1155/2020/8812482.

[104] X. Chen, Y. Zhou, L. Yang, and L. Lv, "Hybrid fog/cloud computing resource allocation: Joint consideration of limited communication resources and user credibility," Computer Communications, vol. 169, pp. 48–58, Mar. 2021, doi: https://doi.org/10.1016/j.comcom.2021.01.026.

[105] K. Zhang, M. Peng, and Y. Sun, "Delay-Optimized Resource Allocation in Fog based Vehicular Networks," IEEE Internet of Things Journal, pp. 1–1, 2020, doi: https://doi.org/10.1109/jiot.2020.3010861.

[106] M. Abbasi, E. Mohammadi-Pasand, and M. R. Khosravi, "Intelligent workload allocation in IoT–Fog–cloud architecture towards mobile edge computing," Computer Communications, vol. 169, pp. 71–80, Mar. 2021, doi: https://doi.org/10.1016/j.comcom.2021.01.022.

[107] C.-C. Lin and J.-W. Yang, "Cost-Efficient Deployment of Fog Computing Systems at Logistics Centers in Industry 4.0," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4603–4611, Oct. 2018, doi: https://doi.org/10.1109/tii.2018.2827920.

[108] D. Rahbari and M. Nickray, "Scheduling of fog networks with optimized knapsack by symbiotic organisms search," In: 2017 21st Conference of Open Innovations Association (FRUCT). IEEE. 2017, pp. 278–283, 2017.

[109] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," Software: Practice and Experience, vol. 47, no. 9, pp. 1275–1296, Jun. 2017, doi: https://doi.org/10.1002/spe.2509.

[110] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System," IEEE Transactions on Computers, vol. 65, no. 12, pp. 3702–3712, Dec. 2016, doi: https://doi.org/10.1109/TC.2016.2536019.

[111] F. Murtaza, A.Akhunzada, S.ul Islam, J.Boudjadar, and R.Buyya, "QoS-aware service provisioning in fog computing," Journal of Network and Computer Applications,vol. 165 , p. 102674, 2020.

[112] X.-Q. Pham, N. D. Man, N. D. T. Tri, N. Q. Thai, and E.-N. Huh, "A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing," International Journal of Distributed Sensor Networks, vol. 13, no. 11, p. 155014771774207, Nov. 2017, doi: https://doi.org/10.1177/1550147717742073.

[113] J. Fan, X. Wei, T. Wang, T. Lan, and S. Subramaniam, "Deadline-Aware Task Scheduling in a Tiered IoT Infrastructure," Global Communications Conference, Dec. 2017, doi: https://doi.org/10.1109/glocom.2017.8255037.

[114] Yadav, Ashish Mohan, Kuldeep Narayan Tripathi, and Subhash Chander Sharma. "An enhanced multi-objective fireworks algorithm for task scheduling in fog computing environment." Cluster Computing 25, no. 2 (2022): 983-998.

[115] L. Yin, J. Luo, and H. Luo. 2018. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. IEEE Transactions on Industrial Informatics 14, 10 (Oct. 2018), 4712–4721.

[116] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing," Concurrency and Computation: Practice and Experience, vol. 32, no. 7, Nov. 2019, doi: https://doi.org/10.1002/cpe.5581.

[117] K. Fizza, N. Auluck, O. Rana, and L. Bittencourt, "PASHE: Privacy aware scheduling in a heterogeneous fog environment," In Proceedings of the 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud'18), pp. 333–340, 2018.

[118] R. Ding, X. Li, X. Liu, and J. Xu, "A cost-effective time-constrained multi-workflow scheduling strategy in fog computing," In Service-Oriented Computing—ICSOC 2018 Workshops. Lecture Notes in Computer Science, vol. 11434. Springer, pp. 194–207.

[119] M. A. Benblidia, B. Brik, L. Merghem-Boulahia, and M. Esseghir," Ranking fog nodes for tasks scheduling in fog-cloud environments: A fuzzy logic approach," In Proceedings of the 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC'19),pp.1451–1457.

