IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 September 2023, accepted 14 October 2023, date of publication 20 October 2023, date of current version 30 October 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3326244*

RESEARCH ARTICLE

# Workload Orchestration in Multi-Access Edge Computing Using Belief Rule-Based Approach

**MOHAMMAD NEWAJ JAMIL[1], MOHAMMAD SHAHADAT HOSSAIN[2], (Senior Member, IEEE), RAIHAN UL ISLAM[1], AND KARL ANDERSSON[1], (Senior Member, IEEE)**

[1]Pervasive and Mobile Computing Laboratory, Luleå University of Technology, 93187 Skellefteå, Sweden
[2]Department of Computer Science and Engineering, University of Chittagong, Chattogram 4331, Bangladesh

Corresponding author: Mohammad Newaj Jamil (mohammad.newaj.jamil@ltu.se)

This work was supported in part by Erasmus Mundus Joint Masters Degree in GrEen NetworkIng And cLoud computing (GENIAL) under Grant 610619-EPP-1-2019-1-FR-EPPKA1-JMD-MOB, and in part by the VINNOVA Project 5G Edge Innovations for Mining under Grant 2021-03663. The work of Mohammad Newaj Jamil was supported in part by the Pervasive and Mobile Computing Laboratory, Luleå University of Technology, Sweden.

**ABSTRACT** Multi-access Edge Computing (MEC) is a standard network architecture for edge computing, which is proposed to handle enormous computation demands from emerging resource-intensive and latency-sensitive applications and services as well as accommodate Quality of Service (QoS) requirements for ever-growing users through computation offloading. Since the demand of end-users is unknown in a rapidly changing dynamic environment, processing offloaded tasks in a non-optimal server can deteriorate QoS due to high latency and increasing task failures. In order to deal with such a challenge in MEC, a two-stage Belief Rule-Based (BRB) workload orchestrator is proposed to distribute the workload of end-users to optimum computing units, support strict QoS requirements, ensure efficient utilization of computational resources, minimize task failures, and reduce the overall service time. The proposed BRB workload orchestrator decides the optimal execution location for each offloaded task from User Equipment (UE) within the overall MEC architecture based on network conditions, computational resources, and task requirements. EdgeCloudSim simulator is used to conduct comprehensive simulation experiments for evaluating the performance of the proposed BRB orchestrator in contrast to four workload orchestration approaches from the literature with different types of applications. Based on the simulation experiments, the proposed workload orchestrator outperforms state-of-the-art workload orchestration approaches and ensures efficient utilization of computational resources while minimizing task failures and reducing the overall service time.

**INDEX TERMS** Multi-access edge computing (MEC), task offloading, workload orchestrator, belief rule base (BRB), performance evaluation.

## I. INTRODUCTION

In recent years, there has been a stupendous proliferation of smart User Equipment (UE), such as smartphones, smartwatches, and smart glasses. Besides, diverse and immensely demanding applications and services, such as smart surveillance, augmented reality, virtual reality, voice recognition, automated transportation, smart farming, smart supply chain management, smart healthcare, and autonomous driving, are getting extremely popular. Hence, data traffic is growing exponentially as more and more latency-sensitive and computation-intensive tasks have been generated from these sophisticated applications and services [1]. However, the processing power, storage capacity, and battery life of UE are limited to locally process computation-intensive tasks, which require high computational resources and energy [2]. Hence, Cloud Computing has been proposed to process computation-intensive tasks generated from UE by utilizing the powerful computing and storage capabilities of cloud servers over a Wide Area Network (WAN) [3]. It can fulfill

The associate editor coordinating the review of this manuscript and approving it for publication was Rodrigo S. Couto.

service demands of resource-intensive applications and tackle the problem of inadequate processing capability and limited battery of UE [4]. However, Cloud Computing cannot meet service demands for latency-sensitive and context-aware applications and services because the latency of WAN is high [5], congestion occurs if the amount of data generated by UE increases that eventually resulting in packet losses [6], and it works centralized and remotely as cloud servers are generally far away from UE [7].

In order to handle a massive amount of diverse data traffic, enhance computation capabilities, and reduce communication latency, Multi-access Edge Computing (MEC) has emerged that brings computational resources at the edge of the network in close vicinity to UE to support real-time applications and services [8]. By deploying edge servers closer to UE at the network edge, applications and services can be served with low latency while ensuring a good Quality of Service (QoS) to end-users. MEC can reduce long communication latency while relieving core network congestion that occurs in Cloud Computing. A multi-tier MEC architecture is usually composed of three tiers and different sorts of networks [9], as shown in Fig. 1.

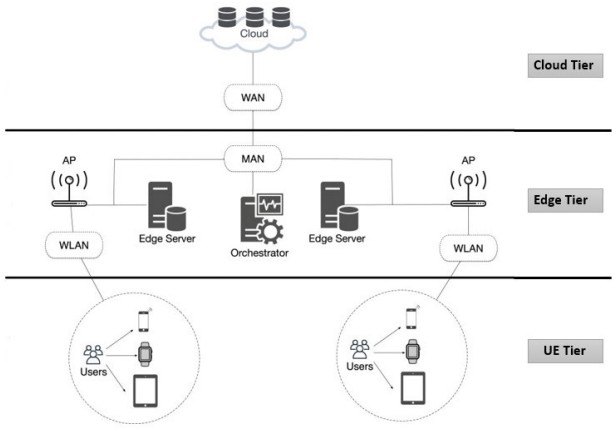



**FIGURE 1.** A multi-tier MEC architecture.

As depicted in Fig. 1, the first tier is UE Tier, which contains different types of UE that communicate with the nearest (local) edge server over Wireless Local Area Network (WLAN). The second tier is Edge Tier, which comprises several edge servers that are interconnected through Metropolitan Area Network (MAN) and an orchestrator that knows the computational resources of edge and cloud servers and network resources and is responsible for managing workload by optimally distributing each offloaded task from UE to an edge or cloud server. The third tier is Cloud Tier, which consists of global cloud server(s) that provide services through WAN.

A salient trait of MEC is computation offloading, which allows UE to perform a task that is not possible for UE to process locally due to resource constraints. It facilitates UE to consume less battery and improve computation performance. A task can be offloaded from UE to the nearest (local) edge server (in this case, UE accesses the server with one hop

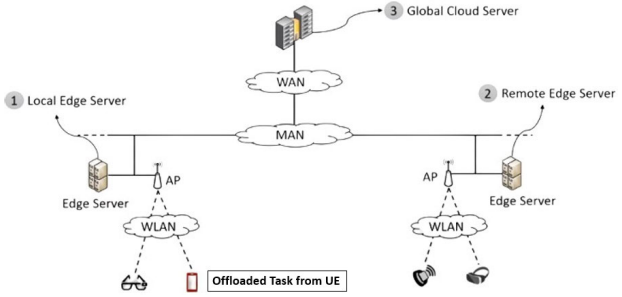



**FIGURE 2.** Task offloading from user equipment to edge or cloud servers.

over WLAN), a neighboring (remote) edge server (in this case, UE accesses the server with two hops over WLAN and MAN), and the global cloud server (in this case, UE accesses the server with three hops over WLAN, MAN, and WAN) [9], as shown in Fig. 2.

However, deciding whether to offload a task from UE to the nearest (local) edge server, a neighboring (remote) edge server, or the global cloud server is challenging because MEC faces a rapidly changing dynamic environment at the network edge due to uncertainty [10]. For instance, end-users moving from one location to another while connecting to different WLANs result in high mobility, whereas a variety of tasks generated by latency-sensitive and computation-intensive applications result in intermittent traffic. As a consequence, the state of the network condition keeps changing, and the computational capacities of edge and cloud servers keep fluctuating [11], [12]. Besides, if offloaded tasks from UE are distributed unevenly and randomly within the overall MEC infrastructure, it will lead to imbalance workloads between edge and cloud servers, increase task latency due to improper task offloading, and impair the performance of MEC [13]. For instance, when a large number of tasks are offloaded to an edge server concurrently, it gets overloaded and causes network congestion though this overloaded problem can be alleviated if those tasks are optimally distributed among edge servers. Therefore, in this heterogeneous dynamic environment, the overall efficiency and scalability of MEC depend on correctly determining the optimal target server for each offloaded task from UE. Ensuring efficient processing of dynamic flow of requests and scalable management of computational resources while satisfying end-user demands and providing a good QoS is challenging in MEC.

The primary factors that should be considered for an effective workload orchestration in MEC are CPU utilization of edge servers and network conditions along the route to these servers, which can be exceptionally dynamic under unforeseen variations of workloads [9]. For example, bandwidth fluctuation occurs in a shared network due to an increase in the number of users, while CPU utilization of a Virtual Machine (VM) changes frequently based on the type of task assigned to it [14]. Besides, MEC consists of various parameters from the MEC infrastructure as well as from the UE and the application running on it, which further convolute the workload orchestration problem [9]. Due to

the rapidly changing dynamic environment in MEC, its associated uncertainty, and numerous parameters, it is hard to build a mathematical model for multi-constraint optimization and decide the execution location for an offloaded task from UE [14]. Besides, the time complexity of solving a multi-constraint optimization problem grows exponentially when the frequency of end-users mobility is high, which, in turn, leads to a rise in latency and inefficient resource utilization [15], [16]. Since the number of offloading requests from different types of UE is unknown in advance and the state of network conditions and computational resources change rapidly, the traditional offline optimization techniques are not suitable for the workload orchestration problem [9]. In this regard, the Belief Rule-Based (BRB) approach can be a suitable candidate to deal with the dynamic and uncertain environment in MEC since it can make decisions by processing multiple parameters comprising both qualitative and quantitative data and handling the uncertainty in dynamic environments without a detailed mathematical model [17]. Besides, computational complexity, which is one of the important requirements for real-time problems, is low for this approach in comparison with other decision-making algorithms [18]. Moreover, complex operational details can be stated in a high-level human understandable format using belief rules [17], [18]. Therefore, a BRB workload orchestrator is proposed in this study to solve the workload orchestration problem in MEC.

The key contributions of this study are as follows.

1) While the BRB approach has been applied to address various problems in several areas, it has not been utilized for the workload orchestration problem in MEC so far. Hence, in this study, the first BRB workload orchestrator is proposed for a multi-tier MEC infrastructure to efficiently handle the workload orchestration problem and meet the diverse needs of end-users.
2) In this study, a two-stage BRB workload orchestrator is considered. Since a BRB can be either conjunctive or disjunctive, both types of BRBs are established in the first and second stages of the BRB orchestrator to determine the most appropriate BRBs in both stages. However, determining whether to use a conjunctive or disjunctive BRB in the first and second stages of the BRB orchestrator is tricky since the BRB orchestrator might perform well for one application when it uses a conjunctive BRB in the first stage and a disjunctive BRB in the second stage, but it might not perform well for another application when it uses the same combination of BRBs. To address this issue, all possible combinations of conjunctive and disjunctive BRBs in the first and second stages of the BRB orchestrator are categorized based on the delay sensitivity of applications so that the best combination of BRBs for all sorts of applications can be identified and the proposed BRB orchestrator can perform equally well for all types of applications. The performances of the two-stage BRB orchestrator for all possible combinations of BRBs are evaluated through a detailed simulation experiment under high workloads in terms of three performance metrics. Based on the performance evaluation, the best combination of BRBs for the two-stage BRB workload orchestrator is determined.
3) After determining the best combination of conjunctive and disjunctive BRBs for the two-stage BRB workload orchestrator, another detailed simulation experiment is conducted to assess the performance of the BRB workload orchestrator with the best combination of BRBs in comparison to four workload orchestration approaches from the literature based on three performance metrics. Simulation results demonstrate the potency of the proposed workload orchestrator in contrast to other approaches and affirm that the BRB orchestrator reduces average service time, minimizes the average number of task failures, and provides lower average VM utilization by efficiently utilizing all VMs on edge servers and balancing the workload among edge servers.

