IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 February 2023, accepted 5 March 2023, date of publication 28 March 2023, date of current version 5 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3262611

RESEARCH ARTICLE

A Distributed Fuzzy Optimal Decision Making Strategy for Task Offloading in Edge Computing Environment

SASMITA RANI BEHERA[1], NIRANJAN PANIGRAHI[2], SOURAV KUMAR BHOI[2], MUHAMMAD BILAL[3], (Senior Member, IEEE), KSHIRA SAGAR SAHOO[4], (Senior Member, IEEE), AND DAEHAN KWAK[5], (Member, IEEE)

[1]Department of Computer Science and Engineering, Faculty of Engineering, Biju Patnaik University of Technology (BPUT), Rourkela, Odisha 769015, India
[2]Department of Computer Science and Engineering, Parala Maharaja Engineering College (Government), Berhampur, Odisha 761003, India
[3]Department of Information and Communication Engineering, Hankuk University of Foreign Studies, Yongin-si 17035, South Korea
[4]Department of Computer Science, SRM University, Amaravati, Andhra Pradesh 522240, India
[5]Department of Computer Science and Technology, Kean University, Union, NJ 07083, USA

Corresponding author: Daehan Kwak (dkwak@kean.edu)

This work was supported in part by the Office of Research and Sponsored Programs, Kean University.

ABSTRACT With the technological evolution of mobile devices, 5G and 6G communication and users' demand for new generation applications viz. face recognition, image processing, augmented reality, etc., has accelerated the new computing paradigm of Mobile Edge Computing (MEC). It operates in close proximity to users by facilitating the execution of computational-intensive tasks from devices through offloading. However, the offloading decision at the device level faces many challenges due to uncertainty in various profiling parameters in modern communication technologies. Further, with the increase in the number of profiling parameters, the fuzzy-based approaches suffer inference searching overheads. In this context, a fuzzy-based approach with an optimal inference strategy is proposed to make a suitable offloading decision. The proposed approach utilizes the Classification and Regression Tree (CART) mechanism at the inference engine with reduced time complexity of $O(\mid V \mid^2 log_2 \mid L \mid))$, as compared to $O(\mid L \mid^{\mid V \mid})$ of state-of-the-art, conventional fuzzy-based offloading approaches, and has been proved to be more efficient. The performance of the proposed approach is evaluated and compared with contemporary offloading algorithms in a python-based fog and edge simulator, YAFS. The simulation results show a reduction in average task processing time, average task completion time, energy consumption, improved server utilization, and tolerance to latency and delay sensitivity for the offloaded tasks in terms of reduced task failure rates.

INDEX TERMS Computation offloading, decision-making, fuzzy logic, MEC.

I. INTRODUCTION

With the growing usage of mobile user equipment (UE), the evolution of modern communication technologies viz. 5G and 6G, there is a proportionate increase in demand for mobile applications which are computationally intensive in nature. In the last decade, such applications have shown extensive usage in mobile healthcare services [1], mobile learning services [2] etc. But, due to the constrained resource capacity of UE, there is a need for helpers or servers to whom the application's inherent tasks can be offloaded to do the task execution in favor of mobile devices and send back the computational results. For example, a study in [3] shows that many mobile apps of different categories like gaming, puzzles, and image manipulation are energy hungry. Usage of these applications has shown a substantial amount of battery drain on mobile devices. For example, the authors considered around 550 apps for face manipulation, more than

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Pau.

7800 apps for gaming, and 717 apps for puzzles, and specific to a chess game, they have considered around 166 numbers. The average energy drain of every app group is compared with the subset. It is observed that 44%, 45%, 42%, and 33% of respective apps which execute high computational operations cause more energy drain than the normal apps. This shows that a significant number of apps that involves high computation require the task to be offloaded and executed remotely to save UE's energy consumption. This leads to the paradigm of mobile cloud computing services. But, this paradigm has faced many challenges such as service delays due to remoteness, the need for higher bandwidth, and degraded Quality of Experience (QoE) for end users.

To overcome the specific challenge of a service delay due to the remoteness of cloud servers, mobile edge computing (MEC) plays an alternative role in the smooth computation of complex applications near to users. It also serves as the backbone paradigm of 5G communication [4]. Still, MEC suffers the common challenges of offloading. Those generic challenges can be classified into three major domains: when to offload, where to offload, and what to offload [5], [6]. This work focuses on the context of when to offload the problem domain where it is required to decide whether the offloading will be effective or not by considering various parameters like the size of the task, communication time, computation time, energy, and delay. The offloading decision should satisfy the strict constraint that the total time taken in offloading the tasks and getting the results from offloaded tasks should be less than the time involved in the local computation. In many cases, due to the dynamism of various device and network-level parameters, there lies uncertainty in decision-making, thus making the decision process a challenging task [7].

Recently, fuzzy-based approaches in offloading decision-making [8], [9], [10], [40] have attracted the research community. But, the majority of fuzzy-based strategies focus on fuzzification, defuzzification, and conventional inference engine design strategies with few offloading parameters. In a realistic environment, the number of offloading parameters may increase, depending on the type of application, device, and environment parameters [31]. As the number of offloading parameters increases, the size of rule sets in fuzzy-based off-loader grows exponentially which leads to a complex fuzzy inference system (CFIS) [32]. In CFIS-based off-loader, the decision-making time will be a bottleneck which will reduce the overall performance of the system and the inherent objective of offloading will not be met. To the best of our knowledge, this work has made a first attempt to design an optimal inference strategy over a conventional fuzzy inference engine for accelerated performance in the decision-making of fuzzy-based off-loader in the MEC environment. To this end, the major contributions of this article are as follows:

1) A fuzzy-based approach is proposed to handle the uncertainty in profiling parameters for task offloading in MEC.
2) The inference engine of the fuzzy-based offloading system is optimized, using a Classification and Regression Tree-based approach on fuzzy offloading rules-set, for optimal decision-making.
3) A thorough mathematical analysis is performed to prove the optimal complexity of the proposed inference mechanism.
4) Extensive simulations are carried out to compare the performance of the proposed optimal offloading method with some state-of-the-art offloading strategies by considering different performance metrics on a python-based simulator, YAFS.

The rest of the article is organized as follows. A background study of different approaches to decision-making in computation offloading is presented in section II. Section III presents a theoretical analysis of conventional fuzzy-based decision-making which motivates the design of a fuzzy optimal decision-making strategy. The proposed network architecture and problem formation are discussed in section IV, followed by system models in section V. Section VI briefly discussed the proposed CART-based fuzzy optimal approach with detailed mathematical analysis in section VII. Section VIII presents the simulation environment using YAFS and the result discussion. Finally, section IX concludes with future work.

II. RELATED WORK

This section presents a comprehensive survey in the context of different approaches followed in offloading decision-making in MEC. They are categorized into machine learning, graph theory, auction theory, and fuzzy-based approaches. The overall survey is summarized in Table 1.

The machine learning approach helps to optimize the network management processes in the 5G environment [11]. The authors have addressed the issue of placement of the Service Function Chain (SFC) under 5G network architecture. The underlying problem can be generalized as a task offloading problem. The SFC is a chain of ordered function requests (analogous to a set of dependent tasks) from User Equipment (UE) where each function request is associated with end-to-end latency and data rate requirements. They adopted a machine learning-based approach for the prediction of the number of VNF requirements for SFC using traffic demand. The problem is formulated as an ILP for jointly optimizing the placement of SFC in VNF and at the same time handling end-to-end delay and data rate using a heuristic. The authors have considered the binary decision strategy for offloading the task. In [12], they have considered better service accuracy by applying the machine learning case study. They have selected the appropriate path to find a suitable edge server in an industry-based IoT environment. They have considered decisions based on the accuracy of the edge server and not considered the constraints on resources. Machine learning techniques are helpful to improve the performance of MEC [13]. Machine learning is applied in

device coordination and task scheduling algorithms. They proposed a task-scheduling algorithm in a distributed environment using a game-theoretic approach. In MEC servers, the scheduling policy decisions are taken by task scheduling strategies. Their procedures perform better coordination of the devices, that want to cooperate for participating in the distributed computing environment. They have considered scheduling policy decisions in the distributed task scheduling scenario in a single MEC server, which may not be suitable for multiple servers.

The game-theoretic approach helps in better scheduling of the tasks and making better decisions in selecting the server in a multi-user, multi-server, and multi-channel environment [14]. They have considered the Nash equilibrium to get the optimized result with a reduction in time, energy, and cost. The decision is based on the covariance of the analytic hierarchical process. They have not considered the selection of the appropriate server in the ad-hoc network. In [15], they have considered application partitioning in a dynamic environment. The authors proposed the MCOP algorithm to make the optimal partitioning, which results in a reduction in time, cost, energy, and delay. They have shown the optimal partitioning in a weighted consumption graph where the mobile-side and server-side executions are clearly distinguished. The offloading decision is based on the computational and communicational cost in the WCG in a task partitioning scenario, which minimizes the response time. They have not considered task partitioning in a parallel manner. In [16], Li. et al. considered the overall problem that arises in the computation and communication model during the offloading process, they have proposed a cluster-oriented graph-based partitioning algorithm to reduce the overhead of the task scheduling process in a dynamic environment. They have shown how the reduction in energy consumption takes place after considering the clustering of tasks when the task size increases. The task scheduling decision helps to minimize the overhead of the unloading and task scheduling process, but they have not considered the priority of the task during execution.

Auction theory played an important role in task scheduling and choosing the MEC server. In [17], the multi-objective decision-making strategies are used based on the analytic hierarchy process model for selecting the suitable server for offloading. And also the improved auction theory-based algorithm is used in the task scheduling procedure of computation offloading to reduce service delay and energy consumption with better QoE which improves the task execution speed. The improved auction theory requires the tasks and VMs in sorted order, but not perform well when considering the task failure rate. Also, auction theory plays an essential role in finding a suitable nearby MEC server [18]. The procedure helps in finding better dynamic pricing in MEC. Their proposed double auction-based method helps in finding suitable servers only within the range. They have only considered similar types of tasks for the offloading in edge servers.