[120] Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. Zhou, "MEETS: Maximal Energy Efficient Task Scheduling in Homogeneous Fog Networks," IEEE Internet of Things Journal, vol. 5, no. 5, pp. 4076–4087, Oct. 2018, doi: https://doi.org/10.1109/JIOT.2018.2846644.

[121] S. Wang, T. Zhao, and S. Pang, "Task Scheduling Algorithm Based on Improved Firework Algorithm in Fog Computing," IEEE Access, vol. 8, pp. 32385–32394, 2020, doi: https://doi.org/10.1109/access.2020.2973758.

[122] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic Scheduling for Stochastic Edge-Cloud Computing Environments using A3C learning and Residual Recurrent Neural Networks," IEEE Transactions on Mobile Computing, pp. 1–1, 2020, doi: https://doi.org/10.1109/tmc.2020.3017079.

[123] R. Vijayalakshmi, V. Vasudevan, Seifedine Kadry, and R. Kumar, "Optimization of makespan and resource utilization in the fog computing environment through task scheduling algorithm," vol. 18, no. 01, pp. 1941025–1941025, Jan. 2020, doi: https://doi.org/10.1142/s021969131941025x.

[124] G. Zhang, F. Shen, N. Chen, P. Zhu, X. Dai, and Y. Yang, "DOTS: Delay-Optimal Task Scheduling Among Voluntary Nodes in Fog Networks," vol. 6, no. 2, pp. 3533–3544, Apr. 2019, doi: https://doi.org/10.1109/jiot.2018.2887264.

[125] L. ZUO and L. ZUO, "Multi-objective integrated ant colony optimization scheduling algorithm based on cloud resource," Journal of Computer Applications, vol. 32, no. 6, pp. 1916–1919, Aug. 2013, doi: https://doi.org/10.3724/sp.j.1087.2012.01916.

[126] C. Shetty and H. Sarojadevi, "Framework for Task scheduling in Cloud using Machine Learning Techniques," Jan. 2020, doi: https://doi.org/10.1109/icisc47916.2020.9171141.

[127] D. RAHBARI and M. NICKRAY, "Low-latency and energy-efficient scheduling in fog-based IoT applications," TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES, pp. 1406–1427, Mar. 2019, doi: https://doi.org/10.3906/elk-1810-47.

[128] X. Q. Pham and E. N. Huh, "Towards task scheduling in a cloud-fog computing system," In: 2016 18th Asia-Pacific network operations and management symposium (APNOMS). IEEE, pp. 1–4, 2016.

[129] B. M. Nguyen, H. Thi Thanh Binh, T. The Anh, and D. Bao Son, "Evolutionary Algorithms to Optimize Task Scheduling Problem for the IoT Based Bag-of-Tasks Application in Cloud–Fog Computing Environment," Applied Sciences, vol. 9, no. 9, p. 1730, Apr. 2019, doi: https://doi.org/10.3390/app9091730.

[130] M. Mukherjee, M. Guo, J. Lloret, R. Iqbal, and Q. Zhang, "Deadline-Aware Fair Scheduling for Offloaded Tasks in Fog Computing With Inter-Fog Dependency," vol. 24, no. 2, pp. 307–311, Feb. 2020, doi: https://doi.org/10.1109/lcomm.2019.2957741.

[131] C. Avasalcai, S. Dustdar, , "Latency-Aware Distributed Resource Provisioning for Deploying IoT Applications at the Edge of the Network," pp. 377–391, Feb. 2019, doi: https://doi.org/10.1007/978-3-030-12388-8_27.

[132] J. Yao, N. Ansari,"QoS-Aware Fog Resource Provisioning and Mobile Device Power Control in IoT Networks," IEEE Transactions on Network and Service Management, vol. 16, no. 1, pp. 167–175, Mar. 2019, doi: https://doi.org/10.1109/tnsm.2018.2888481.