**TABLE 1. List of abbreviations.**

| Notation | Description |
|---|---|
| MEC | Multi-access Edge Computing |
| QoS | Quality of Service |
| BRB | Belief Rule Base |
| UE | User Equipment |
| WLAN | Wireless Local Area Network |
| MAN | Metropolitan Area Network |
| WAN | Wide Area Network |
| VM | Virtual Machine |
| FLB | Fuzzy Logic Base |
| ER | Evidential Reasoning |
| FIS | Fuzzy Inference System |

The remainder of this article is structured as follows. Section II presents a comprehensive review of related work on workload orchestration, while a short overview of the BRB approach and the proposed two-stage BRB workload orchestrator is presented in Section III. Section IV presents the simulated environment that is considered to demonstrate how MEC fits into real-world implementation, while the performance assessment of the two-stage BRB workload orchestrator is presented in Section V. Finally, Section VI concludes the paper and indicates future work. The abbreviations used throughout the paper are listed in Table 1 for better readability.

## II. RELATED WORK

Several workload orchestration approaches have been proposed in edge computing from various perspectives. Wang et al. [19] developed an online approximation algorithm in edge computing that took into account communication and computational resources to balance the load and minimize resource utilization for enhancing application performance. However, this study did not take into account the service

delay. Rodrigues et al. [20] proposed a hybrid method to reduce service delay in edge computing, where the impact of computational and communication demands of tasks was investigated, and the proposed approach was evaluated under realistic conditions by mathematical modeling. However, this study did not take into account delay constraints of applications and task offloading to a cloud server for workload balancing. Fan and Ansari [21] designed an allocation strategy considering both computation and communication delays to reduce service delay in edge computing, where the impact of overloaded VMs on processing time was investigated, and the proposed approach was evaluated with various applications. However, this study did not consider latency-sensitive applications and demonstrate the efficacy of the heterogeneity of VMs in regard to service time. Nan et al. [22] used the Lyapunov technique to develop an online optimization algorithm for task offloading by leveraging resources at the edge, which ensured the processing of a task within a specified time. However, this study did not take into account the impact of computational and communication demands for delay-sensitive applications. Xu et al. [23] developed a model for resource allocation in edge computing to maximize resource utilization and minimize task execution latency. However, this study did not consider upload/download data for applications that are crucial to overall service time. Scoca et al. [24] developed a scour-based algorithm for scheduling offloaded tasks in edge computing by considering both computation and communication parameters and heterogeneous VMs, where the most powerful VMs were used to process heavy computational tasks. However, this study did not take into account server utilization as a key parameter, which could deteriorate the overall service time. Taneja and Davy [25] proposed a resource-aware placement to improve the service time in edge computing, where computational resources at the edge were ranked according to their capabilities, and tasks were assigned to an appropriate server on the basis of task requirements, such as CPU and bandwidth. However, this study did not consider the latency sensitivity of applications. Bittencourt et al. [26] developed an edge-ward placement algorithm for workload orchestration in edge computing, where application requests are processed either in an edge or cloud server according to CPU capacity and delay requirements. However, this study did not consider network congestion since a steady network delay for WLAN, MAN, and WAN communication was used in this study.

From the above discussion, it can be observed that the majority of past studies had some research gaps. For instance, some studies did not consider network congestion, some studies did not consider crucial application characteristics such as delay/latency sensitivity, upload/download data, task interarrival, as well as CPU utilization on edge servers, some studies did not address imbalance workload on edge servers when a vast number of UE run a variety of applications simultaneously, and some studies ignored task offloading either to a neighboring (remote) edge server or global cloud server, which could hamper the optimal distribution of the workload in edge computing since spare computing resources of neighboring (remote) edge servers and adequate computing resources of the global cloud server are not utilized in an efficient way. To address all these issues, Sonmez et al. [9] proposed a Fuzzy Logic-Based (FLB) workload orchestration in edge computing, where a two-stage FLB workload orchestrator was utilized to determine the location of an offloaded task among the local edge, neighboring/remote edge, and global cloud server based on network conditions, computational resources, and application characteristics and four types of applications were considered to evaluate the proposed orchestrator. Almutairi et al. [14] also proposed an FLB orchestrator for task offloading similar to [9], where the only difference is the authors considered resource heterogeneity in addition to application characteristics and network conditions. However, in both studies, the FLB workload orchestrator uses the same fuzzy rule bases for both delay-sensitive and delay-tolerant applications. Since task requirements for both types of applications are different, it is not feasible to use the same fuzzy rule bases to distinguish tasks generated by delay-sensitive and delay-tolerant applications and determine the optimal target server for processing those tasks. Zheng et al. [27] proposed a Deep Reinforcement Learning (DRL) based workload scheduling for MEC, where the authors adopted the Deep-Q-Network (DQN) algorithm to solve the complexity and high dimension of the workload scheduling problem. Yamansavascilar et al. [28] also proposed a DRL-based task orchestrator for MEC, where the authors applied the Double DQN (DDQN) algorithm to meet the dynamic needs of different application types. However, the DRL-based approach often suffers from the delayed action effect due to the stochastic nature of the MEC environment, distinct application types, and user mobility, which hampers the overall efficiency of MEC.

In recent years, the BRB approach [17] has been successfully applied to address different problems in networking such as network security situation prediction [29], network intrusion detection [30], health prediction for a sensor network [31], fault prediction for a Wireless Sensor Network (WSN) [32], and fault diagnosis of a WSN [33]. Besides, it has been found more promising than the FLB approach for uncertain nonlinear systems [34], [35], [36]. Moreover, it does not suffer from the delayed action effect like the DRL-based approach. However, the BRB approach has not been studied yet to address workload orchestration issues in MEC.

Hence, inspired by the study in [9], a two-stage BRB workload orchestrator is proposed in this study to deal with a highly dynamic environment in MEC and alleviate the overloaded problem caused by the imbalanced distribution of computational resources by balancing the workload between edge and cloud servers, as well as minimize task failures and reduce overall service time. The proposed BRB workload orchestrator uses belief rules to specify the required workload

orchestration operations with regard to network conditions, computational resources, and properties of offloaded tasks from UE and decide the execution location for offloaded tasks among the local edge, neighboring/remote edge, and global cloud server, which refrain system administrators from distinguishing operational states and assigning policies to those states. Besides, the best combination of BRBs (between conjunctive and disjunctive) for the two-stage BRB workload orchestrator is determined based on the performance comparison of all possible combinations of BRBs, which facilitates the BRB orchestrator to perform equally well for all sorts of applications.

## III. METHODOLOGY

This section presents a short overview of the BRB approach and the proposed two-stage BRB workload orchestrator.

### A. BELIEF RULE-BASED APPROACH

The BRB approach is composed of a conjunctive or disjunctive BRB that represents the initial knowledge base and Evidential Reasoning (ER) that functions as an inference engine by processing diverse and uncertain data [37]. A belief rule can represent a non-linear causal association between the antecedent and consequent attributes [38]. The logical connective of the antecedent attributes in a belief rule can be either AND ($\wedge$) or OR ($\vee$), which characterizes the conjunctive or disjunctive assumption of the belief rule. Under the conjunctive assumption, each antecedent attribute can have different number of referential values [18]. However, the prerequisite of the disjunctive assumption is all antecedent attributes must contain the same number of referential values [39]. In a conjunctive BRB, the number of belief rules will grow exponentially if the number of antecedent attributes and/or the number of referential values of antecedent attributes increases since the conjunctive assumption considers all possible combinations of referential values of antecedent attributes, which results in the combinatorial explosion problem [40]. However, disjunctive BRB does not go through this problem since all antecedent attributes contain the same number of referential values and the total number of belief rules is equal to the number of referential values of the antecedent attributes. A large conjunctive BRB needs more computational time, while a disjunctive BRB needs less computational time due to less number of disjunctive rules. A conjunctive or disjunctive BRB can be constructed in four ways such as by acquiring conventional or belief rules from expert knowledge, deriving rules by inspecting historical data, using random belief rules if there is no prior knowledge, and using previous conventional rule base or BRB if available [41].

Input transformation, rule activation weight calculation, belief degree update, and rule aggregation are the four steps involved in the ER that functions as an inference engine [37]. During input transformation, a given input for an antecedent attribute is transformed into a distribution on the referential values of the antecedent attribute [18]. The matching degrees between an input and the referential values of all antecedents in a rule are determined by input transformation. Then they are processed to calculate an activation weight for the rule. The activation weight of a rule is generated by aggregating the matching degrees to which all antecedents in that rule are activated [17]. Under the conjunctive assumption, the total degree is calculated using a weighted multiplicative aggregation function. All matching degrees are multiplied under the conjunctive assumption since the consequent of a conjunctive belief rule is believed to be true if all antecedent attributes of the rule are activated [17]. On the contrary, the total degree is calculated using an additive aggregation function under the disjunctive assumption. All matching degrees are summed under the disjunctive assumption since the consequent of a disjunctive belief rule is believed to be true if only one antecedent attribute of the rule is activated [42]. When a conjunctive or disjunctive BRB is established, the degree of belief for each referential value of the consequent attribute in all rules is specified. If the input for an antecedent attribute is not available or partially known due to a lack of data, it can cause an incomplete output in each of the rules where the antecedent attribute is used [17]. In such a situation, the original degree of belief embedded with each referential value of the consequent attribute in a rule is updated to address the incompleteness of the rule or address the ignorance of an antecedent attribute while defining the rule. Afterward, the recursive [17] or analytical [43] ER algorithm can be used to aggregate rules and generate the overall combined degree of belief for each referential value of the consequent attribute. Usually, the analytical ER is preferred over the recursive ER due to its computational efficiency [43].

The uncertainty caused by incomplete or missing input of an attribute and ignorance of an attribute when defining a belief rule is addressed during the process of belief degree update, while the uncertainty caused by ambiguous and vague linguistic information and imprecise numeric information are addressed during the rule aggregation procedure, which facilitates the BRB approach to address various types of uncertainty while processing both quantitative and qualitative data in the same framework [44]. The overall procedure of the BRB approach is shown in Fig. 3.

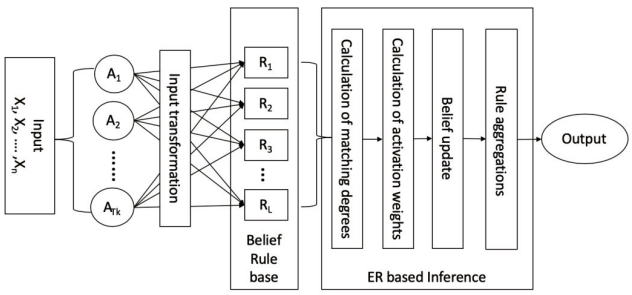



**FIGURE 3.** Belief rule-based approach.

### B. BELIEF RULE-BASED WORKLOAD ORCHESTRATOR

The proposed BRB workload orchestrator is designed based on the Fuzzy Logic-Based (FLB) workload orchestrator [9]. However, there are some similarities and differences between the proposed BRB and FLB workload orchestrators. The similarities are as follows.

(i) The FLB workload orchestrator determines the target server among the nearest (local) edge, neighboring (remote) edge, or global cloud servers. The proposed BRB workload orchestrator also determines the optimal computation unit among those servers for processing each offloaded task from UE.

(ii) The FLB workload orchestrator considers CPU utilization of the nearest (local) edge server, CPU utilization of the least loaded neighboring (remote) edge server, MAN delay, task length, WAN bandwidth, and delay sensitivity of a task as the input parameters. These parameters are also considered as the input parameters for the BRB workload orchestrator since they significantly influence the overall performance of a MEC infrastructure.