In the dynamic environment of the traffic load variations with respect to time and vehicle terminal mobility, making the decision for offloading becomes a crucial task. To make an effective offloading decision and resource allocation, the authors proposed a Lyapunov-based optimization algorithm [19], which helps to maintain the stability of the queue, and the network utility function is minimized by converting the tasks into subtasks. In [20], the authors observed similar repeated task offloading request patterns. So, when the decision for the offloading of the tasks arrives with repeated patterns, then the probabilistic methods will be helpful for making the better decision of offloading. They have computed the computation, communication, and edge computing probabilities based on varying the task arrival time, latency, and the size of the task. They have not considered optimization techniques for decision-making. In [21], the authors proposed the particle swarm optimization for taking the decision of offloading the task into the MEC server. Here they have prepared a mathematical model for the computation of the cost during the offloading. They have taken the channel gain matrix to consider the transmission rate of the task during offloading to the MEC. Their result shows the effective reduction of energy during the offloading process. They have not considered the resource requirement and priority of the task while offloading. In [22], they have proposed the offloading algorithm based on deep supervised learning in a dynamic weighted task environment. They have increased their system utility and near-optimal offloading decisions are achieved by considering only four training samples of each MEC server. They have not considered the MEC scenarios in a dynamically changing environment.

Fuzzy-based approaches are helpful in making better decisions during the offloading process. In [9], the authors have considered a fuzzy strategy in a collaborative environment among edge servers by considering the delay-sensitivity of tasks and suggested a server selection method to improve QoS. In [10], the input parameters considered for fuzzification are WAN bandwidth, VM utilization, length of the incoming task, and delay-sensitivity of the task. Additionally, the authors have also proposed a fuzzy-based orchestration strategy among edge servers with the objective to balance the workload among local edge servers, remote edge servers, and cloud servers. The authors have not considered any optimization techniques on rule-set and also have not considered the task migration scenario between servers. In [23], they have used the fuzzy method for decision-making and the optimal algorithm for node selection. Their algorithm reduces the cost and improves the response time. The authors have not considered the workload balance in the user mobility scenario. In [24], a suitable 3-tier architecture has been proposed by considering a heterogeneous and dynamic environment. Keeping in view of the applications with latency-sensitive nature, the proposed algorithm reduced the service time with better resource utilization, resulting in a reduction in task failure rate. They have not considered

the task dependencies while scheduling the tasks during offloading. A multi-tier architecture is also proposed in [25] based on the fuzzy decision-making mobile edge orchestrator. Their experiment provides a better result in improving the processing time and reduction in task failure rate in a real-time environment. They have not considered optimization techniques for task scheduling. In [26], a fuzzy-based scheme is proposed to select edge servers where the decision parameters considered are latency-sensitivity, the capacity of servers, and network parameters. FTOM algorithm is proposed by considering the fuzzy decision for choosing a better computing device during offloading. They have shown improvements in the reduction of the completion time and success rate of the execution of tasks. They have considered an orchestrator management layer for better management in the distribution of tasks among the servers. The authors have not considered any optimization techniques for offloading the task.

In summary, the above-surveyed works reflect that through a series of fuzzy-based and similar approaches are available in the existing literature for task offloading decision-making, the state-of-the-art approaches suffer inference-searching overheads to make the offloading decisions, with the increase in the number of offloading parameters. The same is thoroughly analyzed in the succeeding section. Further, in the current generation of 5G/6G networks, delayed offloading decisions may lead to the degradation of the Quality of Service (QoS) of the edge computing environment. Hence, it is desired for optimal offloading decision-making in the MEC environment.

III. MOTIVATION: A PRIORI ANALYSIS

The above comprehensive review carried out on offloading strategy in MEC reveals that fuzzy-based approach [10], [26], [29] has recently gained much attention in task offloading decision problem in MEC due to their effectiveness of modeling uncertainty in offloading parameters in 5G communication. The majority of strategies focus on fuzzification, defuzzification, and inference engine design with a limited set of offloading parameters. But in a realistic environment, with the popularity of the 5G technology, the number of offloading parameters may vary from 15-20, based on application, device, and environment parameters [31]. As the number of offloading parameters increases, the size of rule-sets grows exponentially which leads to a complex fuzzy inference system (CFIS) [32]. For example, if the number of offloading parameters is 15, the rule-sets with 3 linguistic variables will consist of $3^{15} = 1, 43, 48, 907$ rules. In such CFIS, the inference engine searching mechanism which performs the matching of fuzzified values with the antecedent part of rule-sets, has a significant impact on the performance of FIS. In 5G environment, with the availability of a faster computation facility, the decision of choosing the offloading for computation should be faster to cope with the 5G technology. This can be further analyzed in detail as follows:

For a FIS-based offloading framework, the overall time complexity (T_o) can be expressed as:

$$T_o = T_{fuz} + T_{inference} + T_{defuz} \tag{1}$$

where T_{fuz} = time complexity of the fuzzification process, $T_{inference}$ = time complexity of the inference engine, and T_{defuz} = time complexity of the defuzzification process. Since T_{fuz} and T_{defuz} processes are dependent on offloading parameters $\mid V \mid$ and their linguistic variables $\mid L \mid$, and independent of the number of rule-sets, so, they can be expressed as:

$$T_{fuz} = O(K_1 \mid L \mid\mid V \mid) \tag{2}$$

$$T_{defuz} = O(K_2 \mid L \mid\mid V \mid) \tag{3}$$

where K_1, K_2 = constants time factor for computations involved in fuzzification and defuzzification process.

But, inference time, $T_{inference}$, has major rule-set searching overhead because with the increase in offloading parameters, the rule-sets grow exponentially, i.e., $\mid L \mid^{\mid V \mid}$. So, $T_{inference}$ can be expressed as:

$$T_{inference} = O(\mid L \mid^{\mid V \mid} + K_3) \tag{4}$$

where K_3 = constant time factor for aggregation, activation accumulation in the inference steps [26]. Using (2), (3) and (4), (1) can be re-written as:

$$\begin{aligned} T_o &= O(K_1 \mid L \mid\mid V \mid) + O(\mid L \mid^{\mid V \mid} + K_3) + O(K_2 \mid L \mid\mid V \mid) \\ &= O(\mid L \mid^{\mid V \mid}) \end{aligned} \tag{5}$$

Equation (4) and (5) indicates that the inference searching mechanism has a significant impact on the overall time complexity T_o of CFIS. Hence, it is required to optimize the inference searching mechanism which will be more beneficial for delay-sensitive task offloading.

Based on this priori analysis and motivation, the following section presents the problem formulation and the underlying network architecture for the proposed fuzzy-optimal offloading strategy.

IV. NETWORK ARCHITECTURE AND PROBLEM FORMULATION

This section presents the underlying network architecture, the offloading framework, and the formalization of the offloading problem through a mathematical description of the proposed fuzzy-based task offloading strategy.

A. NETWORK ARCHITECTURE

The proposed 2-tier network architecture is given in Fig. 1. The different types of user devices (U_i) are present in the lower layer of the architecture. The middle layer represents the different access points (AP_i) available in the nearby places, and the upper layer consists of the nearby available edge servers (ES) to compute the computationally intensive tasks in a 5G environment. The edge layer combines the different access points and available edge servers. The device layer combines the different access points and user devices.

TABLE 1. Summary of related works.

Authors	Approach	Decision Strategy	Performance Metrics
Y. Wang et al. [27]	Local Optimization based on univariate search	Partial	Energy and latency
Z. Luo et al. [4]	Genetic Algorithm	Full	Energy and latency
H. Wu et al. [15]	Graph based on Weighted Consumption Graph	Partial	Time, energy and latency
G. Li et al. [16]	Graph-Based Energy-Clustering Algorithm	Full	Energy and latency
C. Sonmez et al. [10]	Fuzzy-based	Full	Time, latency and task failure rate
Q. Tang et al. [28]	Optimization problem for minimizing the users' overhead	Partial	Time, energy and latency
N. Shan et al. [14]	Game Theoretic and Hierarchy Process based on Covariance (Cov-AHP)	Full	Time, energy and latency
M. D. Hussain et al. [26]	Fuzzy-based	Full	Time, latency and task failure rate
V. Nguyen et al. [29]	Fuzzy-based	Full	Time, latency and task failure rate
Z. Abbas et al. [30]	Deep Learning-based partitioning	Partial	Energy and latency

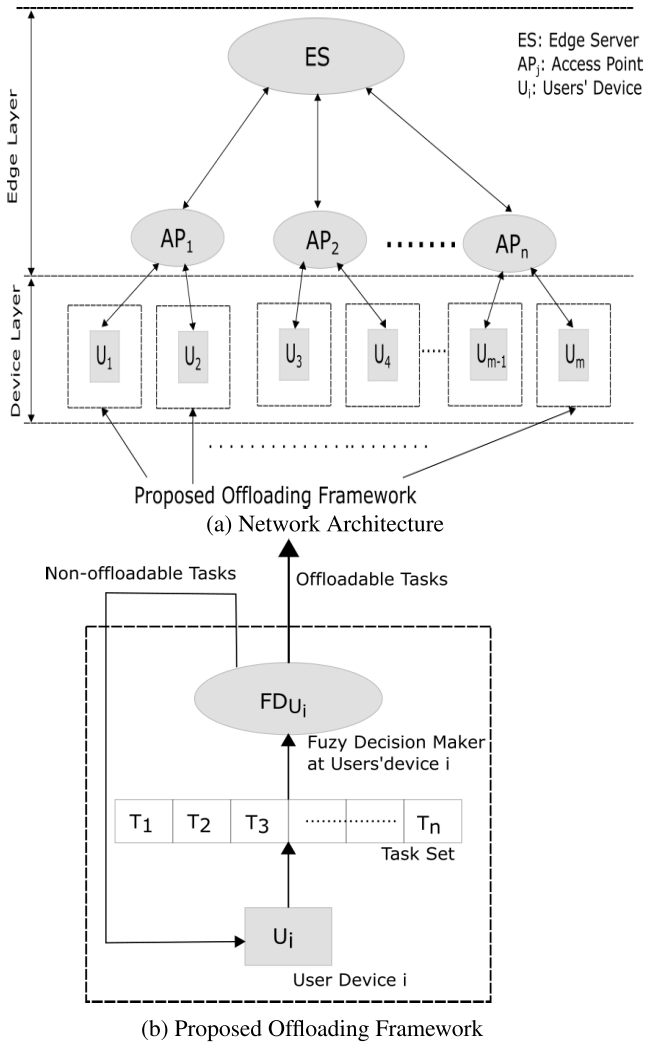


FIGURE 1. (a) MEC Network Architecture, (b) Proposed Fuzzy-based Offloading Framework.