[133] J. Santos, T. Wauters, B.Volckaert, and F. D. Turck, "Resource provisioning in fog computing through deep reinforcement learning," In 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 431-437, IEEE, 2021.

[134] B. Donassolo, I.Fajjari, A. Legrand, and P.Mertikopoulos, "Fog based framework for IoT service provisioning," In 2019 16th IEEE annual consumer communications & networking conference (CCNC), pp. 1-6. IEEE, 2019.

[135] S. Agarwal, S. Yadav, and A. K. Yadav, "An efficient architecture and algorithm for resource provisioning in fog computing," International Journal of Information Engineering and Electronic Business, vol. 8, no. 1, pp. 48, 2016.

[136] S. O. Ogundoyin and I. A. Kamil, "A Fuzzy-AHP based prioritization of trust criteria in fog computing services," Applied Soft Computing, vol. 97, p. 106789, Dec. 2020, doi: https://doi.org/10.1016/j.asoc.2020.106789.

[137] Md. A. Maruf, A. Singh, A. Azim, and N. Auluck, "Faster Fog Computing based Over-the-air Vehicular Updates: A Transfer Learning Approach," IEEE Transactions on Services Computing, pp. 1–1, 2022, doi: https://doi.org/10.1109/tsc.2021.3099897.

[138] S. Dehnavi, H. R. Faragardi, M. Kargahi, and T. Fahringer, "A reliability-aware resource provisioning scheme for real-time industrial applications in a Fog-integrated smart factory," Microprocessors and Microsystems, vol. 70, pp. 1–14, Oct. 2019, doi: https://doi.org/10.1016/j.micpro.2019.05.011.

[139] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, "Resource provisioning for IoT services in the fog computing environment: An autonomic approach," Computer Communications, vol. 161, pp. 109–131, Sep. 2020, doi: https://doi.org/10.1016/j.comcom.2020.07.028.

[140] M. Aazam, E.N. Huh, "Dynamic resource provisioning through fog micro datacenter," IEEE Int. Conf. Pervasive Comput. Commun. Workshop PerCom Workshop 2015, pp. 105–110, 2015. https://doi.org/10.1109/PERCOMW.2015.7134002

[141] S. Zhao, L. Yu, and B. Cheng, "An event-driven service provisioning mechanism for IoT (Internet of Things) system interaction," IEEE Access 4, pp. 5038–5051, 2016. https://doi.org/10.1109/ACCESS.2016.2606407

[142] H.M. Fard, R. Prodan, F. Wolf, A container-driven approach for resource provisioning in edge-fog cloud, in Algorithmic Aspects of Cloud Computing. ed. by I. Brandic, T.A.L. Genez, I. Pietri, R. Sakellariou (Springer International Publishing, Cham, 2020), pp. 59–76.

[143] A.V. Chandak, N.K. Ray, Multi agent based resource provisioning in fog computing, in Trends in Computational Intelligence, Security and Internet of Things. ed. by N. Kar, A. Saha, S. Deb (Springer International Publishing, Cham, 2020), pp. 317–327

[144] H.N. Pham-Nguyen, Q. Tran-Minh, Dynamic resource provisioning on fog landscapes. Secur. Commun. Netw. 2019https://doi.org/10.1155/2019/1798391

[145] Yao, Hong, Changmin Bai, Muzhou Xiong, Deze Zeng, and Zhangjie Fu. "Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing." Concurrency and Computation: Practice and Experience 29, no. 16 (2017): e3975.

[146] Liu, Chubo, Kenli Li, and Keqin Li. "Minimal cost server configuration for meeting time-varying resource demands in cloud centers." IEEE Transactions on Parallel and Distributed Systems 29, no. 11 (2018): 2503-2513.

[147] Abouaomar A, Cherkaoui S, Mlika Z, Kobbane A (2021) Resource provisioning in edge computing for latency sensitive applications. IEEE Internet Things J. https:// doi. org/ 10. 1109/ JIOT. 2021. 30520 82.