(iii) Handling a vast amount of fuzzy rules by considering all the above parameters in a single Fuzzy Logic System (FLS) is a complex operation. Therefore, the FLB workload orchestrator uses two-stage FLS to decrease this complexity. Similarly, handling a large number of belief rules by considering all the above parameters within a single BRB module is a time-consuming and complex operation. Therefore, to decrease the complexity, a two-stage BRB workload orchestrator is considered by employing two BRB modules.

On the contrary, the differences are as follows.

(i) The FLB workload orchestrator uses the same fuzzy rule bases for both delay-sensitive and delay-tolerant applications in the first and second stages. However, since task requirements for both types of applications are different, it is not feasible to use the same fuzzy rule bases to determine the optimal target server for processing tasks generated by delay-sensitive and delay-tolerant applications. This issue is addressed in the BRB workload orchestrator. Since a BRB can be either conjunctive or disjunctive, at first, both types of BRBs are established in the first and second stages of the BRB orchestrator. However, determining whether to use a conjunctive or disjunctive BRB in the first and second stages of the BRB orchestrator is tricky since the BRB orchestrator might perform well for one application when it uses a conjunctive BRB in the first stage and a disjunctive BRB in the second stage, but it might not perform well for another application when it uses the same combination of BRBs. To solve this problem, all possible combinations of conjunctive and disjunctive BRBs in the first and second stages of the BRB orchestrator are categorized based on the delay sensitivity of applications. Then the best combination of BRBs (between conjunctive and disjunctive) for the two-stage BRB workload orchestrator is determined based on the performance comparison of all possible combinations of BRBs, which facilitates the BRB orchestrator to perform equally well for all sorts of applications.

(ii) The Fuzzy Inference System (FIS) in the FLB workload orchestrator can handle uncertainties due to vagueness, imprecision, and ambiguity, but it cannot deal with uncertainties due to ignorance and incompleteness [35]. However, the ER that acts as an inference engine in the BRB workload orchestrator can deal with all sorts of uncertainties while processing both quantitative and qualitative data.

To reiterate, one of the main contributions of this study is determining the best combination of conjunctive and disjunctive BRBs for the two-stage BRB workload orchestrator, which has been done by categorizing all possible combinations of BRBs (between conjunctive and disjunctive) based on the delay sensitivity of applications and conducting a performance comparison among them. Identifying the best combination of BRBs (between conjunctive and disjunctive) for the two-stage BRB workload orchestrator is critical since it facilitates the BRB orchestrator to handle the diverse needs of delay-sensitive and delay-tolerant applications by correctly offloading their tasks into the optimal target server for processing.

### C. TWO-STAGE BRB WORKLOAD ORCHESTRATOR

In this study, a two-stage BRB workload orchestrator is designed by utilizing two BRB modules, where the first BRB module (BRBM1) in the first stage determines the candidate edge server between the local edge server and neighboring/remote edge servers in the edge tier based on CPU utilization of the local edge server, CPU utilization of the least loaded remote edge server, and MAN delay, while the second BRB module (BRBM2) in the second stage determines the optimal target server between the candidate edge server and the global cloud server based on CPU utilization of the candidate edge server, task length, WAN bandwidth, and delay sensitivity of a task, as shown in Fig. 4.

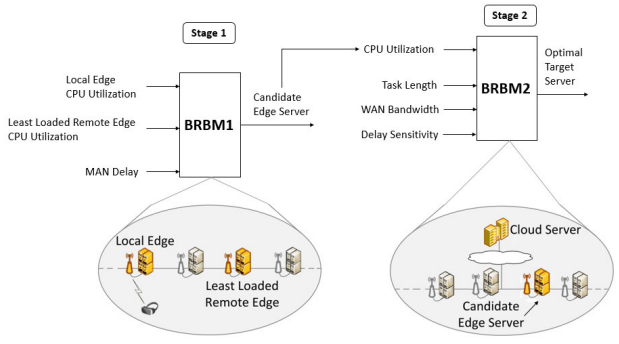



**FIGURE 4.** Two-stage belief rule-based workload orchestrator.

1) FIRST STAGE

Following [9], three input variables, namely CPU Utilization of the Local Edge Server, CPU Utilization of the Least Loaded Remote Edge Server, MAN Delay, and one output variable named Candidate Edge Server, are considered in the first stage.

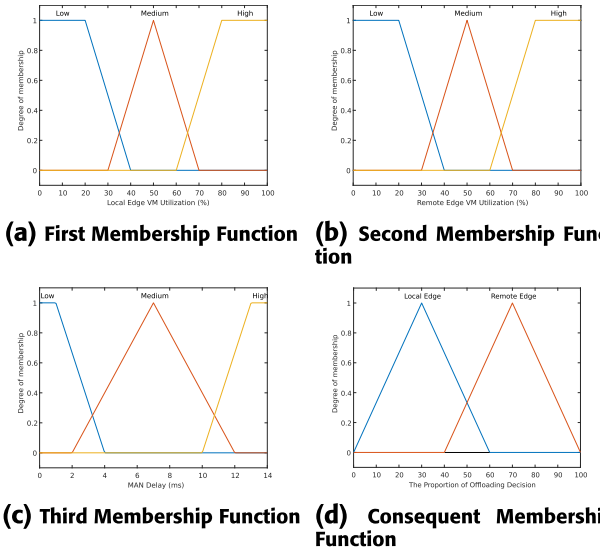
(i) Local Edge CPU Utilization: Offloading a task from UE to the nearest (local) edge server is generally advantageous to refrain from consuming MAN resources. However, if end-users at distinct locations are not evenly distributed, offloading a task to the nearest (local) edge server is not rational. Since the computation capacity of an edge server is modest in comparison to a cloud server, constantly offloading tasks to the nearest (local) edge server results in a lot of task failures due to insufficient computational resources, especially in hotspot locations. Hence, distributing offloaded tasks among the neighboring/remote edge servers can improve the performance of a MEC infrastructure if MAN is not congested. The state of computational resources in a local edge server is determined by the CPU utilization of that edge server. It is assumed that a local edge server runs multiple VMs. Hence, Local Edge CPU Utilization stands for the average utilization of all VMs running on that local edge server [9].

(ii) Least Loaded Remote Edge CPU Utilization: Similar to a nearest (local) edge server, the state of computational resources in a remote edge server is determined by the CPU utilization of that edge server. It is assumed that offloading a task to any neighboring/remote edge server has the same MAN usage cost [9]. Hence, CPU utilization of the least loaded remote edge server is considered. Offloading a task to a neighboring/remote edge server is desirable if the CPU utilization of the least loaded neighboring/remote edge server is low in comparison to the nearest (local) edge server and the MAN delay is minimal. Analogous to a local edge server, a neighboring/remote edge server runs multiple VMs. Therefore, the CPU utilization of a remote edge server stands for the average utilization of all VMs running on that remote edge server [9].

(iii) MAN Delay: MAN delay becomes a significant bottleneck in a MEC infrastructure when a lot of end-users use MAN concurrently. Hence, MAN delay is taken into account when selecting the best edge server. MAN is likely to be congested if the MAN delay is longer than desired. In that case, a task should be offloaded to a local edge server.

Fig. 5 shows the membership functions for each input and output variable considered in the first stage of the Fuzzy Logic-Based (FLB) workload orchestrator in [9], which were determined empirically by testing various membership functions.

Each input and output variable, their linguistic terms, and degree of membership values for each linguistic term used in




**(a) First Membership Function** **(b) Second Membership Function**

**(c) Third Membership Function** **(d) Consequent Membership Function**

**FIGURE 5. Membership functions used in the first stage of the FLB workload orchestrator [9].**

the first stage of the FLB workload orchestrator is mapped for the first BRB module (BRBM1) used in the first stage of the proposed BRB workload orchestrator. For BRBM1, the three input variables are the three antecedent attributes, the output variable is the consequent attribute, linguistic terms for each input and output variable are referential values for the antecedent and consequent attributes. The utility value for each referential value of the antecedent and consequent attributes is determined by taking the middle value from the degree of membership of each linguistic term that was used in the first stage of the FLB workload orchestrator. For example, the range of the degree of membership of the first linguistic term ('Low') associated with the first input variable ('Local Edge CPU Utilization') in the first stage of the FLB orchestrator was $[0 - 40]$. Hence, for BRBM1, the utility value for the third referential value ('Low') of the first antecedent attribute ('Local Edge CPU Utilization') in the first stage of the BRB orchestrator is 20. In a similar way, the utility value for each referential value of the antecedent and consequent attributes is determined, as shown in Table 2.

**TABLE 2. Details of antecedent and consequent attributes for BRBM1.**

| | X1 | | | X2 | | | X3 | | |
|---|---|---|---|---|---|---|---|---|---|
| **Referential Values** | H | M | L | H | M | L | H | M | L |
| **Utility Values** | 80 | 50 | 20 | 80 | 50 | 20 | .013 | .007 | .001 |

**(a) Antecedent Attributes (X1: Local Edge CPU Utilization, X2: Least Loaded Remote Edge CPU Utilization, X3: MAN Delay; H: High, M: Medium, L: Low)**

| | Y1 | |
|---|---|---|
| **Referential Values** | Remote Edge | Local Edge |
| **Utility Values** | 70 | 30 |

**(b) Consequent Attribute (Y1: Candidate Edge Server)**

Since the first stage of the FLB workload orchestrator had three input variables and each input variable had three linguistic terms, the number of fuzzy rules in the fuzzy rule

base for the first stage was 27 ($3 \times 3 \times 3$), where each rule was determined empirically by testing various combinations of fuzzy rules [9]. For simplicity, only the first three and last three rules from the fuzzy rule base is shown in Table 3, where X1 (Local Edge CPU Utilization), X2 (Least Loaded Remote Edge CPU Utilization), X3 (MAN Delay) are the three input variables and Y1 (Candidate Edge Server) is the output variable.

**TABLE 3. Fuzzy rule base used in the first stage of the FLB workload orchestrator [9].**

| Rule Index | Rule Weight | IF | | | THEN |
|---|---|---|---|---|---|
| | | X1 | X2 | X3 | Y1 |
| 1 | 1 | High | High | High | Local Edge |
| 2 | 1 | High | High | Medium | Remote Edge |
| 3 | 1 | High | High | Low | Remote Edge |
| ... | ... | ... | ... | ... | ... |
| 25 | 1 | Low | Low | High | Local Edge |
| 26 | 1 | Low | Low | Medium | Local Edge |
| 27 | 1 | Low | Low | Low | Remote Edge |

Now for BRBM1, a conjunctive BRB is established by transforming the fuzzy rule base used in the first stage of the FLB workload orchestrator [9]. As discussed above, BRBM1 has three antecedent attributes and each of them has three referential values. So by considering all possible combinations of referential values of antecedent attributes under the conjunctive assumption, the total number of belief rules in the conjunctive BRB is 27 ($3 \times 3 \times 3$). The only difference between fuzzy rule base and BRB lies in the fact that belief degree is embedded with each referential value of the consequent attribute in a BRB. For simplicity, only the first three and last three rules from the conjunctive BRB are shown in Table 4, where X1 (Local Edge CPU Utilization), X2 (Least Loaded Remote Edge CPU Utilization), X3 (MAN Delay) are the three antecedent attributes and Y1 (Candidate Edge Server) is the consequent attribute.