B. OFFLOADING FRAMEWORK

In this section, we will discuss the proposed offloading framework process as shown in Fig. 1(b). The bottom layer represents the user devices with low-power computing processes. When there will be a requirement of offloading the task based on the (6), then by using the fuzzy decision maker, the user will decide to offload the task in the edge server by using the access points and after the execution is completed the user can download the result back from the edge server.

C. PROBLEM DESCRIPTION

Considering the underlying network architecture as shown in Fig. 1(a) and the offloading framework shown in Fig. 1(b), the fuzzy-optimal decision-making for task offloading problem can formally be defined as follows.

Let, at a user U, the number of independent tasks generated at a particular instance is represented by a task set, $T = T_i$, i=1, 2, …, n. Based on various profiling parameters denoted as a set, $P = P_i$, i=1, 2, …, m, it is required to design the fuzzy-optimal decision maker FD_U which will decide whether a task T_i will be executed locally at U or it will be offloaded for execution to MEC server ES. Without loss of generality, the profiling parameters can also be treated as offloading input parameters since they affect the offloading decision-making. The output of the FD_U is denoted as Y^* for a task T_i which is defined as follows:

$$Y^*(T_i) = \begin{cases} \text{Local execution of task} T_i, \; if \; Y^*(T_i) < Th \\ \text{Edge server execution of task} T_i, \; if \; Y^*(T_i) > Th \end{cases} \tag{6}$$

where Th = threshold value for making the offloading decision, based on the crisp output Y^*, produced by the fuzzy-based decision maker FD_U. The FD_U will run independently at each user to support distributed decision-making for task offloading.

The efficacy of FD_U will be evaluated in a simulation environment using performance metrics; offloading decision time versus increase in rule-set size, average task processing time, average task completion time, energy consumption, server utilization, and tolerance to latency and delay sensitivity for the offloaded tasks in terms of task failure rates.

V. SYSTEM MODELS

In this section, we will discuss the different types of system models required to perform the task offloading approaches. In the decision of choosing between local or MEC computation we have to consider many parameters for smooth computation to build different supporting system models are discussed below.

A. TASK MODEL

End users are producing various tasks which are either computed locally or in the MEC server. The decision of where to compute the task based on some attributes, which are used to represent the task with 3 tuples as $T_i = (In_t, Out_t, C_t)$, where In_t is the size of the input data, Out_t is the size of the output data and C_t is the number of CPU cycles required to execute the task during computation. In our proposed model, we assume that the tasks are independent and full offloading of the task is taken place.

B. COMMUNICATION MODEL

In a 5G network, each mobile device will come across a decision of choosing whether to offload or execute tasks locally. When the decision is taken for offloading, then the offloading transmission rate is given by

$$O_T = Blog_2\left(1 + \frac{T_i \times G_i}{N_i}\right) \tag{7}$$

where B is the bandwidth of the channel, T_i is the transmission power of a device, G_i is the channel gain and N_i is the noise received during the transmission. So, we have used Shannon's theorem to calculate the transmission rate, as it produced a maximum transmission rate with minimum delay [14].

C. COMPUTATION MODEL

Here we will discuss the computation in local and MEC servers by considering the size of the data and the clock cycle of the CPU required to compute the task.

1) LOCAL COMPUTING

When the task is executed locally on the user device, the time required to complete the task is shown in (8)

$$T_L = \frac{C_i}{R_L} \tag{8}$$

where C_i is the computation resource of task I and R_L is the rate of the execution of CPU.

2) MEC SERVER COMPUTING

When the task is computed in the MEC server, the total time required to offload the selected task in the MEC server is as follows:

$$T_{offload} = T_{upload} + T_{execute} + T_{download} \tag{9}$$

where T_{upload} = Time required to upload the task in the MEC server, $T_{execute}$ = Time required to execute the task in the MEC server, and $T_{download}$ = Time required to download the task from the MEC server after the successfully executed As the output of the executed task is very smaller than that of the input, we can ignore the download time [33], [34], then the above offloading (9) will become:

$$T_{offload} = T_{upload} + T_{execute} \tag{10}$$

The transmission time of T_{upload} will represented as below:

$$T_{upload} = \frac{D_i}{Blog_2(1 + \frac{P_{upload} \times d^{-\alpha}}{N})} \tag{11}$$

where D_i is the size of the data uploaded to the MEC server, P_{upload} is the uploading power of the user device, $d^{-\alpha}$ is the distance-based channel gain with α value 4 [17], N is the noise power of the channel. The time to execute the task in the MEC server is

$$T_{execute} = \frac{C_i}{R_{MEC}} \tag{12}$$

where R_{MEC} is the execution rate of the MEC server. Now we can compute the final $T_{offload}$ by combining the (11) and (12) as follows:

$$T_{offload} = \frac{D_i}{Blog_2(1 + \frac{P_{upload} \times d^{-\alpha}}{N})} + \frac{C_i}{R_{MEC}} \tag{13}$$

In a 5G environment, the decision of choosing a MEC server for computation offloading will be beneficial when the offloading time will be less than the local computation time can be expressed as below:

$$T_{offload} < T_{local} \tag{14}$$

D. ENERGY MODEL

Here, we have considered the energy in terms of battery consumption by the own user device during the computation that takes place locally. By applying the (8) we can calculate the energy consumption by a local computing device as:

$$E_L = P_L \times T_L = P_L\left(\frac{C_i}{R_L}\right) \tag{15}$$

where P_L is the local processing power of the user device.

VI. PROPOSED FUZZY-OPTIMAL APPROACH

When the user wants to execute any computationally intensive task and needs to review his decision of choosing between the offloading in the MEC server and the local execution in its own device. The decision of offloading the tasks becomes a type of NP-hard problem. For creating a lower complexity environment, we are using a fuzzy-based approach as shown in Fig. 2. The five different crisp inputs are used in the fuzzification process which results in suitable and optimized fuzzy rules from where we get the crisp output for the decision to choose a suitable offloading place by applying the weighted average method. The detailed process is shown in Fig. 2.

A. FUZZIFICATION

In the fuzzification process, we have considered five different crisp input variables which are converted to the resultant fuzzy values. The input variables are assigned with the corresponding linguistic variables with suitable ranges as described in Table 2. The input variables such as task size, network bandwidth, network delay, energy consumption, and MEC VM (Virtual Machine) utilization are used for making

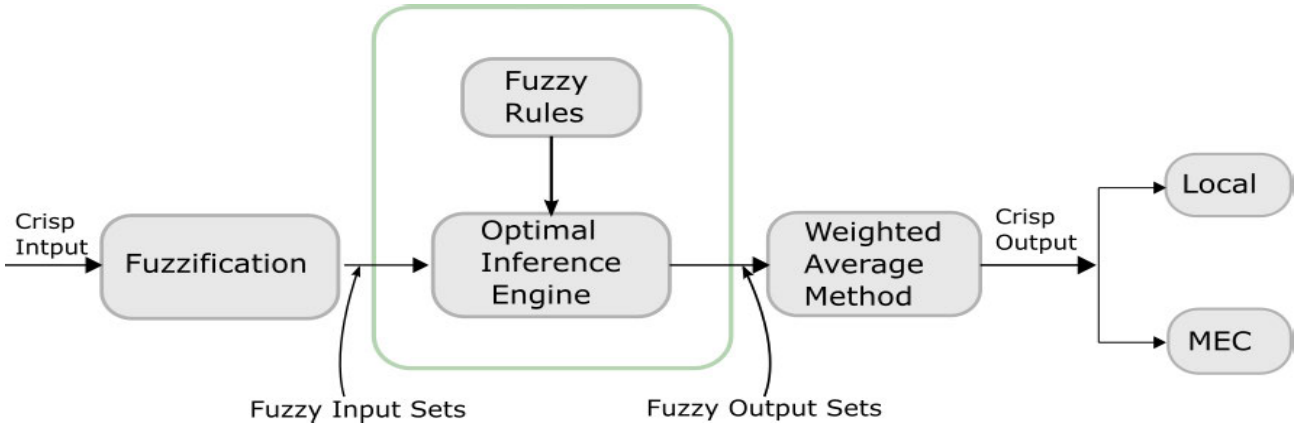


FIGURE 2. The Proposed Fuzzy Logic Architecture.

TABLE 2. Input variables with assigned ranges.

Input variables	Notation	Fuzzy Set	Range
Task Size (GI)	α	Small	0-8
		Medium	6-15
		Large	13-50
Network Bandwidth (Mbps)	β	Low	0-8
		Medium	6-15
		High	13-50
Network Delay (ms)	ρ	Small	0-8
		Medium	6-15
		High	13-50
Energy Consumption (kjoule)	δ	Low	0-30
		Normal	20-70
		High	60-100
MEC VM Utilization (%)	γ	Light	0-30
		Medium	20-70
		Heavy	60-100

efficient decisions for task offloading. The representation of the input variables, $V = \{\alpha, \beta, \rho, \delta, \gamma\}$ where, α represents the size of the task for calculation of the execution time; β denotes the bandwidth requirements of the network for successful offloading; ρ represents the delay in the network while offloading process, δ represents the need of the energy required during the execution of the task; γ denotes the utilization (%) of the VM in MEC.

By considering the fuzzy logic system, three different linguistic variables are assigned with the five different input variables. The linguistic variables help to take the proper decision as when the task size will be large then the role of MEC evolved, and similarly when the bandwidth utilization of MEC will be heavy then choosing the MEC for computation will become a bad option which is trailing towards the increasing rate of the task failure percentage. In a similar way, considering the different linguistic variables result in a better decision-making process. Fig. 3 represents the different linguistic value representations in a trapezoidal member function where the x-axis represents the input variables and the y-axis represents the degree of membership.