[148] Santos J, Wauters T, Volckaert B, De Turck F (2021) Towards end-to-end resource provisioning infog computing over low power wide area networks. J NetwComput Appl 175:102915. https:// doi.org/ 10. 1016/j. jnca. 2020. 102915

[149] Son J, Buyya R (2019) Latency-aware virtualized network function provisioning for distributededge clouds. J SystSoftw 152:24–31. https:// doi. org/ 10. 1016/j. jss. 2019. 02. 030

[150] Emre Yigitoglu et al. "Foggy: A framework for continuous automated iotapplicationdeployment in fog computing". In: 2017 IEEE International Conference on AI &MobileServices (AIMS). IEEE. 2017, pp. 38–45.

[151] Per-Olov Östberg et al. "Reliable capacity provisioning for distributed cloud/edge/fog computingapplications". In: 2017 European conference on networks and communications(EuCNC). IEEE. 2017, pp. 1–6.

[152] Mohammad FM, Jabbehdari S, Javadi HHS (2020) A dynamic fog service provisioningapproachfor IoT applications. Int J CommunSyst 33(14):e4541.

[153] Shahidinejad A, Ghobaei-Arani M (2020) Joint computation offloading and resource provisioningfor edge-cloud computing environment: a machine learning-based approach. Softw: PractExper50: 2212– 2230. https:// doi. org/ 10. 1002/ spe. 2888.

[154] Wang N, Matthaiou M, Nikolopoulos DS, Varghese B (2020) DYVERSE: dynamic vertical scalingin multi-tenant edge environments. Future Gener ComputSyst 108:598–612. https:// doi. org/ 10.1016/j. future. 2020. 02. 043

[155] Mulinti RB, Nagendra M (2021) An efficient latency aware resource provisioning in cloudassisted mobile edge framework. Peer-to-Peer Netw Appl 14:1044–1057. https:// doi. org/ 10.1007/s12083- 020- 01070-6.

[156] Roy B, Multi-Criteria Methodology for Decision Aiding. Kluwer Academic Publishers, 1996.

[157] Evangelos Triantaphyllou, Multi-Criteria Decision Making Methods: A Comparative Study, Springer-Science + Business Media B.V, 2000.

[158] R. Venkata Rao, Decision Making in the Manufacturing Environment UsingGraph Theory and Fuzzy Multiple Attribute Decision Making Methods, SpringerSeries in Advanced Manufacturing Volume 2, 2013.

[159] Golam Kabir, RehanSadiq& Solomon Tesfamariam, A review of multi-criteria decision-making methods for infrastructure management, Structure and Infrastructure Engineering: Maintenance, Management, Life-Cycle Design and Performance, 10:9, 1176-1210, 2014,DOI: 10.1080/15732479.2013.795978

[160] Weistroffer, H. R., Smith, C. H., and Narula, S. C., "Multiple criteria decision support software", Ch 24 in: Figueira, J., Greco, S., and Ehrgott, M., eds,Multiple Criteria Decision Analysis: State of the Art Surveys Series, Springer: New York, 2005.

[161] Senouci MA, Mushtaq MS, Hoceini S, Mellouk A (2016) TOPSISbased dynamic approach for mobile network interface selection. ComputNetw 107:304–314.

[162] Russell, Roberta S. and Taylor III, Bernard W. Operations Management 4th edition. Upper Saddle River, New Jersey: Prentice Hall, (2003).

[163] Vahidnia, M. H.; Alesheika, A. A.; Alimohammadi, A.,"Hospital site selection using AHP and its derivatives", Journal of Environmental Management, 90, pp. 3048-3056. (2009).

[164] Saaty, T. L., "A scaling method for priorities in hierarchical structures." Journal of Mathematical Psychology 15(3): 234-281, 1977.

[165] Saaty, T.L, "Decision Making for Leaders: The Analytic Hierarchy Process for Decisions in complex World, RWS" Publications, Pittsburgh, PA, 1986.