**TABLE 4. Conjunctive belief rule base for BRBM1.**

| Rule ID | Rule Weight | IF | | | THEN (Y1) | |
|---|---|---|---|---|---|---|
| | | X1 | X2 | X3 | Remote Edge | Local Edge |
| 1 | 1 | High | High | High | 0 | 1 |
| 2 | 1 | High | High | Medium | 1 | 0 |
| 3 | 1 | High | High | Low | 1 | 0 |
| ... | ... | ... | ... | ... | ... | ... |
| 25 | 1 | Low | Low | High | 0 | 1 |
| 26 | 1 | Low | Low | Medium | 0 | 1 |
| 27 | 1 | Low | Low | Low | 1 | 0 |

The first belief rule taken from Table 4 can be expressed as follows.

$$R_1 : \begin{cases} \text{IF Local Edge CPU Utilization is High} \\ \text{AND Least Loaded Remote Edge CPU Utilization is} \\ \text{High AND MAN Delay is High THEN Candidate} \\ \text{Edge Server is } \{(\text{Remote Edge}, 0), (\text{Local Edge}, 1)\} \end{cases}$$

In the above belief rule, "Local Edge CPU Utilization", "Least Loaded Remote Edge CPU Utilization", and "MAN Delay" are the antecedent attributes, while 'High', 'High', and 'High' are their corresponding referential values. "Candidate Edge Server" is the consequent attribute and its referential values are 'Remote Edge' and 'Local Edge'. The belief distribution of this consequent attribute is (Remote Edge, 0) and (Local Edge, 1). Here, 0 and 1 are the degree of belief to which the consequent referential value 'Remote Edge' and 'Local Edge' are believed to be true. This rule is considered complete because the summation of belief degrees associated with each referential value of the consequent attribute is 1 ($0 + 1 = 1$). In the first stage, full computation offloading is considered, which means a whole task from UE is offloaded to a Local Edge or Remote Edge server for processing without any split of the task. Hence, belief degrees embedded with referential values ('Remote Edge' and 'Local Edge') of the consequent attribute ("Candidate Edge Server") in all conjunctive belief rules are considered binary values (either 0 or 1).

In this study, a disjunctive BRB is also considered to determine whether a conjunctive or disjunctive BRB will be most appropriate for BRBM1 in the first stage. Since there was no disjunctive fuzzy rule base for the FLB workload orchestrator in the first stage, the disjunctive BRB for BRBM1 are constructed based on empirical analysis by testing various combinations of disjunctive belief rules and choosing the best combination among them. As discussed in subsection III-A, the prerequisite of the disjunctive assumption is all antecedent attributes must contain the same number of referential values. For BRBM1, it has three antecedent attributes and all antecedent attributes contain the same number of referential values, which is 3. Hence, the total number of belief rules in the disjunctive BRB for BRBM1 is 3. The disjunctive BRB is shown in Table 5, where X1 (Local Edge CPU Utilization), X2 (Least Loaded Remote Edge CPU Utilization), X3 (MAN Delay) are the three antecedent attributes and Y1 (Candidate Edge Server) is the consequent attribute.

**TABLE 5. Disjunctive belief rule base for BRBM1.**

| Rule ID | Rule Weight | IF | | | THEN (Y1) | |
|---|---|---|---|---|---|---|
| | | X1 | X2 | X3 | Remote Edge | Local Edge |
| 1 | 1 | High | High | High | 1 | 0 |
| 2 | 1 | Medium | Medium | Medium | 1 | 0 |
| 3 | 1 | Low | Low | Low | 0 | 1 |

The first stage of the FLB workload orchestrator consisted of three main steps, namely fuzzification, fuzzy inference, and defuzzification. After defuzzification, the output of the first stage became a crisp value between 0 and 100. If the output was less than or equal to 50, the Local Edge Server was chosen as the Candidate Edge Server. Otherwise, the least loaded Remote Edge Server was chosen as the Candidate Edge Server [9].

For BRBM1, ER acts as an inference engine that comprises four steps, as discussed in subsection III-A. The fourth step determines the final outcome for the consequent attribute in the first stage. If the outcome is less than or equal to 50, the Local Edge Server will be chosen as the Candidate Edge Server. Otherwise, the least loaded Remote Edge Server will be chosen as the Candidate Edge Server after the first stage of the BRB workload orchestrator.

### 2) SECOND STAGE

Following [9], four input variables, namely CPU Utilization of the Candidate Edge Server, Task Length, WAN Bandwidth, Delay Sensitivity and one output variable named Optimal Target Server, are considered in the second stage.

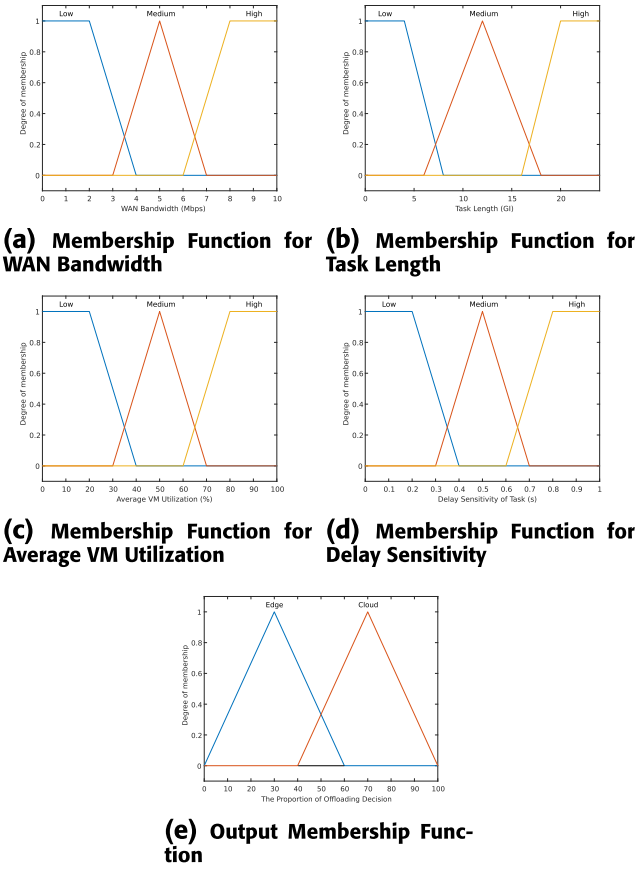
(i) Candidate Edge CPU Utilization: The CPU utilization of the candidate edge server supplies information on the residual processing capacity. If the CPU utilization of the candidate edge server exceeds a certain threshold, it can be considered congested. In this instance, offloading to a global cloud server could be viable based on the WAN circumstances and the task's delay sensitivity. As mentioned before, it is assumed that each edge server runs multiple VMs. Hence, the CPU utilization of the candidate edge server stands for the average utilization of all VMs running on that edge server [9].

(ii) Task Length: In this study, the number of instructions indicates the length of a task [9]. The length of a task determines the execution time of that task on an edge/cloud server in relation to the processing power of the Virtual Machine (VM) running on that edge/cloud server. Hence, the length of a task is taken into account to decide the optimal target server for the corresponding task. A complicated task with a large number of instructions should be offloaded to a cloud server, while an uncomplicated task can be offloaded to an edge server.

(iii) WAN Bandwidth: Before offloading a task to the cloud server, WAN Bandwidth is a crucial criterion that needs to be taken into account. Offloading a task to the cloud server is not beneficial if WAN latency is higher than an application's latency requirement or if WAN congestion is severe enough to trigger packet losses.

(iv) Delay Sensitivity: The delay sensitivity of a task indicates the task's tolerance for taking a long time to complete due to network conditions or CPU utilization of a server. It is an important criterion to consider for providing a good QoS by a MEC infrastructure. It is assumed that an application profile defines the delay sensitivity. In real-life scenarios, a network administrator can define the delay sensitivity of an application, or an application developer can incorporate the delay sensitivity into a particular application [9].

Fig. 6 shows the membership functions for each input and output variable considered in the second stage of the FLB workload orchestrator in [9], which were determined empirically by testing various membership functions.




**(a) Membership Function for WAN Bandwidth** **(b) Membership Function for Task Length**

**(c) Membership Function for Average VM Utilization** **(d) Membership Function for Delay Sensitivity**

**(e) Output Membership Function**

**FIGURE 6. Membership functions used in the second stage of the FLB workload orchestrator [9].**

Each input and output variable, their linguistic terms, and degree of membership values for each linguistic term used in the second stage of the FLB workload orchestrator is mapped for the second BRB module (BRBM2) used in the second stage of the proposed BRB workload orchestrator. For BRBM2, the four input variables are the four antecedent attributes, the output variable is the consequent attribute, linguistic terms for each input and output variable are referential values for the antecedent and consequent attributes. Similar to BRBM1, the utility value for each referential value of the antecedent and consequent attributes is determined by taking the middle value from the degree of membership of each linguistic term that was used in the second stage of the FLB workload orchestrator. For example, the range of the degree of membership of the first linguistic term ('Low') associated with the first input variable ('WAN Bandwidth') in the second stage of the FLB orchestrator was [0−4]. Hence, for BRBM2, the utility value for the third referential value ('Low') of the third antecedent attribute ('WAN Bandwidth') in the second stage of the BRB orchestrator is 2. In a similar way, the utility value for each referential value of the antecedent and consequent attributes is determined, as shown in Table 6.

**TABLE 6.** Details of antecedent and consequent attributes for BRBM2.

| | X4 | | | X5 | | | X6 | | |
|---|---|---|---|---|---|---|---|---|---|
| **Referential Values** | H | M | L | H | M | L | H | M | L |
| **Utility Values** | 80 | 50 | 20 | 20000 | 12000 | 4000 | 8 | 5 | 2 |

**(a) Antecedent Attributes (X4: Candidate Edge CPU Utilization, X5: Task Length, X6: WAN Bandwidth; H: High, M: Medium, L: Low)**

| | X7 | | | Y2 | |
|---|---|---|---|---|---|
| **Referential Values** | High | Medium | Low | Cloud | Candidate Edge |
| **Utility Values** | 0.8 | 0.5 | 0.2 | 70 | 30 |

**(b) Antecedent Attributes (X7: Delay Sensitivity), Consequent Attribute (Y2: Optimal Target Server)**

**TABLE 7.** Fuzzy rule base used in the second stage of the FLB workload orchestrator [9].

| Rule Index | Rule Weight | IF | | | | THEN |
|---|---|---|---|---|---|---|
| | | X4 | X5 | X6 | X7 | Y2 |
| 1 | 1 | High | High | High | High | Cloud |
| 2 | 1 | High | High | High | Medium | Cloud |
| 3 | 1 | High | High | High | Low | Cloud |
| ... | ... | ... | ... | ... | ... | ... |
| 79 | 1 | Low | Low | Low | High | Candidate Edge |
| 80 | 1 | Low | Low | Low | Medium | Candidate Edge |
| 81 | 1 | Low | Low | Low | Low | Candidate Edge |

Since the second stage of the FLB workload orchestrator had four input variables and each input variable had three linguistic terms, the number of fuzzy rules in the fuzzy rule base for the second stage was 81 ($3 \times 3 \times 3 \times 3$), where each rule was determined empirically by testing various combinations of fuzzy rules [9]. For simplicity, only the first three and last three rules from the fuzzy rule base is shown in Table 7, where X4 (Candidate Edge CPU Utilization), X5 (Task Length), X6 (WAN Bandwidth), X7 (Delay Sensitivity) are the four input variables and Y2 (Optimal Target Server) is the output variable.

Now for BRBM2, a conjunctive BRB is established by transforming the fuzzy rule base used in the second stage of the FLB workload orchestrator [9]. As discussed above, BRBM2 has four antecedent attributes and each of them has three referential values. So by considering all possible combinations of referential values of antecedent attributes under the conjunctive assumption, the total number of belief rules in the conjunctive BRB is 81 ($3 \times 3 \times 3 \times 3$). As mentioned previously, the only difference between fuzzy rule base and BRB lies in the fact that belief degree is embedded with each referential value of the consequent attribute in a BRB. For simplicity, only the first three and last three rules from the conjunctive BRB is shown in Table 8, where X4 (Candidate Edge CPU Utilization), X5 (Task Length), X6 (WAN Bandwidth), X7 (Delay Sensitivity) are the four antecedent attributes and Y2 (Optimal Target Server) is the consequent attribute.