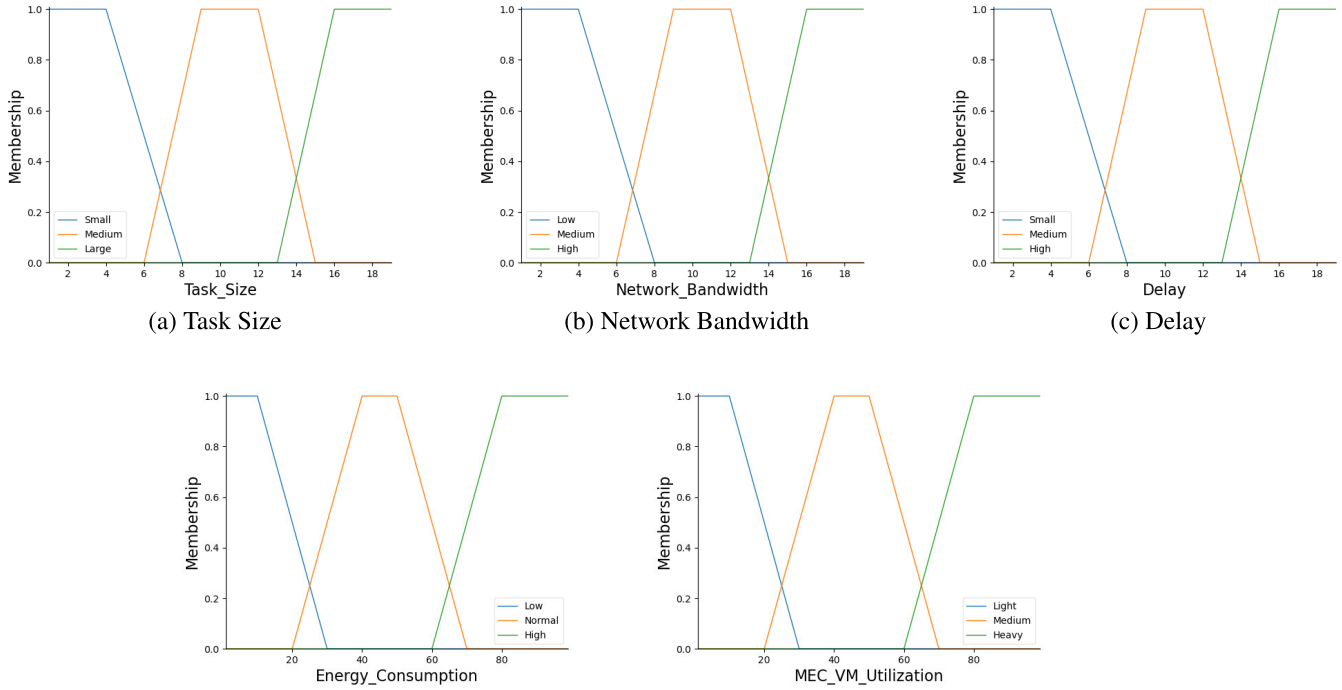
The smooth functioning of the fuzzification process needs different linguistic values such as small (S), medium (M), and large (L) represents task size. For network bandwidth we have considered low (L), medium (M), and high (H) linguistic values; the network delay is associated with the small (S), medium (M), and high (H); the energy consumption is assigned with low (L), normal (N), and high (H); and similarly the MEC VM utilization is linked with light (L), medium (M), and heavy (H) linguistic values.

Membership functions play an important role to produce the fuzzy result. The result is calculated by mapping the input variables with the degree of membership. The output of the result ranges from 0 to 1 and is denoted by $\mu_A(V) \in [0, 1]$. Where $\mu_A(V)$ represents the membership function of A, V represents the element of a fuzzy set and the range of membership values is from 0 to 1. Among the different types of membership functions such as singleton, sigmoid, Gaussian, triangular or trapezoidal, we have used the trapezoidal membership function for the fuzzification process as it produces better results with low complexity in a computational-intensive environment. By considering the trapezoidal membership function in the Fig. 4, the corresponding membership equation 16 is given below:

$$\mu_A(v) = \begin{cases} 0 & v \leq a_1 \\ \dfrac{v - a_1}{a_2 - a_1} & a_1 \leq v \leq a_2 \\ 1 & a_2 \leq v \leq a_3 \\ \dfrac{a_4 - v}{a_4 - a_3} & a_3 \leq v \leq a_4 \\ 0 & v \geq a_4 \end{cases} \tag{16}$$

Here A is a fuzzy set and a_1, a_2, a_3 and a_4 represent the x co-ordinates of the membership function where a_1 and a_4 are the lower and upper limit respectively, and a_2 and a_3 parameters are having the highest membership functions in between them. The y-axis represents the membership value between 0 to 1.

The trapezoidal membership function plays an important role in calculating the fuzzy results. Many papers [35], [36] have proven the effectiveness of the trapezoidal membership



(a) Task Size (b) Network Bandwidth (c) Delay

(d) Energy Consumption (e) MEC VM Utilization

FIGURE 3. Membership function of proposed variables.

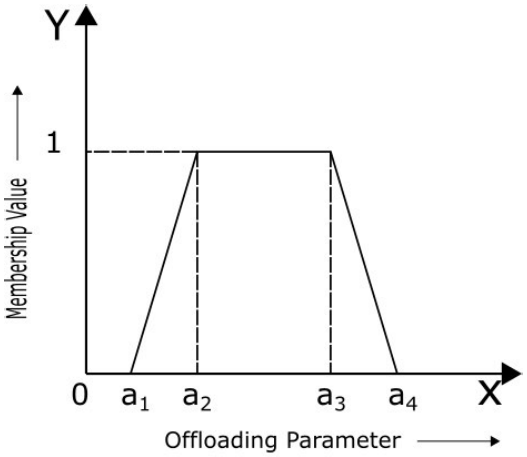


FIGURE 4. The Trapezoidal Membership function.

function. The representations of the trapezoidal membership function for our five input variables are shown in Fig. 3. Considering our designed trapezoidal membership functions, we got the respective fuzzy inputs which are combined with the different inference rules to make the final decision for the system. In the next section, we will discuss the better way to generate an optimal inference engine.

B. OPTIMAL INFERENCE ENGINE

The inference engine considers the fuzzy outputs produced by the fuzzification process and results in some rules, and all these rules are combined together to make a suitable decision of choosing between the local computation and MEC offloading.

We got 243 rules after considering our five different variables with their respective linguistic values. Each rule is linked with its own decision for example, IF α is small AND β is medium AND ρ is medium AND δ is low AND γ is light THEN local execution takes place. We have used the IF-THEN rule with the AND operator for making the inference rules.

During the formation of rules, we observed that some of the rules are coming under one cluster. So, we have started clustering the inference rules by adding the CART algorithm based on Gini index, where we have observed that the resultant decision tree has pruned decision paths. We are getting m-way search in the general decision tree when we have considered all 243 rules individually. Here the tree results in wider branching factors whereas the CART-based optimal binary decision tree has very less paths due to the post-pruning clustering effect. As the resultant decision tree has very less branches, (also called paths) so it produces the decision quickly. The Gini index for the rule-set RS is calculated by using the following equation.

$$Gini(RS) = 1 - \sum_{i=1}^{m} P_i^2 \tag{17}$$

where i is the number of resultant decisions of the rule-set RS i. e. the value of i = 1, 2, …, m and P_i is the relative frequency of the decision taken for RS. In our problem, we are

considering two resultant classes, so the above formula is further derived as below:

$$Gini(RS) = 1 - \left(\frac{P_o}{T}\right)^2 - \left(\frac{N_o}{T}\right)^2 \tag{18}$$

where "P_o" is the number of positive decisions, i.e., MEC execution and "N_o" is the number of negative decisions, i.e., local execution and T is the total number of decisions. For individual variable (V), we have calculated the Gini by using the following formula given below:

$$Gini_v(RS) = \frac{|RS_1|}{|RS|} gini(RS_1) + \frac{|RS_2|}{|RS|} gini(RS_2) \tag{19}$$

Here, RS_1 and RS_2 are the 2 decision clusters of rule-set RS. From here we will be able to know which linguistic variables will be combined together to form a suitable 2-way division by considering the lowest result of the Gini.

Based on the 243 rule-sets from table 3 with our proposed algorithm 1, we have created a CART-based decision tree, based on Gini index by clustering the similar types of decision produced by some of the rules. Fig. 5 shows the overall representation of the decision tree based on Gini index.

Algorithm 1 Optimal Decision Tree Algorithm

Data: Rule-Set (combination of input variables, V= TS, NB, DE, EC, VU with corresponding linguistic variables as (SMALL, MEDIUM, LARGE), (LOW, MEDIUM, HIGH), (SMALL, MEDIUM, HIGH), (LOW, NORMAL, HIGH), (LIGHT, MEDIUM, HEAVY))
Result: Optimized decision tree
Calculate $Gini(RS) = 1 - (P_o/T_o)^2 - (N_o/T_o)^2$; Where RS is the rule-set, P_o is the positive decisions, N_o is the negative decisions and T_o is the total decisions.
for *i=1 to n* **do**
  Compute $(Gini)_v(RS) = 1 - \sum_{i=1}^{n} \frac{RS_i}{RS} gini(RS_i)$;
  Where RS_i is the greedy division of the rule-set RS.
end for
Set $parent(node) \leftarrow min(Gini)$;
Repeat steps 2 to 4 to find the MEC execution and Local execution subgroups of the tree until a leaf node (no further subgroup for decision) is reached;

C. WEIGHTED AVERAGE METHOD

In literature [37], [38], it has been proved that the fuzzy weighted average is used to derive the exact membership function, so we have considered the fuzzy weighted average to generate the crisp output for making a suitable decision. Many research papers [8], [26] used the centroid method to get the final crisp result, but we have considered the weighted average method for calculating the resultant crisp value. This process helps us to avoid the complicated iteration process followed by the other approaches. The weighted average process gives the constant output with less computation and works well in multiple input and single output environments. The weighted average method performs best in an uncertain environment. Algorithm 3 describes the calculation of the crisp result (Y^*) by using the weighted average method where W_i is the minimum weighted value calculated from the fuzzy input parameters and Y_i is the suitable linear equation corresponding to the available rules of the rule-set. The threshold value (Th) is compared with the crisp result and makes a faster decision of the suitable place for computation. The threshold value is derived from the algorithm 2. From the set of generated Y^* values, the average of Y^* generates the possible place of Th, which results in two sets of Y^* values based on the two decisions as explained in (6). By applying the (20), as shown at the bottom of the next page in the algorithm 2, we got the suitable value of Th.