[166] Nooramin Amir Saeed; Kiani Moghadam Mansoor; Moazen Jahromi Ali Reza; Sayareh Jafar, "Comparison of AHP and FAHP for Selecting Yard Gantry Cranes in Marine Container Terminals", Journal of the Persian Gulf (Marine Science)/Vol. 3/No. 7/March 2012 /12/59-70.

[167] Saaty, T.L, "Multi-criteria Decision Making: The Analytic Hierarchy Process, 1988; Revised and published by the author; Original version published by McGraw-Hill, New York, 1980.

[168] Saaty, Thomas L. "Fundamentals of the analytic network process—Dependence and feedback in decision-making with a single network." Journal of Systems science and Systems engineering 13, no. 2 (2004): 129-157.

[169] Opricovic, S. "Multi-criteria Optimization of Civil Engineering Systems, Faculty of Civil Engineering, Belgrade." Table II The performance matrix (1998).

[170] García-Melón, Mónica, Javier Ferrís-Oñate, Jerónimo Aznar-Bellver, Pablo Aragonés-Beltrán, and Rocío Poveda-Bautista. "Farmland appraisal based on the analytic network process." Journal of Global Optimization 42, no. 2 (2008): 143-155.

[171] Kheybari, Siamak, Fariba Mahdi Rezaie, and Hadis Farazmand. "Analytic network process: An overview of applications." Applied Mathematics and Computation 367 (2020): 124780.

[172] Martin Aruldoss, T. Miranda Lakshmi, V. Prasanna Venkatesan, "A Survey on Multi Criteria Decision Making Methods and Its Applications", American Journal of Information Systems, Vol. 1, No. 1, 31-43, (2013).

[173] Hwang, C.L.; Yoon, K. "Multiple Attribute Decision Making: Methods and Applications", New York: Springer-Verlag, (1981).

[174] Chen, C. T. "Extensions of the TOPSİS for Group Decision Making Under Fuzzy Environment." Fuzzy Sets and Systems, 114: 1-9, (2000).

[175] Wang, J. W., Cheng, C. H. & Huang, K. C. "Fuzzy Hierarchical TOPSÝS for Supplier Selection." Applied Soft Computing, 9: 377-386, (2009).

[176] Brans, Jean-Pierre, and Ph Vincke. "Note—A Preference Ranking Organisation Method: (The PROMETHEE Method for Multiple Criteria Decision-Making)." Management science 31, no. 6 (1985): 647-656.

[177] Giurca, Ioan, Ioan ASCHILEAN, Calin Ovidiu SAFIRESCU, and Dan MURESAN. "Choosing Photovoltaic Panels using the PROMETHEE Method." In Proceedings of the INTERNATIONAL MANAGEMENT CONFERENCE, vol. 8, no. 1, pp. 1087-1098. Faculty of Management, Academy of Economic Studies, Bucharest, Romania, 2014.

[178] J-P. Brans, and B. Mareschal, "Chapter 5 promethee methods", Multiple Criteria Decision Analysis: State of the Art Surveys, pp.163-186, 2005.

[179] J.P. Brans, P. Vincke, and B. Mareschal, "How to select and how to rank projects: The PROMETHEE method", Eur. J. Oper. Res., vol. 24, pp. 228-238, 1986.

[180] A. Kumar, "A review of multi criteria decision making (MCDM) towards sustainable renewable energy development", Renew. Sustain. Energy Rev., vol. 69, pp. 596-609, 2017.

[181] L. Abdullah, W. Chan, and A. Afshari, "Application of PROMETHEE method for green supplier selection: A comparative result based on preference functions", J. Indust. Eng. Int., vol. 15, no. 2, pp. 271-285, 2019.

[182] J.P.Brans& P.Vincke,"A preference ranking organization method: The PROMETHEE method for MCDM". Management Science,31(6),647-656,1985.

[183] M. Behzadian, R. Kazemzadeh, A. Albadvi, and M. Aghdasi, "PROMETHEE: A comprehensive literature review on methodologies and applications", European Journal of Operational Research, vol. 200, no. 1, pp. 198-215,2010.