**TABLE 8.** Conjunctive belief rule base for BRBM2.

| Rule ID | Rule Weight | IF | | | | THEN (Y2) | |
|---|---|---|---|---|---|---|---|
| | | X4 | X5 | X6 | X7 | Cloud | Candidate Edge |
| 1 | 1 | High | High | High | High | 1 | 0 |
| 2 | 1 | High | High | High | Medium | 1 | 0 |
| 3 | 1 | High | High | High | Low | 1 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 79 | 1 | Low | Low | Low | High | 0 | 1 |
| 80 | 1 | Low | Low | Low | Medium | 0 | 1 |
| 81 | 1 | Low | Low | Low | Low | 0 | 1 |

The first belief rule taken from Table 8 can be expressed as follows.

$$R_1: \begin{cases} \text{IF Candidate Edge CPU Utilization is High AND} \\ \text{Task Length is High AND WAN Bandwidth is High} \\ \text{AND Delay Sensitivity is High THEN Optimal} \\ \text{Target Server is } \{(\text{Cloud}, 1), (\text{Candidate Edge}, 0)\} \end{cases}$$

In the above belief rule, ''Candidate Edge CPU Utilization'', ''Task Length'', ''WAN Bandwidth'', and ''Delay Sensitivity'' are the antecedent attributes, while 'High', 'High', 'High', and 'High' are their corresponding referential values. ''Optimal Target Server'' is the consequent attribute and its referential values are 'Cloud' and 'Candidate Edge'. The belief distribution of this consequent attribute is (Cloud, 1) and (Candidate Edge, 0). Here, 1 and 0 are the degree of belief to which the consequent referential value 'Cloud' and 'Candidate Edge' are believed to be true. This rule is considered complete because the summation of belief degrees associated with each referential values of the consequent attribute is 1 ($1 + 0 = 1$). Similar to the first stage, full computation offloading is considered in the second stage, which means a whole task from UE is offloaded to a Candidate Edge or Cloud server for processing without any split of the task. Hence, belief degrees embedded with referential values ('Cloud' and 'Candidate Edge') of the consequent attribute (''Optimal Target Server'') in all conjunctive belief rules are considered binary values (either 0 or 1).

**TABLE 9.** Disjunctive belief rule base for BRBM2 (H: High, M: Medium, L: Low).

| Rule ID | Rule Weight | IF | | | | THEN (Y2) | |
|---|---|---|---|---|---|---|---|
| | | X4 | X5 | X6 | X7 | Cloud | Candidate Edge |
| 1 | 1 | H | H | H | H | 1 | 0 |
| 2 | 1 | M | M | M | M | 0 | 1 |
| 3 | 1 | L | L | L | L | 0 | 1 |

Similar to BRBM1, a disjunctive BRB is also considered to determine whether a conjunctive or disjunctive BRB will be most appropriate for BRBM2 in the second stage. Since there was no disjunctive fuzzy rule base for the FLB workload orchestrator in the second stage, the disjunctive BRB for BRBM2 are constructed based on empirical analysis by testing various combinations of disjunctive belief rules and

choosing the best combination among them. As discussed in subsection III-A, the prerequisite of the disjunctive assumption is all antecedent attributes must contain the same number of referential values. For BRBM2, it has four antecedent attributes and all antecedent attributes contain the same number of referential values, which is 3. Hence, the total number of belief rules in the disjunctive BRB for BRBM2 is 3. The disjunctive BRB is shown in Table 9, where X4 (Candidate Edge CPU Utilization), X5 (Task Length), X6 (WAN Bandwidth), X7 (Delay Sensitivity) are the four antecedent attributes and Y2 (Optimal Target Server) is the consequent attribute.

Similar to the first stage, the second stage of the FLB workload orchestrator consisted of three main steps, namely fuzzification, fuzzy inference, and defuzzification. After defuzzification, the output of the second stage became a crisp value between 0 and 100. If the output was less than or equal to 50, the Candidate Edge Server was chosen as the Optimal Target Server. Otherwise, the Cloud Server was chosen as the Optimal Target Server [9].

Similar to BRBM1, ER acts as an inference engine for BRBM2 and the fourth step of ER determines the final outcome for the consequent attribute in the second stage. If the outcome is less than or equal to 50, the Candidate Edge Server will be chosen as the Optimal Target Server. Otherwise, the Cloud Server will be chosen as the Optimal Target Server after the second stage of the BRB workload orchestrator.

Algorithm 1 presents the proposed two-stage BRB workload orchestrator algorithm.

## IV. SIMULATED ENVIRONMENT

A university campus scenario presented in [9] is simulated in this study. In this scenario, students and other end-users request services while moving to different locations within the campus, such as classroom premises, libraries, cafeterias, student hostels, and administrative offices. End-users carry different types of UE, such as smartphones, smartwatches, and smart glasses, which run different types of applications and offload tasks generated by these applications that result in diverse data traffic. Several edge servers interconnected through MAN are located on the campus to provide services. It is assumed that MAN is a Gigabit Ethernet of campus LAN [9]. A dedicated Wi-Fi access point covers each location. When end-users move to the coverage area of the access point, they join the related WLAN [45]. End-users offload tasks from UE to the nearest (local) edge server via WLAN. The MAN connection is used if a task is offloaded to a neighboring (remote) edge server, while the WAN connection (broadband connection), provided by the Wi-Fi access point, is used if a task is offloaded to the cloud server [9], [45].

To summarize, a task can be offloaded from UE to the nearest (local) edge server (in this case, UE accesses the server with one hop over WLAN), a neighboring (remote) edge server (in this case, UE accesses the server with two hops over WLAN and MAN), and the global cloud server (in this case, UE accesses the server with three hops over WLAN, MAN, and WAN) [9]. This task offloading decision is handled by the proposed two-stage BRB workload orchestrator.

In order to imitate the mobility of end-users realistically, different locations on the campus are presumed to have a distinct level of attractiveness, which affect the dwell time that an end-user spends in a related location. For instance, students usually stay more on classroom premises in the morning, while they gather in libraries and cafeterias in the afternoon and evening and spend more time in student hostels at night. The end-user density, hence task offloading requests (data traffic) at each location, differs based on the level of attractiveness of that location. In this scenario, three location categories have been considered with distinct attractiveness levels, which are depicted in Fig. 7.

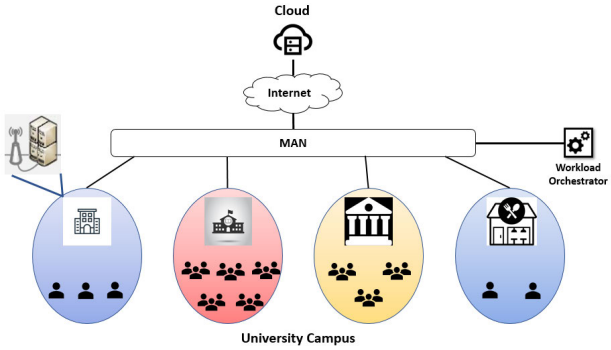



**FIGURE 7.** A university campus scenario with three location categories [9].

In Fig. 7, a Type 1 location, shown with a red ellipse, has the highest attractiveness level and a Type 2 location, shown using an orange ellipse, has the medium attractiveness level, while two Type 3 locations, shown with two blue ellipses, have the lowest attractiveness level. The end-user density affects the overall performance of a MEC infrastructure because of its direct relation with task offloading requests (data traffic). For instance, due to high end-user density in a Type 1 location, both network and computational resources can be congested in that location. The proposed BRB workload orchestrator deals with this issue by optimally distributing offloaded tasks from UE among local edge, neighboring/remote edge, and global cloud servers.

### A. SIMULATION SETUP

In this study, EdgeCloudSim is chosen as the simulator because it enables to modeling of a realistic MEC environment and supports the simulation of the multi-tier MEC infrastructure where several edge servers run in cooperation with global cloud servers [45]. It supports a range of crucial functions such as network modeling specific to WLAN, MAN, and WAN, UE mobility model, realistic and tunable load generator, computational resource modeling such as VM creation with a specified capacity, and orchestration actions modeling in MEC scenarios with an edge orchestrator module [46].

**Algorithm 1** Two-Stage Belief Rule-Based Workload Orchestrator Algorithm

**Input:** Offloaded Task, $T$
**Output:** Optimal Target Server, $O_{server}$

Read Local Edge CPU Utilization: $U_{LocalEdge}$
Read Least Loaded Remote Edge CPU Utilization: $U_{RemoteEdge}$
Read MAN Delay: $D_{MAN}$
$C_1 = \text{BRBM1}(U_{LocalEdge}, U_{RemoteEdge}, D_{MAN})$
**if** $C_1 \leq 50$ **then**
    Candidate Edge = Local Edge
    $U_{CandidateEdge} = U_{LocalEdge}$ `// Candidate Edge CPU Utilization`
**else**
    Candidate Edge = Remote Edge
    $U_{CandidateEdge} = U_{RemoteEdge}$ `// Candidate Edge CPU Utilization`
Read Length of the offloaded task: $T_{Length}$
Read WAN Bandwidth: $B_{WAN}$
Read Delay Sensitivity of the offloaded task: $T_{DelaySensitivity}$
$C_2 = \text{BRBM2}(U_{CandidateEdge}, T_{Length}, B_{WAN}, T_{DelaySensitivity})$
**if** $C_2 \leq 50$ **then**
    $O_{server}$ = Candidate Edge
**else**
    $O_{server}$ = Cloud
**return** $O_{server}$

**TABLE 10.** Types of applications used in simulation experiments [9].

| Applications | Usage percentage (%) | Task Interarrival (sec) | Delay Sensitivity | Active/Idle Period (sec) | Upload/Download Data (KB) | Task Length (GI) | VM Utilization on Edge (%) | VM Utilization on Cloud (%) |
|---|---|---|---|---|---|---|---|---|
| **AR** | 30 | 2 | 0.9 | 40/20 | 1500/25 | 9 | 6 | 0.6 |
| **Health** | 20 | 3 | 0.7 | 45/90 | 20/1250 | 3 | 2 | 0.2 |
| **Infotainment** | 30 | 7 | 0.3 | 30/45 | 25/1000 | 15 | 10 | 1 |
| **Compute-Intensive** | 20 | 20 | 0.1 | 60/120 | 2500/200 | 45 | 30 | 3 |

In MEC, UE can run several applications or services whose characteristics may vary according to different requirements. A task offloading request to an edge or cloud server can generate separate CPU and network loads on a MEC infrastructure. For example, a compute-intensive task may require vast CPU resources, but that task might not use excessive network capacity. On the contrary, a data backup application may require a small number of CPU resources, but the data backup process might need a significant amount of data transmission. In order to simulate real-life scenarios, four types of applications found in the literature are considered in this study. The first one is an Augmented Reality (AR) application on Google Glass presented in [47], the second one is a health application presented in [48], the third one is an infotainment application discussed in [49], and the fourth one is a compute-intensive application discussed in [50]. In simulation experiments, it is assumed that end-users use these four different applications with various types of UE that offload tasks belonging to these application categories, and an edge or cloud server provides corresponding services. For instance, an end-user uses the AR application by wearing smart glass and it offloads captured pictures to an edge server that provides face recognition service. Similarly, a user uses the health application by carrying a foot-mounted inertial sensor and it offloads sensor data to an edge/cloud server that provides a fall risk detection service. In the same way, end-users use infotainment and compute-intensive applications with various UE, which offload tasks to either cloud or edge servers that provide corresponding services. The properties of the four types of applications used in simulation experiments are shown in Table 10.