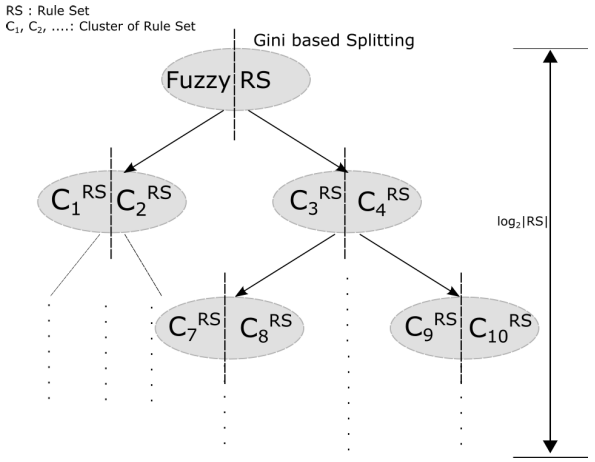


FIGURE 5. The CART-based binary tree for proposed fuzzy optimal offloading.

Algorithm 2 Finding the Threshold Value

Data: $Y^* = \{Y_1^*, Y_2^*, \ldots\ldots, Y_n^*\}$
Result: Threshold '*Th*'
Find_Threshold (Y^*);
Initialize $i \leftarrow 1, j \leftarrow 1, k \leftarrow 1$, Decision1 = [ ] and Decision2 = [ ];
$Th \leftarrow \frac{\sum_{i=1}^{n} Y_i^*}{n}$;
for *i = 1 to n* **do**
  if $y^*[i] < Th$ **then**
    $Decision1[j] \leftarrow Y^*[i]$;
    $j \leftarrow j + 1$;
  end if
  else
    $Decision2[k] \leftarrow Y^*[i]$;
    $k \leftarrow k + 1$;
  end if
end for
while *(|| Decision1 | - |Decision2|| ≥ 0)* **do**
  Compute Th using (20);
end while
return *Th*;

TABLE 3. Inference Rule-Set.

Number of Rules	Task Size	Network Bandwidth	Network Delay	Energy Consumption	MEC VM Utilization	Offload Decision
0	Small	Low	Small	Low	Light	Local
1	Small	Low	Small	Low	Medium	Local
2	Small	Low	Small	Low	Heavy	Local
3	Small	Medium	Medium	High	Light	MEC
4	Small	Medium	Medium	High	Medium	MEC
5	Medium	High	High	High	Light	MEC
6	Medium	High	High	Low	Light	MEC
..	..	..	..	..	..	..
..	..	..	..	..	..	..
241	Large	High	High	High	Medium	MEC
242	Large	High	High	High	Heavy	MEC

Algorithm 3 Fuzzy Based Optimal Decision in MEC (FODM)

Data: Rule-Set
Result: DECISION
Crisp_Value_Generate (Rule_Set);
for $i = 1$ *to* n **do**
  Build Trapezoidal Membership Functions using $\{\alpha, \beta, \rho, \delta, \gamma\}$
  Find min (W_i) and Y_i
  Calculate $Y^* = \sum_{i=1}^{n} \frac{W_i.Y_i}{W_i}$
end for
return Y^*;
if $Y^* < Th$ **then**
  DECISION ← LOCAL COMPUTATION;
end if
else
  DECISION ← MEC COMPUTATION;
end if

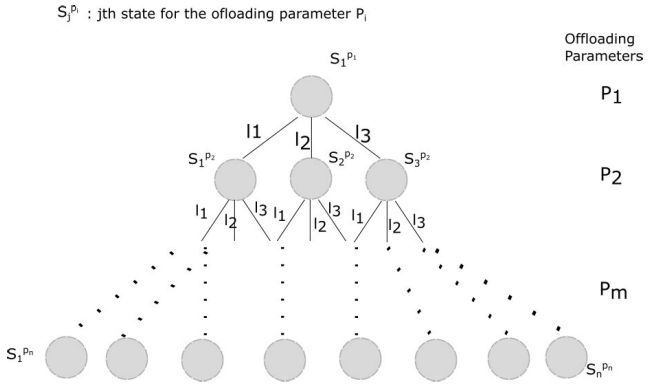


FIGURE 6. State-space search tree.

VII. MATHEMATICAL ANALYSIS

Lemma 1: The conventional fuzzy-based offloading inference engine is inherently a state-space search tree of height $|V|$ and branching factor $|L|$ where $|V|$ = number of offloading parameters and $|L|$ = number of linguistic values for fuzzification of offloading parameters.

Proof: In a conventional fuzzy-based offloading framework, the inference engine searches suitable rules for aggregation and activation by mapping fuzzified values of offloading parameters with the antecedent part of rule-sets. Inherently, the searching process can be logically represented as a state-space search tree as shown in Fig. 6 where the process starts with one of the offloading parameters (p_1) as root-level. The branching factor b of the decision tree will be b= $|L|$, where $|L|$ = number of linguistic variables used for the offloading parameters. The search process explores the next level of nodes for offloading parameter (p_2) and so on. Hence, the height of the tree will be h = $|V|$, where $|V|$ = number of offloading parameters. □

Lemma 2: The search time, T_s, on rule-sets in a conventional fuzzy-based offloading inference engine is upper bounded by $O(|L|^{|V|})$

Proof: As proved in Lemma 1, the conventional fuzzy-based offloading inference engine inherently searches the rules-set in the form of a state-space search tree of height $|V|$ and branching factor $|L|$. With the fuzzified value of the rootlevel offloading parameter (p_1), the next level node states are generated for offloading parameter (p_2) and the number of node states for p_2 is = $|L|^2$ and so on. For, the number of offloading parameters = $|V|$, the state-space search tree will be of depth $|V|$. Hence, the total number of states generated are:

$$S = 1 + |L| + |L|^2 + |L|^3 + \ldots + |L|^{|V|} \tag{21}$$

Since the inference engine searches every rule for aggregation and activation, the searching strategy visits every edge of the state-space search tree and every edge corresponds to one state in the search space. Hence, the total number of states (S) will affect the searching time (T_s) for a conventional

$$Th = \begin{cases} Th & if\,(0 \le ||Decision1| - |Decision2|| \le 1) \\ Th + Step\_Value & if\,((|Decision2| - |Decision1|) > 0) \\ Th - Step\_Value & if\,((|Decision1| - |Decision2|) > 0) \end{cases} \tag{20}$$

fuzzy-based offloading strategy. So, using (21), T_s can be expressed as:

$$\begin{aligned} T_s = O(S) &= O(1 + |L| + |L|^2 + |L|^3 + \ldots + |L|^{|V|}) \\ &= O(|L|^{|V|} - 1)/(|L| - 1) \\ &= O(|L|^{|V|}) \end{aligned} \tag{22}$$

□

Lemma 3: The proposed optimal CART-based offloading inference engine is inherently a binary tree of height, bounded by $\theta(log_2|RS|)$, where $(|RS|) = |L|^|V|$

Proof: For a fuzzified value of an offloading parameter (p_1), the Gini-index is computed, and based on Gini-index, the sample rule-sets are split into two clusters of rules-set as shown in Fig. 5. This process continues iteratively for other offloading parameters. At each level, the clusters of rule-sets are split into two further clusters. So, it inherently induces a binary tree by partitioning the rule-sets $|RS|$ at each level where every node in the tree contains cluster of rule-sets. Since it induces a binary tree, by assuming balance partitioning, the height of the tree will be bounded by $\theta(log_2 |RS|)$. □

Lemma 4: The search time, Ts, on rules-set in a CART-based offloading inference engine is upper bounded by $O(|V|^2\, log_2\, |L|)$

Proof: As proved in Lemma 3, the proposed CART-based offloading inference engine produces a binary tree of height = $\theta(log_2\, |RS|)$. In a binary tree search space, the searching complexity is majorly affected by the height of the tree. Considering, Gini-computation of each rule on offloading parameters, $|V|$, it can be represented as $O(|V|)$. Hence, T_s can be written as:

$$\begin{aligned} T_s &= O(|V|\, \theta(log_2\, |RS|)) \\ &= O(|V|\, log_2\, |L|^{|V|}) \\ &= O(|V|^2\, log_2\, |L|) \end{aligned} \tag{23}$$

So, T_s for the proposed CART-based offloading inference engine is upper bounded by $O(|V|^2\, log_2\, |L|)$. □

VIII. SIMULATION AND OBSERVATIONS

In this section, we discuss briefly the setup of the simulation environment, followed by the performance evaluation using standard performance metrics and models.

A. SIMULATION SET UP

For the implementation of aforesaid algorithms, a state-of-the-art, python-based simulator, YAFS [39] is used under Windows 8, 4 GB RAM, i5 processor, and python 3.8 environment. The simulator mainly consists of five important components; network topology, application, population model, selector model, and placement algorithm. In the network topology component, the proposed network architecture is realized as a hierarchical topology by mapping the simulator's node entity as UEs, APs, and ES, and the link entity in the simulator is mapped to the connection link among UEs to APs and APs to ES. The characteristics of the node entity, IPT (Instructions per simulation time), and RAM (memory available), and the characteristics of link entity, BW (Channel Bandwidth), and Channel Propagation speed (PR) are set as per values given in Table 2 and the simulation parameters given in Table 4. The latency is dynamically computed using task size, BW, and PR.

The proposed fuzzy-based algorithms are embedded in the application component of the simulator with user-defined functions with the defined fuzzy-based rule sets as NumPy arrays as given in Table 3. The population model, selector model, i.e., the service orchestration model, and placement algorithm are set as default in the simulator because they have the least impact on the proposed device-level offloading decision-making. Based on the decision outcome produced from the proposed FODM algorithm, the off-loadable tasks are sent through message (m_a buffer in the simulator) and responses are collected through reply message (m_b buffer). The simulator after the successful execution of tasks produces service time, time_in (the instance of receiving tasks at ES), time_out (the instance of completion of tasks by ES and time_reception (response received by device, i.e., UEs). These are recorded in a CSV file which is produced by the simulator. From these parameters, the performance of the algorithms is evaluated using standard performance metrics and is shown in Fig. 8 to 14.