[184] S. Opricovic and G.-H. Tzeng, "Extended VIKOR method in comparison with outranking methods," European Journal of Operational Research, vol. 178, no. 2, pp. 514–529, Apr. 2007, doi: https://doi.org/10.1016/j.ejor.2006.01.020.

[185] 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, Jul. 2004, doi: https://doi.org/10.1016/s0377-2217(03)00020-1.

[186] C.-L. Chang, "A modified VIKOR method for multiple criteria analysis," Environmental Monitoring and Assessment, vol. 168, no. 1–4, pp. 339–344, Aug. 2009, doi: https://doi.org/10.1007/s10661-009-1117-0.

[187] R. Sandhu and S. K. Sood, "Scheduling of big data applications on distributed cloud based on QoS parameters," Cluster Computing, vol. 18, no. 2, pp. 817–828, Dec. 2014, doi: https://doi.org/10.1007/s10586-014-0416-6.

[188] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host CPU utilization in cloud computing using recurrent neural networks," 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), Dec. 2017, doi: https://doi.org/10.23919/icitst.2017.8356348.

[189]E.Triantaphyllou and E. Triantaphyllou,"A sensitivity analysis approach for MCDM Methods,"Multi-criteria decision making methods: A comparative study, pp. 131-175, 2000.

[190]E.Erkut and M. Tarimcilar,"On Sensitivity Analysis in the Analytic Hierarchy Process,"IMA Journal of Management Mathematics, vol. 3, no. 1, pp. 61–83, 1991.

[191]G. Kabir and M. A. A. Hasin, "Integrating fuzzy AHP with TOPSIS method for optimal power substation location selection," International Journal of Logistics Economics and Globalisation, vol. 5, no. 4, p. 312, 2013, doi: https://doi.org/10.1504/ijleg.2013.059166

[192]G. Kabir and R. S. Sumi, "Power substation location selection using fuzzy analytic hierarchy process and PROMETHEE: A case study from Bangladesh," Energy, vol. 72, pp. 717–730, Aug. 2014, doi: https://doi.org/10.1016/j.energy.2014.05.098.

LIST OF PUBLICATIONS

Journals

1. S.Varshney, R.Sandhu, and P.K. Gupta, "Cost-Effective Scheduling in Fog computing: An Environment based on Modified PROMETHEE Technique", Journal of Universal Computer Science (JUCS), vol. 29, no. 4, pp. 397-416, 2023. [SCIE, Impact Factor: 1.056]

2. S.Varshney, R.Sandhu, and P.K. Gupta, "QoE-Based Multi-Criteria Decision Making for Resource Provisioning in Fog Computing Using AHP Technique," International Journal of Knowledge and Systems Science (IJKSS), vol. 11, no. 4, pp. 17-30, 2020. [SCOPUS]

3. S.Varshney, R.Sandhu, and P.K. Gupta, " An Effective Multi Criteria Decision Making Approach For Allocation of Resources in the Fog Computing Environment," World Scientific Journal. [SCIE, Impact Factor: 3.508]

Conferences

1. S. Varshney, R. Sandhu, and P.K. Gupta, "QoE-based Resource Management of Applications in the Fog Computing Environment using AHP Technique," In 2021 6th International Conference on Signal Processing, Computing and Control (ISPCC), pp. 669-673. IEEE, 2021.

2. S. Varshney, R. Sandhu, and P.K. Gupta, "Developing MCDM-based technique to calculate trustworthiness of advertised QoE parameters in Fog Computing Environment," In 3rd International Conference on Machine Learning, Image Processing, Network Security and Data Sciences (MIND), pp. 705-714, Springer, 2023.

Book Chapter

1. S. Varshney, R. Sandhu, and P. Gupta, "Multicriteria decision-making in health informatics using IoT," in IoT-Based Data Analytics for the Healthcare Industry}, Elsevier, pp. 105—121.