In Table 10, the usage percentage of an application represents the portion of UE running this application. For instance, 30% of UE runs the AR application, while 20% of UE runs the health application. Task interarrival time specifies how often UE sends a related task of an application to the workload orchestrator. For instance, every 7 seconds, UE sends a related task of the infotainment application

to the workload orchestrator. For each application, task interarrival time is exponentially distributed. The delay sensitivity specifies whether an application is delay-sensitive or not. If this value is high ($\geq$0.5), an application is delay-sensitive, otherwise, it is delay-tolerant. For instance, the delay sensitivity of the AR application is 0.9, so it is a delay-sensitive application. On the contrary, the delay sensitivity of the infotainment application is 0.3, so it is a delay-tolerant application. In real-life scenarios, UE does not generate service requests continuously since an application depicts an intermittent behavior where it actively generates tasks for a period and remains idle between active periods. For instance, an application may log data for a period of time from a sensor before sending that data to an edge/cloud server for additional processing. This pattern is modeled using the active/idle period for an application such that the application creates tasks during the active period and does nothing in the idle period. For instance, the AR application generates tasks for 40 seconds and stays idle for 20 seconds.

An application generates tasks, which contain random lengths with regard to the number of instructions and random input/output file sizes to upload/download [45]. For instance, an AR application may need high CPU computation, a large input file to upload, and a small output file to download, while an infotainment application may need low CPU power in comparison to the AR application due to the fact that it sends requests with a small input to a server and the server returns a large output as a response [45]. Hence, upload/download data for an application depicts the corresponding amount of data sent to/received from a server. For example, upload/download data for the AR application is considered ⟨1500 KB, 25 KB ⟩ to indicate the fact that a typical image is uploaded to a server and the server returns text metadata of a recognized person as a response. Task length deduces the needed CPU resource for a task in the GI (billion instructions) unit. For instance, it is assumed that an average of 9 billion instructions (GI) is required for the AR application. The length of a task determines the execution time of that task on an edge/cloud server in relation to the processing power of the Virtual Machine (VM) running on that edge/cloud server. The unit of the processing speed of a VM on an edge/cloud server is GIPS (Giga Instructions Per Second) [9]. In simulation experiments, it is assumed for all scenarios that the cloud server has adequate computational resources while edge servers have limited computational resources. Therefore, the cloud server executes each offloaded task on the cloud VMs without blocking, while congestion may take place in edge servers due to limited computational resources. For each application, task length is exponentially distributed. VM utilization on Edge and VM utilization on Cloud parameters are used to limit the number of concurrent tasks that can run on an edge VM or cloud VM. In simulation experiments, all VMs are empowered in terms of CPU resources. Hence, a task offloaded to an edge server will fail if there is no sufficient CPU resource on that edge VM to process that task. For example, VM Utilization on Edge for the infotainment application is 10%, which means up to 10 tasks can be executed simultaneously on an Edge VM for the infotainment application.

**TABLE 11. Simulation parameters [9].**

| Parameter | Value |
|---|---|
| Number of Repetitions | 30 |
| Number of Edge/Cloud Server | 14/1 |
| Number of VMs per Edge/Cloud Server | 8/4 |
| Number of Cores per Edge/Cloud VM CPU | 2/4 |
| VM CPU Speed per Edge/Cloud | 10/100 GIPS |
| VM RAM per Edge/Cloud | 2/32 GB |
| VM Storage per Edge/Cloud | 50/1000 GB |
| Number of UE | 200 to 2400 |
| WAN/WLAN Bandwidth | Empirical [9] |
| MAN Delay | MMPP/M/1 model |
| LAN Propagation Delay | 5 milliseconds (ms) |
| WAN Propagation Delay | 100 ms |
| User Mobility Model | Nomadic Mobility Model |
| Probability of Selecting a Location Type | Equal |
| Number of Locations Type 1/2/3 | 8/4/2 |
| Mean Dwell Time in Type 1/2/3 | 8/5/2 minutes |

Other important simulation parameters are stated in Table 11. In this study, all simulation experiments are repeated 30 times. 1 cloud server with 4 VMs and 14 edge servers with 8 VMs on each edge server are considered in all simulations. Each edge server has an equal capacity and all 14 edge servers operate in 14 locations. Each VM in an edge server has 2 core CPU with 10 GIPS capacity, 2 GB RAM, and 50 GB storage, while each VM in the cloud server has 4 core CPU with 100 GIPS capacity, 32 GB RAM, and 1000 GB storage for processing each offloaded task from UE.

The performance of a system should be analyzed under different load. The load can be increased in various ways. For example, by using a fixed amount of devices, a system can be overloaded by increasing requests generated by those devices. Another approach for making a system overloaded is to increase the number of devices while maintaining the service requests constant. The latter approach is followed in this study. In order to simulate various workloads, the minimum and maximum number of UE is considered 200 and 2400, respectively [9]. Each simulation starts with the minimum number of UE and the number of UE is increased by 200 until it reaches the maximum number of UE.

For network delay modeling, the study conducted by [9] is followed. The authors used the result of an empirical study for WLAN and WAN delays. They used an 802.11 family access point for WLAN delay observation and utilized a fiber internet connection in Istanbul to observe WAN delays [45]. They used the average bandwidth of ten consecutive experiments for WLAN bandwidth and determined WAN bandwidth by taking the average values of measurements at different times of the day for a week. They observed MAN delay via an MMPP/M/1 queueing model (a single server queue model with Markov-modulated Poisson process

**TABLE 12.** Possible combinations of BRBs for the Two-stage BRB workload orchestrator.

| Combinations | BRBM1 | BRBM2 |
|---|---|---|
| BRB-comb-1 | Conjunctive | Conjunctive |
| BRB-comb-2 | Conjunctive | Disjunctive |
| BRB-comb-3 | Disjunctive | Conjunctive |
| BRB-comb-4 | Disjunctive | Disjunctive |

**(a) For all Applicatons (AR, Health, Infotainment, and Compute-Intensive)**

| *Application type* | *Delay-Sensitive* | | *Delay-Tolerant* | |
|---|---|---|---|---|
| **Combinations** | **BRBM1** | **BRBM2** | **BRBM1** | **BRBM2** |
| BRB-comb-5 | Conjunctive | Conjunctive | Conjunctive | Disjunctive |
| BRB-comb-6 | Conjunctive | Conjunctive | Disjunctive | Conjunctive |
| BRB-comb-7 | Conjunctive | Conjunctive | Disjunctive | Disjunctive |
| BRB-comb-8 | Conjunctive | Disjunctive | Conjunctive | Conjunctive |
| BRB-comb-9 | Conjunctive | Disjunctive | Disjunctive | Conjunctive |
| BRB-comb-10 | Conjunctive | Disjunctive | Disjunctive | Disjunctive |
| BRB-comb-11 | Disjunctive | Conjunctive | Conjunctive | Conjunctive |
| BRB-comb-12 | Disjunctive | Conjunctive | Conjunctive | Disjunctive |
| BRB-comb-13 | Disjunctive | Conjunctive | Disjunctive | Disjunctive |
| BRB-comb-14 | Disjunctive | Disjunctive | Conjunctive | Conjunctive |
| BRB-comb-15 | Disjunctive | Disjunctive | Conjunctive | Disjunctive |
| BRB-comb-16 | Disjunctive | Disjunctive | Disjunctive | Conjunctive |

**(b) For Delay-Sensitive (AR, Health) and Delay-Tolerant (Infotainment, and Compute-Intensive) Applicatons**

arrivals), which updates the mean arrival rate of tasks when the system congestion level changes. They also considered LAN propagation delay of 5 ms and WAN propagation delay of 100 ms. The details of network delay modeling can be found in [9] and [45].

It is assumed that students and other end-users move from one location to another location within the campus following a nomadic mobility model [51], which dictates when a user moves to a location, the user waits/stays there for a random amount of time (dwell time) and then moves from that location to another location. So the location of a user changes after a specified amount of time. As discussed in Section IV, the university campus scenario has three location categories with distinct levels of attractiveness. Students and other end-users stay in those locations for a random dwell time, which is proportional to the attractiveness level. An end-user will spend more time in a location if the level of attractiveness of that location is high. The likelihood of moving to any location is considered equal for all locations. It is presumed that there are eight locations of Type 1 with the highest attractiveness level, four locations of Type 2 with medium attractiveness level, and two locations of Type 3 with the lowest attractiveness level, while the waiting duration (mean dwell time) in locations of Type 1, 2, and 3 are 8, 5, and 2 minutes, respectively.

### B. COMBINATIONS OF BRBS FOR THE TWO-STAGE BRB ORCHESTRATOR

As discussed in section III-B, both conjunctive and disjunctive BRBs are established in the first and second stages of the proposed BRB orchestrator to determine the most appropriate BRBs for BRBM1 (in the first stage) and BRBM2 (in the second stage). However, determining whether to use a conjunctive or disjunctive BRB in the first and second stages of the BRB orchestrator is tricky since the BRB orchestrator might perform well for one application when it uses a conjunctive BRB in the first stage and a disjunctive BRB in the second stage, but it might not perform well for another application when it uses the same combination of BRBs. To address this issue, all possible combinations of conjunctive and disjunctive BRBs in the first and second stages of the BRB orchestrator are categorized based on the delay sensitivity of applications so that the best combination of BRBs for all sorts of applications can be identified and the proposed BRB orchestrator can perform equally well for all types of applications. As discussed in subsection IV-A, four types of applications are considered in this study. Two of them are delay-sensitive applications (AR and Health), while the other two are delay-tolerant applications (Infotainment and Compute-Intensive). At first, four possible combinations of BRBs are obtained for all types of applications. After that, the applications are divided into delay-sensitive and delay-tolerant applications, from which twelve more combinations of BRBs are found, as presented in Table 12.

As shown in Table 12, the BRB workload orchestrator uses a Conjunctive BRB for BRBM1 in the first stage and a disjunctive BRB for BRBM2 in the second stage for all types of applications (including both delay-sensitive and delay-tolerant applications) when BRB-comb-2 is used. However, if BRB-comb-12 is used, the BRB workload orchestrator uses a disjunctive BRB for BRBM1 in the first stage and a conjunctive BRB for BRBM2 in the second stage for delay-sensitive applications (AR and Health), while it uses a Conjunctive BRB for BRBM1 in the first stage and a disjunctive BRB for BRBM2 in the second stage for delay-tolerant applications (Infotainment and Compute-Intensive).

## V. RESULTS AND DISCUSSION

This section presents the performance evaluation of the two-stage BRB workload orchestrator for all possible combinations of BRBs in terms of three performance metrics. Based on the performance evaluation, the best combination of BRBs for the two-stage BRB workload orchestrator is determined. After that, the performance of the BRB workload orchestrator with the best combination of BRBs is compared with four workload orchestration approaches from the literature regarding three performance metrics.

### A. PERFORMANCE EVALUATION OF THE TWO-STAGE BRB WORKLOAD ORCHESTRATOR

An extensive number of simulation experiments are conducted using the EdgeCloudSim simulator [45] to assess the performance of the two-stage BRB workload orchestrator for each combination of BRBs in the first and second stages. As discussed in subsection IV-A, the simulation experiment is repeated 30 times with 200 to 2400 User Equipment (UE) and four types of applications are used. Following [9], three performance metrics, namely the average failed task (%), average service time (sec), and average VM utilization on edge servers (%), are considered for performance evaluation. In order to conduct a performance comparison under high workloads, the average failed task, the average service time, and the average VM utilization for each combination of BRBs in the two-stage BRB workload orchestrator regarding 2400 UE are considered, as depicted in Fig. 8, Fig. 9, and Fig. 10, respectively.