B. SIMULATION RESULT

In this section, we have presented the performance of the proposed FODM scheme and justified the benefits in the offloading environment. FODM is evaluated using task processing time, completion time, task failure rate, server utilization rate, and energy with respect to the number of tasks produced by the user devices by using the YAFS simulator in the Python environment. For a fair comparison, we have considered a similar set of parameters for decision-making at UE level or device-level, as that of three fuzzy-based approaches: FCTO [9], Sonmez et al. [10], and FTOM [26]. Before evaluating the proposed scheme, the justification of the proposed optimal inference engine is explained first.

The conventional inference engine uses the m-ary decision tree for the decision-making process, which results in overhead to the system as the resulting complexity is in the exponential term already proved in the lemma 1 discussed earlier in this paper. To reduce the complexity, we have used the CART-based binary decision tree for the decision-making process as shown in Fig. 5. Due to the logarithmic time complexity of the binary decision tree, the resultant decision-making process becomes faster than the conventional decision tree model. Fig. 7 represents the behavioral graph among the number of offloading decision-making times versus the increasing number of offloading parameters and the size of the rule-set. The number of offloading parameters is considered from 1 to 15, the size of the rule-set is taken in the logarithmic scale format for simplicity and the offloading decision-making time is

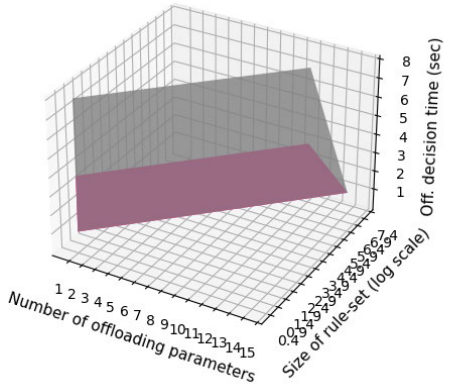


FIGURE 7. Performance evaluation between fuzzy-based and CART-based offloading decision.

represented in seconds. The grey color plot represents the fuzzy-based decision-making process and the pink color plot represents the CART-based decision-making process. In Fig. 7, when the rule-set size is small, then both the conventional decision tree and the CART-based decision tree result in a similar type of decision-making time. But when the number of parameters increased more than 3, then the difference is observed between the fuzzy-based and CART-based decision-making processes. The different paper uses different parameters to find out the effective result such as Sonmez et al. uses WAN, MAN, and WLAN to represent 4 variables, FCTO uses 5 variables to represent local MEC and neighboring MEC and FTOM uses 5 variables to represent WAN with different VM utilization, but when we will combine the approaches to make an efficient model with a large number of variables then the complexity will be increased exponentially, which can be overcome by our optimal inference engine with logarithmic complexity. Fig. 7 justifies the use of CART in the optimal inference engine for making a much faster decision than the fuzzy-based conventional offloading decision.

The efficiency of the proposed FODM scheme is measured by using the different simulation results discussed in Fig. 8 to 14. Table 4 represents the list of simulation parameters used in our result analysis. Fig. 8 represents the performance evaluation based on the average processing time (in y-axis) with respect to the number of tasks (in x-axis) by comparing the different schemes such as FCTO, Sonmez et al., and FTOM with our proposed FODM scheme. Here all four approaches result similarly when the number of tasks is within 200. When the task size increases the Sonmez et al., and FCTO approaches take more processing time due to the traffic congestion in WAN and conventional decision-making process respectively. Our proposed approach and FTOM perform the same till the number of tasks becomes 150. When the number of tasks increases gradually, there is a slight difference in the performance because of the difference in the decision-making process as the FTOM uses the conventional method of decision-making. For example,

TABLE 4. Simulation parameters.

Parameter	Value
Number of tasks	500
Size of the task	2 to 20 GI
Number of VMs per edge server	2 to 8
VM processing speed	10 GIPS
Latency-sensitivity	0 to 10

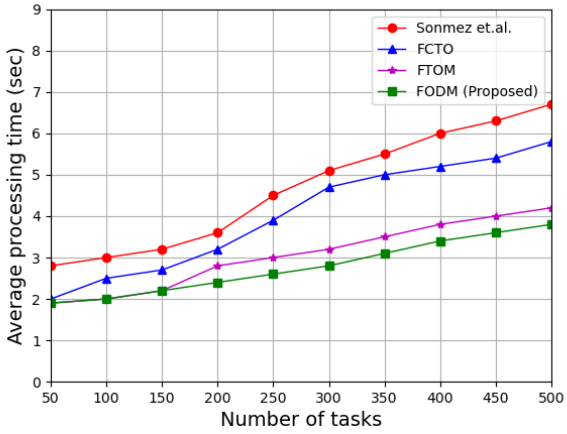


FIGURE 8. Performance evaluation using average task processing times.

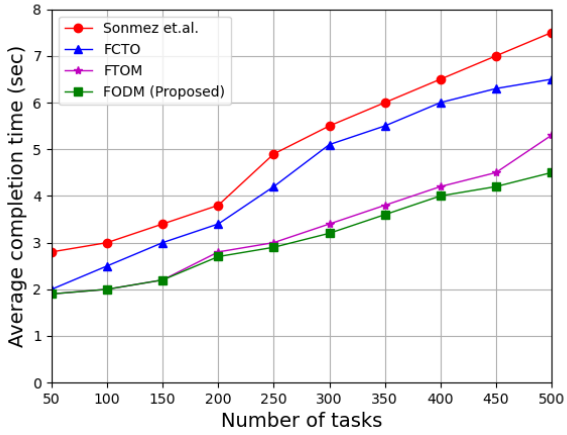


FIGURE 9. Performance evaluation using average task completion time with a number of tasks.

considering a point at 200 tasks, the average processing time of Sonmez et al., FCTO, FTOM, and FODM are 3.6, 3.2, 2.8, and 2.4 respectively, which clearly shows that our proposed approach is performing the best compared to other three approaches.

Fig. 9, represents the task completion time on the y-axis with the number of tasks on the x-axis. The task completion time is the combination of the processing time and network delay. So, by considering the Fig. 8 result, we can conclude that a better processing time results in better performance in the task completion time. The simulation result shows, our proposed approach reduces the task completion time by approximately 37.08%, 27.24%, and 4.46% when compared

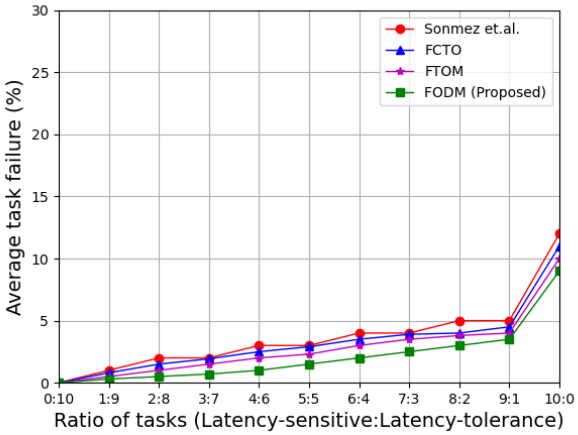


FIGURE 10. Performance analysis based on latency-sensitive and latency-tolerant task ratio with average task failure rate.

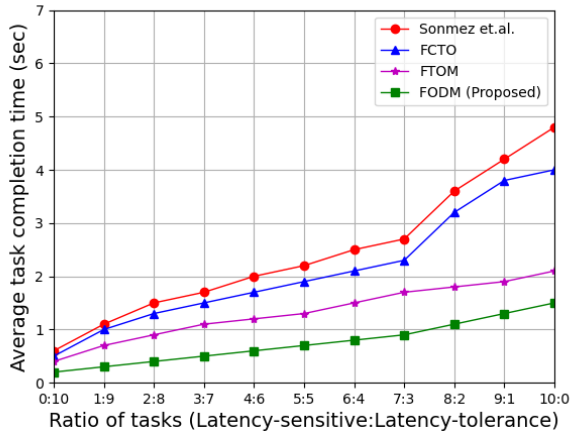


FIGURE 11. Performance analysis based on latency-sensitive and latency-tolerant task ratio with average task completion time.

with the Sonmez et al., FCTO, and FTOM approaches respectively.

The latency-sensitive and latency-tolerant are complementary to each other means, when any task is of a latency-sensitive type, then it never comes under the latency-tolerant type, and also similarly latency-tolerable tasks are never sensitive to the latency in the 5G environment. So, we have considered the ratio of latency-sensitive and latency-tolerable tasks with respect to the average task failure rate and average task completion time. The starting index of the x-axis begins with 0:10, which means we are considering the latency-tolerable offloaded tasks. The x-axis ends with 10:0, which means we are considering the latency-sensitive offloaded tasks. In Fig. 10, all the latency-tolerant tasks are executed at any time with minimum or maximum latency, resulting in very low task failure rates as 0.3%, 0.24%, 0.23%, and 0.22% for the approaches Sonmez et al., FCTO, FTOM, and FODM respectively. But all the latency-sensitive tasks cannot wait for the maximum time to be executed, resulting in maximum task failure rates as 15.59%, 10.46%, 8.92%, and 6.54% for the approaches Sonmez et al., FCTO, FTOM, and FODM respectively. Our proposed approach performs better when we are considering the latency-sensitive offloaded tasks, as we have used the optimal inference engine to make the faster decision, which helps in the faster execution of the latency-sensitive tasks without waiting for a longer time in the 5G network. From the above discussion, we can conclude that latency-sensitive tasks are more prone to failures, and need to be handled carefully in a 5G environment.

In Fig. 11, the average task completion time is represented by the ratio of tasks. The simulation result is considered by taking heavier task sizes in a latency-sensitive environment. Here the latency-tolerant tasks are having very little completion time and all the approaches are taken within 0 to 1 second time to complete the execution of the task. But when we are considering the latency-sensitive tasks, results in a clear distinction among all the approaches. The best performance is produced by our proposed FODM approach.

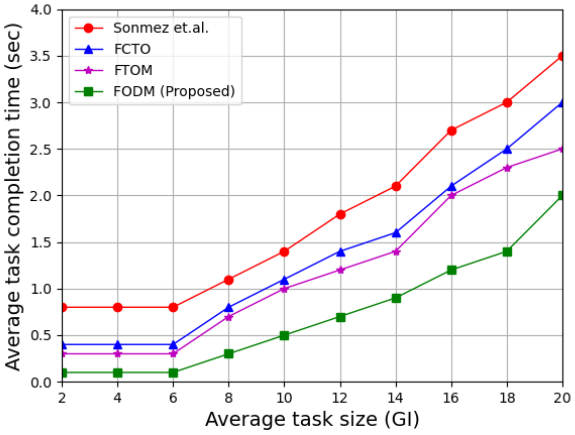


FIGURE 12. Performance evaluation using average task completion time with task size.

Our proposed approach reduces the task completion time by 69.4%, 64.59%, and 46.01% when compared with the Sonmez et al., FCTO, and FTOM approaches respectively.

In Fig. 12, FODM performance is analyzed by considering the task completion time in the y-axis with the size of the task in the x-axis. From the task size 2 to 6 GI, all the approaches are performing in a similar pattern, when the task size increases more than 6 GI shows the distinct variation of the other approaches in comparison with the proposed FODM. As we have discussed earlier, the completion time is dependent on the processing time, due to the faster processing approach proposed in our method, the resultant completion time also becomes less in comparison with the other approaches. Here we have analyzed the approaches by considering the offloading process only in MEC.

The utilization percentage of the server is analyzed in corresponding with the task size is explained in Fig. 13. Here our proposed approach does not perform well as we have not considered the VM utilization in the MEC server. When the size of the task is small, the Sonmez et al. and our approach behave similarly, but as the task size is gradually increasing

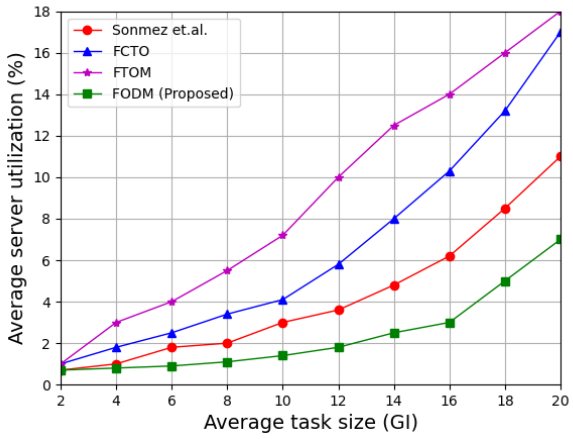


FIGURE 13. Performance evaluation using average server utilization with task size.

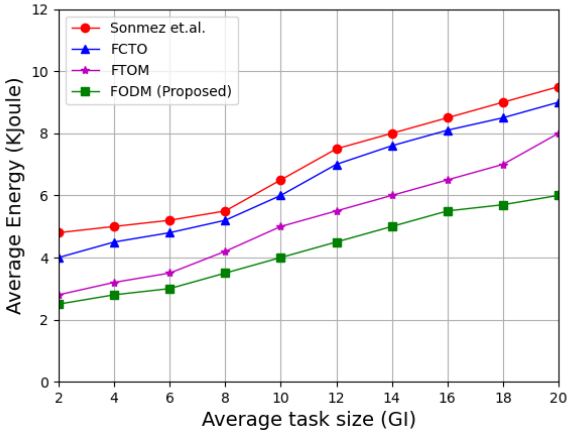


FIGURE 14. Performance evaluation using average energy with task size.

more than the 4 GI, the Sonmez et al. approach performs better than our approach. When the task size is 20 GI, the server utilization of FODM, Sonmez et al. FCTO and FTOM are 7.21%, 11.52%, 17.15%, and 18.24% respectively. From this result, we can conclude that the FTOM approach performs well in comparison with the other approaches as FTOM has implemented the VM utilization in MEC servers in an effective manner.

Fig. 14 shows the energy consumption plot with respect to the size of the task. The minimization of energy consumption plays a vital role when we are considering the computationally intensive task in a resource constraint environment 5G environment. Mobile devices are associated with limited energy, so the main aim is to how effectively utilize the energy for getting better performance. In the graph, Sonmez et al. and FCTO approaches are consuming more energy than the FTOM and FODM approaches. At the beginning when the sizes of the tasks are small, then the energy consumption in FTOM and FODM are resulting in similar performances, but when the task size is more than 6 GI, the plot distinguishes the approaches clearly. The FODM scheme performs well as we have used the optimal inference engine to take faster decisions which help in the reduction of the energy consumption by 16.79%, 34.71%, and 39.53% with the approaches FTOM, FCTO, and Sonmez et al. respectively.

IX. CONCLUSION

The development of task offloading strategies for mobile edge computing has become a center of attraction because of its multi-faceted challenges at various levels. At the device level, taking an optimal decision to offload a task or not, is extremely challenging in a real environment due to the uncertainty present in profiling parameters that affect offloading of tasks. Further, in a realistic environment, with the increase in the number of offloading parameters, the recent state-of-the-art fuzzy-based approaches face degradation in performance due to exponential searching time complexity at the inference engine. In this context, an optimal fuzzy-based strategy, FODM, is proposed using classification and regression tree-based inference mechanism to minimize searching time complexity and optimize the decision-making time of offloading which is specifically beneficial for delay-sensitive tasks in a real-time network. The proposed FODM is compared with state-of-the-art fuzzy-based approaches for task offloading using different performance metrics through simulations and an in-depth mathematical analysis is carried out to prove its efficacy. The extension of this work will be carried out on multiple-users and multi-server environments by embedding centralized task queuing and edge orchestration methods.

REFERENCES

[1] D. B. Hoang and L. Chen, "Mobile cloud for assistive healthcare (MoCAsH)," in *Proc. IEEE Asia–Pacific Services Comput. Conf.*, Dec. 2010, pp. 325–332.
[2] G. Sun and J. Shen, "Facilitating social collaboration in mobile cloud-based learning: A teamwork as a service (TaaS) approach," *IEEE Trans. Learn. Technol.*, vol. 7, no. 3, pp. 207–220, Jul. 2014.
[3] H. Flores, "Mobile code offloading: From concept to practice and beyond," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 80–88, Mar. 2015.
[4] Z. Luo, M. LiWang, Z. Lin, L. Huang, X. Du, and M. Guizani, "Energy-efficient caching for mobile edge computing in 5G networks," *Appl. Sci.*, vol. 7, no. 6, p. 557, May 2017.
[5] V. Sharma, V. P. Rai, and K. K. Sharma, "Edge computing: Needs, concerns and challenges," *Int. J. Sci. Eng. Res.*, vol. 8, no. 4, pp. 154–166, 2017.
[6] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2016, pp. 20–26.
[7] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.
[8] V. Nguyen, T. T. Khanh, T. D. T. Nguyen, C. S. Hong, and E.-N. Huh, "Flexible computation offloading in a fuzzy-based mobile edge orchestrator for IoT applications," *J. Cloud Comput.*, vol. 9, no. 1, pp. 1–18, Dec. 2020.
[9] M. D. Hossain, T. Sultana, V. Nguyen, W. U. Rahman, T. D. T. Nguyen, L. N. T. Huynh, and E.-N. Huh, "Fuzzy based collaborative task offloading scheme in the densely deployed small-cell networks with multi-access edge computing," *Appl. Sci.*, vol. 10, no. 9, p. 3115, Apr. 2020.
[10] C. Sonmez, A. Ozgovde, and C. Ersoy, "Fuzzy workload orchestration for edge computing," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 769–782, Jun. 2019.
[11] T. Subramanya, D. Harutyunyan, and R. Riggio, "Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks," *Comput. Netw.*, vol. 166, Jan. 2020, Art. no. 106980.

[12] W. Sun, J. Liu, and Y. Yue, "AI-enhanced offloading in edge computing: When machine learning meets industrial IoT," *IEEE Netw.*, vol. 33, no. 5, pp. 68–74, Sep. 2019.

[13] Z. Lv, D. Chen, R. Lou, and Q. Wang, "Intelligent edge computing based on machine learning for smart city," *Future Gener. Comput. Syst.*, vol. 115, pp. 90–99, Feb. 2021.

[14] N. Shan, Y. Li, and X. Cui, "A multilevel optimization framework for computation offloading in mobile edge computing," *Math. Problems Eng.*, vol. 2020, pp. 1–17, Jun. 2020.

[15] H. Wu, W. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 7, pp. 1464–1480, Jul. 2019.

[16] G. Li, Q. Lin, J. Wu, Y. Zhang, and J. Yan, "Dynamic computation offloading based on graph partitioning in mobile edge computing," *IEEE Access*, vol. 7, pp. 185131–185139, 2019.

[17] J. Sheng, J. Hu, X. Teng, B. Wang, and X. Pan, "Computation offloading strategy in mobile edge computing," *Information*, vol. 10, no. 6, pp. 1–20, 2019.

[18] W. Sun, J. Liu, Y. Yue, and H. Zhang, "Double auction-based resource allocation for mobile edge computing in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4692–4701, Oct. 2018.

[19] X. Huang, K. Xu, C. Lai, Q. Chen, and J. Zhang, "Energy-efficient offloading decision-making for mobile edge computing in vehicular networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 1–16, Dec. 2020.

[20] B. B. Bista, J. Wang, and T. Takata, "Probabilistic computation offloading for mobile edge computing in dynamic network environment," *Internet Things*, vol. 11, Sep. 2020, Art. no. 100225.

[21] J. Cheng and D. Guan, "Research on task-offloading decision mechanism in mobile edge computing-based Internet of Vehicle," *EURASIP J. Wireless Commun. Netw.*, vol. 2021, no. 1, pp. 1–14, Dec. 2021.

[22] S. Yang, G. Lee, and L. Huang, "Deep learning-based dynamic computation task offloading for mobile edge computing networks," *Sensors*, vol. 22, no. 11, pp. 1–18, 2022.