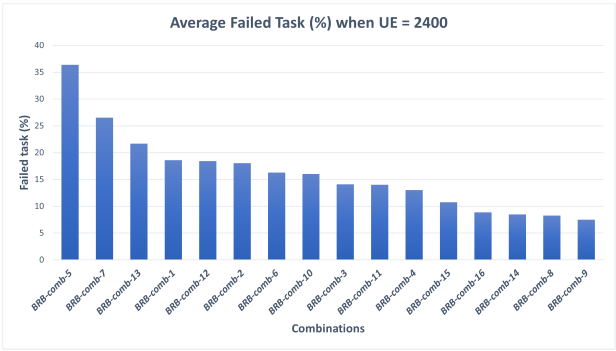



**FIGURE 8.** Average failed task for each combination of BRBs in the BRB workload orchestrator.

When the number of UE is 2400, it can be noticed from Fig. 8 that the average failed task (%) for BRB-comb-9 is the lowest among all combinations. However, it can be observed from Fig. 9 and Fig. 10 that the average service time (sec) and the average VM utilization (%) for BRB-comb-14 are the lowest among all combinations, while BRB-comb-9 has the third-lowest average service time and the fourth-lowest average VM utilization. Since BRB-comb-14 performs best in aspects of two performance metrics, BRB-comb-14 is considered the best combination of BRBs for the two-stage BRB workload orchestrator. It means the proposed two-stage BRB workload orchestrator performs best when disjunctive BRBs are used for both

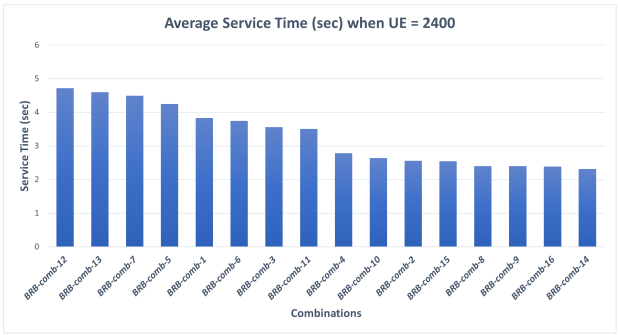



**FIGURE 9.** Average service time for each combination of BRBs in the BRB workload orchestrator.

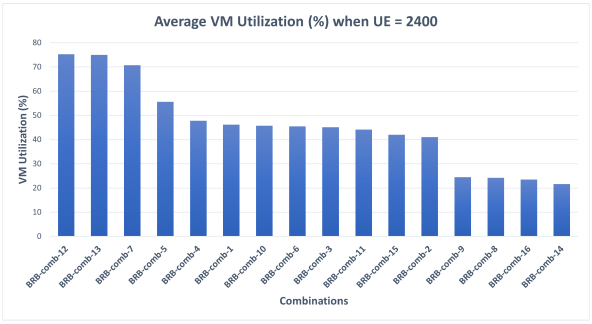



**FIGURE 10.** Average VM utilization for each combination of BRBs in the BRB workload orchestrator.

BRBM1 (in the first stage) and BRBM2 (in the second stage) for delay-sensitive applications (AR and Health), and Conjunctive BRBs are used for both BRBM1 (in the first stage) and BRBM2 (in the second stage) for delay-tolerant applications (Infotainment and Compute-Intensive). In the next subsection, the performance of the two-stage BRB workload orchestrator with the best combination of BRBs is compared with four workload orchestration approaches from the literature.

### B. PERFORMANCE COMPARISON WITH OTHER WORKLOAD ORCHESTRATION APPROACHES

The performance of the proposed BRB workload orchestration approach is comparatively evaluated under various loads with four workload orchestration approaches from the literature, namely Fuzzy Logic-Based (FLB) workload orchestration [9], Bandwidth-based workload orchestration (which uses WAN bandwidth for task offloading decision) [9], Utilization-based workload orchestration (which uses CPU utilization of Edge VMs for task offloading decision) [9], and Hybrid workload orchestration (which uses both WAN bandwidth and CPU utilization of Edge VMs for task offloading decision) [9]. The details of these workload orchestration approaches can be found in [9]. As discussed in section IV-A, the simulation experiment is repeated 30 times with 200 to 2400 User Equipment (UE) and four types of applications are used. Following [9], three performance metrics, namely the average failed task (%), average service time (sec), and average VM utilization on Edge servers

(%), are considered to perform comparisons among these workload orchestration approaches.

### 1) AVERAGE FAILED TASK

For all applications, the average failed task (%) of all workload orchestration approaches regarding the number of UE is depicted in Fig. 11.

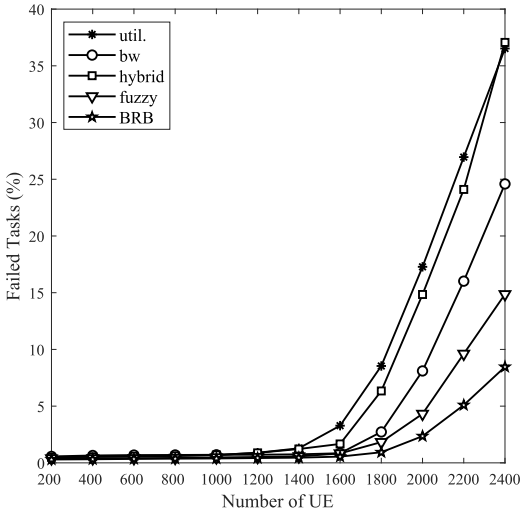



**FIGURE 11.** Average failed task for different workload orchestration approaches.

From Fig. 11, it can be observed that when the number of UE is between 200 and 1400, the average failed task for all workload orchestration approaches are similar. It means all workload orchestration approaches perform well when workloads are not too high. However, when the number of UE is between 1600 and 2400, the average failed task for all workload orchestration approaches starts increasing due to high workloads. In that case, the average failed task for the proposed BRB workload orchestrator is the lowest compared to other approaches. It means the BRB workload orchestrator can handle high workloads better than other approaches while maintaining a minimal failed task ratio.

After this performance comparison, the performance of all approaches is investigated with regard to the average failed task for each application. For each application, the average failed task (%) of all workload orchestration approaches regarding the number of UE is depicted in Fig. 12.

As discussed in section IV-A, the AR and Health applications mostly generate medium-sized and small tasks (task length of the AR and Health application is an average of 9 GI and 3 GI, which leads to 6% and 2% CPU utilization on an Edge VM) and they are delay-sensitive applications, while the Infotainment and Compute-Intensive applications generate moderate and big tasks (task length of the Infotainment and Compute-Intensive application is an average of 15 GI and 45 GI, which leads to 10% and 30% CPU utilization on an edge VM) and they are delay-tolerant applications. As discussed in subsection V-A, the proposed BRB workload orchestrator uses disjunctive BRBs for delay-sensitive applications and Conjunctive BRBs

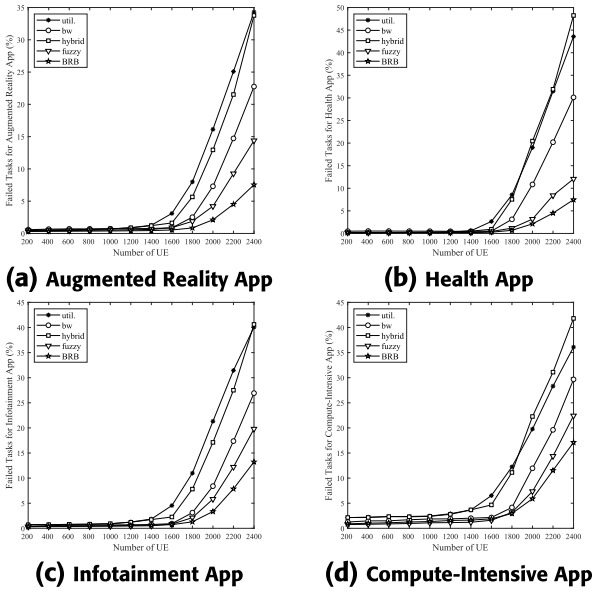



**FIGURE 12.** Average failed task for different applications.

for delay-tolerant applications. Both disjunctive BRBs and conjunctive BRBs are established in such a way that they can properly distinguish delay-sensitive or delay-tolerant tasks with different lengths and determine the optimal target server for processing each task accordingly. For this reason, when the BRB workload orchestrator is used, the average failed task for each application is the lowest compared to other approaches, as shown in Fig. 12.

After this investigation, the reasons for task failures for all workload orchestration approaches are analyzed. During the simulation, task failures occur for all workload orchestration approaches because of the following four reasons.

(i) Lack of VM Capacity on Edge Servers: A task offloaded to an edge server fails if the average CPU utilization of that edge server is extravagant and it does not have adequate VM capacity to process it.

(ii) Mobility of End-users: If an end-user changes his location before receiving a response for an offloaded task, it fails since the end-user is out of his previous WLAN coverage and connected to a new WLAN after changing the previous location. Task failures due to the mobility of end-users are inevitable for all workload orchestration approaches since task level handoff is not taken into account.

(iii) Lack of MAN capacity: If MAN is utilized for numerous end-users simultaneously, it causes MAN congestion due to insufficient MAN bandwidth resulting in task failures.

(iv) Lack of WAN capacity: Similar to MAN, if WAN is used for numerous end-users simultaneously, it causes WAN congestion due to insufficient WAN bandwidth resulting in task failures. Lack of WLAN capacity is also a reason for task failures, but only a small number of task losses because of insufficient WLAN capacity

in the simulation for all approaches. Hence, this reason is neglected.

The task failure ratio with 2400 UE is considered to analyze task failure reasons for all workload orchestration approaches under high workloads, as illustrated in Fig. 13.

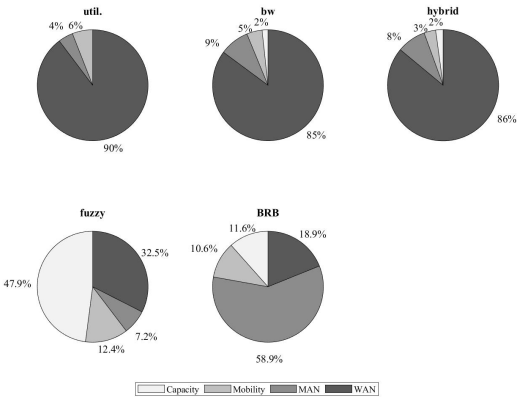



**FIGURE 13.** Reasons of task failures for different workload orchestration approaches.

From Fig. 13, it can be observed that when utilization-based, bandwidth-based, and hybrid workload orchestration approaches are used under high workloads, most tasks fail due to lack of WAN capacity, while few tasks fail due to mobility of end-users, lack of VM capacity on Edge servers, and lack of MAN capacity. These three approaches only consider WAN bandwidth and/or CPU utilization of Edge VMs for task offloading decisions, but they do not consider MAN delay, task length, and delay sensitivity of a task. As a result, they fail to optimally distribute tasks between edge and cloud servers and send a vast amount of tasks to the cloud server simultaneously, which causes WAN congestion due to insufficient WAN bandwidth resulting in task failures.

When it comes to the FLB and BRB workload orchestration approaches, both approaches consider CPU utilization of Local Edge VMs and Remote Edge VMs, MAN delay, WAN bandwidth, task length, and delay-sensitivity of a task to determine the optimal target server (among Local Edge, Remote Edge, and Cloud) for each offloaded task generated by delay-sensitive and delay-tolerant applications. Since computation capacities of edge servers are modest in comparison to cloud servers, Local and Remote Edge servers are usually preferred to process delay-sensitive and moderate tasks, while the cloud server is preferred to process delay-tolerant and resource-intensive tasks so that a balance of workloads between edge and cloud servers can be ensured. The BRB workload orchestrator uses disjunctive BRBs for delay-sensitive applications and conjunctive BRBs for delay-tolerant applications. Both disjunctive BRBs and conjunctive BRBs are established in such a way that they can properly distinguish delay-sensitive or delay-tolerant tasks with different lengths and determine the optimal target server for processing each task accordingly. As discussed in section IV-A, the AR and Health applications are delay-sensitive applications, and the interarrival time of tasks of these two applications is 2 and 3 seconds, which means a related task of the AR and Health application is sent from UE to the workload orchestrator every 2 and 3 seconds. When the number of UE is high, a considerable amount of tasks generated by these delay-sensitive applications are sent to the workload orchestrator. To meet end-user demands as well as low-latency requirements, the BRB workload orchestrator mostly distributes these delay-sensitive tasks between Local Edge and Remote Edge servers. However, when a vast amount of tasks are sent to Remote Edge servers via MAN, it causes MAN congestion due to insufficient MAN bandwidth. As a result, when the BRB workload orchestration approach is used, most tasks fail due to lack of MAN capacity, while few tasks fail due to mobility of end-users, lack of VM capacity on Edge servers, and lack of WAN capacity.