[23] F. Basic, A. Aral, and I. Brandic, "Fuzzy handoff control in edge offloading," in *Proc. IEEE Int. Conf. Fog Comput. (ICFC)*, Jun. 2019, pp. 87–96.

[24] J. Almutairi and M. Aldossary, "A novel approach for IoT tasks offloading in edge-cloud environments," *J. Cloud Comput.*, vol. 10, no. 1, pp. 1–19, Apr. 2021.

[25] T. T. Khanh, V. Nguyen, and E.-N. Huh, "Fuzzy-based mobile edge orchestrators in heterogeneous IoT environments: An online workload balancing approach," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–19, Aug. 2021.

[26] M. D. Hossain, "Fuzzy decision-based efficient task offloading management scheme in multi-tier MEC-enabled networks," *Sensors*, vol. 21, no. 4, pp. 1–26, 2021.

[27] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Aug. 2016.

[28] Q. Tang, H. Lyu, G. Han, J. Wang, and K. Wang, "Partial offloading strategy for mobile edge computing considering mixed overhead of time and energy," *Neural Comput. Appl.*, vol. 32, no. 19, pp. 15383–15397, Oct. 2020.

[29] V. Nguyen, T. T. Khanh, T. Z. Oo, N. H. Tran, E.-N. Huh, and C. S. Hong, "Latency minimization in a fuzzy-based mobile edge orchestrator for IoT applications," *IEEE Commun. Lett.*, vol. 25, no. 1, pp. 84–88, Jan. 2021.

[30] Z. H. Abbas, "Computational offloading in mobile edge with comprehensive and energy efficient cost function: A deep learning approach," *Sensors*, vol. 21, no. 10, pp. 1–18, 2021.

[31] M. Elkalamawy, "Effect of mobile device profiling in mobile computation offloading," *Int. J. Comput. Sci. Inf. Secur.*, vol. 16, no. 9, pp. 39–45, 2018.

[32] L. T. Hong Lan, T. M. Tuan, T. T. Ngan, L. H. Son, N. L. Giang, V. T. Nhu Ngoc, and P. V. Hai, "A new complex fuzzy inference system with fuzzy knowledge graph and extensions in decision making," *IEEE Access*, vol. 8, pp. 164899–164921, 2020.

[33] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[34] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[35] E. Van Broekhoven and B. D. Baets, "Fast and accurate center of gravity defuzzification of fuzzy system outputs defined on trapezoidal fuzzy partitions," *Fuzzy Sets Syst.*, vol. 157, no. 7, pp. 904–918, Apr. 2006.

[36] S. Husain, Y. Ahmad, M. Sharma, and S. Ali, "Comparative analysis of defuzzification approaches from an aspect of real life problem," *IOSR J. Comput. Eng.*, vol. 19, no. 6, pp. 19–25, 2017.

[37] Y.-Y. Guh, R.-W. Po, and E. S. Lee, "The fuzzy weighted average within a generalized means function," *Comput. Math. With Appl.*, vol. 55, no. 12, pp. 2699–2706, Jun. 2008.

[38] P. Van Den Broek and J. Noppen, "Exact membership functions for the fuzzy weighted average," in *Computational Intelligence*, vol. 343. Berlin, Germany: Springer, 2011, pp. 85–99.

[39] I. Lera, C. Guerrero, and C. Juiz, "YAFS: A simulator for IoT scenarios in fog computing," *IEEE Access*, vol. 7, pp. 91745–91758, 2019.

[40] T. K. Mishra, K. S. Sahoo, M. Bilal, S. C. Shah, and M. K. Mishra, "Adaptive congestion control mechanism to enhance TCP performance in cooperative IoV," *IEEE Access*, vol. 11, pp. 9000–9013, 2023.

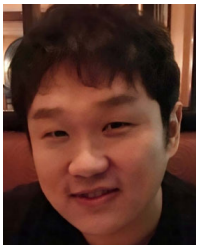
SASMITA RANI BEHERA received the B.Tech. degree in computer science and engineering from the Biju Patnaik University of Technology (BPUT), Odisha, India, in 2003, and the M.Tech. degree in computer science from the National Institute of Technology, Rourkela, India, in 2009. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, BPUT. She is an Assistant Professor with the Department of Computer Science and Engineering, Parala Maharaja Engineering College (Government), Berhampur, India. She has teaching and research experience of more than 12 years. She has published research papers in reputed international journals and conferences. Her research interests include the IoT, edge computing, distributed systems, and software engineering. She is a Life Member of the Indian Society of Technical Education.

NIRANJAN PANIGRAHI received the M.Tech. and Ph.D. degrees in computer science and engineering from the National Institute of Technology (NIT), Rourkela, India, in 2009 and 2017, respectively. He has a total of 18 years of teaching and research experience. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Parala Maharaja Engineering College (PMEC), Odisha, Berhampur, India, an autonomous college of government. He is in-charge of the High-Performance Computing Laboratory, Centre-of-Excellence, PMEC. He is a member of the SWAYAM-NPTEL Mapping Committee, Biju Patnaik University of Technology, Rourkela. He has research publications in reputed journals of IET, Elsevier, and Springer, and has also contributions to many book chapters and conferences. His research interests include cloud and edge computing, wireless sensor networks, applied machine learning, parallel algorithm, and soft computing. He is a Life-Time Member of ISTE, IAENG, and UACEE.

SOURAV KUMAR BHOI received the M.Tech. degree in information security and the Ph.D. degree from the Department of Computer Science and Engineering, National Institute of Technology (NIT), Rourkela, India, in 2013 and 2017, respectively. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Parala Maharaja Engineering College (Government), Berhampur, India. He has nearly ten years of teaching and research experience. He also completed a three-month CSIR Summer Research Training Program in online mode from NEIST, Jorhat, Assam (Government of India), in August 2020. He acted as a Co-PI in a project of NPIU, MHRD, Government of India. He is an India Book of Record Holder for his research work, in 2022. He has more than 110 publications in reputed international journals, conferences, books, book chapters, technical articles, patents, and theses. His research interests include machine learning, the Internet of Things, edge and fog computing, ad hoc and sensor networks, and information security. He is a member of many professional bodies, such as a member of IAENG, a Life Member of CSI, an Associate Member of IEI, and a fellow of SIESRP. He acted as a member of TPC and the session chair of many international conferences. He received the prestigious IET Premium Award from *IET Networks* journal, in 2016. He also received many other awards and honors, such as the University Foundation Day Faculty Research Award in CSE and the Sadananda Memorial Award from the Institution of Engineers (India), in 2021 and 2020, respectively. He was a reviewer for many international journals and conferences. He delivers several invited talks in reputed organizations.

MUHAMMAD BILAL (Senior Member, IEEE) received the Ph.D. degree in information and communication network engineering from the School of Electronics and Telecommunications Research Institute (ETRI), Korea University of Science and Technology, in 2017. From 2017 to 2018, he was with Korea University, where he was a Postdoctoral Research Fellow with the Smart Quantum Communication Center. In 2018, he joined the Hankuk University of Foreign Studies, South Korea, where he is currently an Associate Professor with the Division of Computer and Electronic Systems Engineering. He is the author/coauthor of more than 90 articles published in renowned journals, including IEEE Transactions on Intelligent Transportation Systems, IEEE Internet of Things Journal, IEEE Systems Journal, IEEE Transactions on Industrial Informatics, IEEE Transactions on Network and Service Management, and IEEE Transactions on Network Science and Engineering, one book editorship, two book chapters, seven published proceedings articles, three issued U.S. patents, and six Korean patents. His research interests include network optimization, cyber security, the Internet of Things, vehicular networks, information-centric networking, digital twin, artificial intelligence, and cloud/fog computing. He has served as a Technical Program Committee Member for many international conferences, including the IEEE VTC, the IEEE ICC, ACM SigCom, and the IEEE CCNC. He is also an Editorial Board Member of IEEE Transactions on Intelligent Transportation Systems, IEEE Future Directions in Technology, Policy, and Ethics Newsletter, *Alexandria Engineering Journal* (Elsevier), *Intelligent Automation and Soft Computing*, *Frontiers in Communications and Networks*, and *Frontiers in the Internet of Things*, and the Co-Editor-in-Chief of the *International Journal of Smart Vehicles and Smart Transportation*.

KSHIRA SAGAR SAHOO (Senior Member, IEEE) received the master's degree in information and communication technology from the Indian Institute of Technology Kharagpur, India, in 2014, and the Ph.D. degree in computer science and engineering from the National Institute of Technology, Rourkela, India, in 2019. He is currently working as an Assistant Professor with SRM University, Amaravati, AP, India. He has been Associated with ADS Laboratory, Umeå University, Sweden, since October 2022. He has more than five years of teaching experience, two years of industry experience, and four years of research experience. He has published more than 80 research papers in various top international journals and conferences, including the IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Network and Service Management, IEEE Systems Journal, IEEE Internet of Things Journal, IEEE Sensors, *ACM Transactions on Multimedia Computing, Communications, and Applications*, and *Future Generation Computer Systems* (Elsevier). His research interests include future-generation network infrastructure, such as SDN, edge computing, the IoT, anomaly detection, and the industrial IoT. He is a Senior Member of the IEEE Computer Society and an Associate Member of the Institute of Engineers (IE), India.



DAEHAN KWAK (Member, IEEE) received the M.S. degree from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2008, and the Ph.D. degree in computer science from Rutgers University, New Brunswick, NJ, USA, in 2017. During his graduate studies, he was with the Networked Systems and Security (Disco) Laboratory, Rutgers University, and the Telematics and USN Research Division, Electronics and Telecommunications Research Institute (ETRI). He was a Research Staff with the UWB Wireless Research Center, Inha University, the Wireless Internet and Networks Laboratory, KAIST, and the Network Management and Optimization Laboratory, Yonsei University. He is currently an Assistant Professor with the Department of Computer Science and Technology, Kean University, Union, NJ, USA. His research interests include intelligent systems, cyber-physical systems, the IoT, systems and networking, wireless and sensor systems, mobile and vehicular computing, smart transportation, and smart health. He served as the Guest Editor for the *Electronics* and *Wireless Communications and Mobile Computing* journals. He is a reviewer for several international journals and conferences and undergraduate proceedings.