However, the FLB workload orchestrator uses the same fuzzy rule bases for both delay-sensitive and delay-tolerant applications. Since task requirements for both types of applications are different, it is not feasible to use the same fuzzy rule bases to distinguish tasks generated by delay-sensitive and delay-tolerant applications and determine the optimal target server for processing those tasks. Therefore, under high workloads, when the FLB workload orchestrator sends some delay-tolerant resource-intensive tasks to an edge server for processing rather than the cloud server due to incorrectly distinguishing those tasks, the CPU utilization of that edge server rises extremely, and it fails to process other incoming delay-sensitive tasks due to lack of VM capacity. Similarly, when it sends many delay-sensitive medium-sized tasks to the cloud server for processing due to incorrectly distinguishing those tasks, WAN congestion occurs due to insufficient WAN bandwidth. As a result, when the FLB workload orchestration approach is used, most tasks fail due to a lack of VM capacity on edge servers and lack of WAN capacity, while few tasks fail due to mobility of end-users and lack of MAN capacity.

The analysis of task failures will help service providers to identify the main reasons for task failures and allocate computational resources (VM capacity on Edge servers) and network resources (MAN and WAN bandwidth) for a MEC infrastructure accordingly.

### 2) AVERAGE SERVICE TIME

For all applications, the average service time (sec) of all workload orchestration approaches regarding the number of UE is shown in Fig. 14.

From Fig. 14, it can be noticed that as the number of UE increases, the average service time for all workload orchestration approaches rises due to an increase in workloads. For different types of workloads, the average service time for the proposed BRB workload orchestrator is consistently the lowest compared to other approaches. It means the BRB workload orchestrator performs better compared to other approaches by efficiently distributing each offloaded task

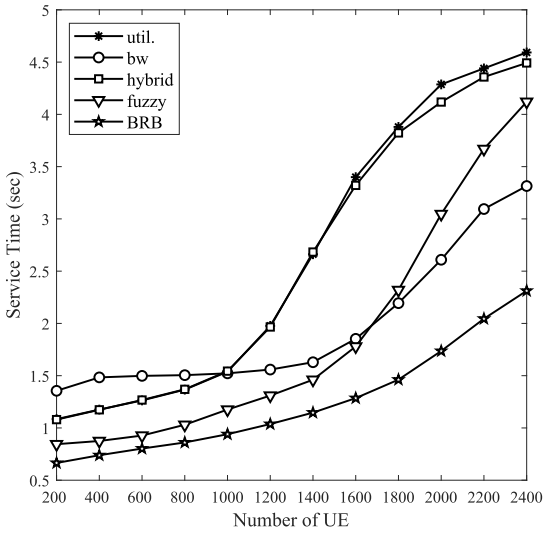



**FIGURE 14.** Average service time for different workload orchestration approaches.

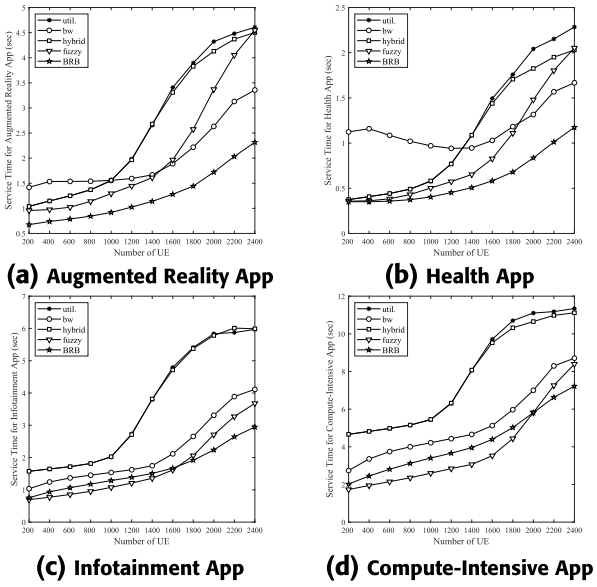



**FIGURE 15.** Average service time for different applications.

to the optimal target server under various workloads while maintaining a minimal average service time.

After this performance comparison, the performance of all approaches is investigated with regard to the average service time for each application. For each application, the average service time (sec) of all workload orchestration approaches regarding the number of UE is depicted in Fig. 15.

As discussed in section IV-A, the AR and Health applications are delay-sensitive applications, which mostly generate medium-sized and small tasks and require a faster response, while The Infotainment and Compute-Intensive applications are delay-tolerant applications, which mostly generate moderate and big tasks and have no strict response requirement.

The BRB workload orchestrator uses disjunctive BRBs for delay-sensitive applications, which mostly distributes these delay-sensitive small and medium-sized tasks between Local Edge and Remote Edge servers to provide faster response. In some situations, it also distributes delay-sensitive tasks to the cloud server according to disjunctive belief rules to fulfill end-user demands. For this reason, when the BRB workload orchestrator is used, the average service time for both delay-sensitive applications is the lowest compared to other approaches.

The BRB workload orchestrator uses conjunctive BRBs for delay-tolerant applications, which considers different possible network conditions, computation resources, and task properties in terms of conjunctive belief rules and then determines the optimal target server among Local Edge, Remote Edge, and Cloud servers to fulfill end-user demands as much as possible. When the BRB workload orchestrator is used under low workloads, the average service time for both delay-tolerant applications is a bit high compared to the FLB workload orchestration approach since it sends resource-intensive tasks to the cloud server rather than an edge server to maintain a balance of workloads between edge and cloud servers, which causes a relative high WAN delay. However, when the BRB workload orchestrator is used under high workloads, the average service time for both delay-tolerant applications is the lowest compared to all approaches since it maintains a balance of workloads by optimally distributing delay-tolerant tasks between edge and cloud servers.

After this investigation, the average service time for all workload orchestration approaches is analyzed. The service time comprises both processing time on edge/cloud servers and network delay (WLAN/MAN/WAN). If a task is executed on a Local Edge/Remote Edge server, the total service time includes WLAN/MAN delay and the processing time on a Local/Remote Edge VM. On the contrary, if a task is executed on the cloud server, the total service time comprises WAN delay and the processing time on a cloud VM. The service time on edge and cloud servers comprising both processing time and network delay for all workload orchestration approaches regarding the number of UE is shown in Fig. 16.

Since Local/Remote edge servers are located closer to UE at the network edge, WLAN/MAN delay is usually shorter, while the processing time on Local/Remote edge Servers is longer compared to WLAN/MAN delay due to their modest computation capacities. If the number of UE increases, both processing time and communication delay rise on edge servers due to an increase in workloads, but the growth rate of processing time is larger compared to WLAN/MAN delay on edge servers since it takes a substantial amount of time to execute a vast amount of tasks offloaded to edge servers due to their modest computation capacities. On the contrary, since the cloud server is located far from UE, WAN delay is longer, while the processing time on the cloud server is shorter compared to WAN delay due to its powerful computation capacities. If the number of UE increases, both processing time and WAN delay rise on the cloud server due to an increase in workloads, but the growth rate of processing time

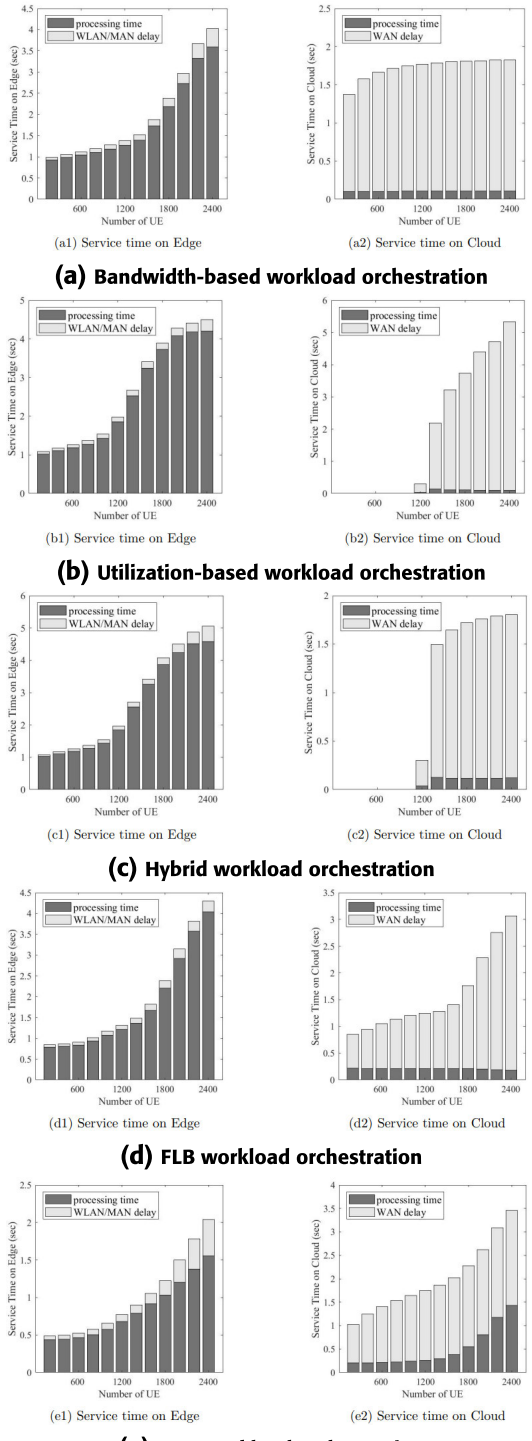



**FIGURE 16.** Average service time analysis for different workload orchestration approaches.

is lower compared to WAN delay since the cloud server can execute a lot of resource-intensive tasks at high speed due to its powerful computation capacities. For all workload orchestration approaches, these characteristics of service time on edge/cloud servers can be seen in Fig. 16.

Bandwidth-based workload orchestration approach prefers offloading tasks to the cloud server as long as WAN is not congested. If the current WAN bandwidth is greater than the minimum WAN bandwidth (threshold), it offloads tasks to the cloud server, otherwise, it offloads tasks to edge servers. From Fig. 16a, it can be seen that when the workload is low, this approach first offloads most tasks to the cloud server, and when WAN is congested, it offloads the rest of the tasks to edge servers. Hence, the service time on the cloud server is much larger due to a high WAN delay. As the workload increases, the service time on edge servers keeps rising since it pushes most tasks to edge servers after WAN is congested. Though the cloud server has a huge computation capacity, it can not process any more tasks since WAN is congested. As a result, the service time on edge servers is quite high and there is no workload balance between cloud and edge servers.

Utilization-based workload orchestration approach prefers offloading tasks to edge servers until Edge VMs are not congested. If the current CPU utilization of VMs running on edge servers is lower than the maximum CPU utilization (threshold), it offloads tasks to edge servers, otherwise, it offloads tasks to the cloud server. From Fig. 16b, it can be seen that when the workload is low, this approach offloads all tasks to edge servers, and the service time remains low on edge servers. As the workload increases, it first offloads most tasks to edge servers, and when Edge VMs are congested, it offloads the remaining tasks to the cloud server. Since it first pushes most tasks to edge servers, the service time on edge servers keeps increasing since it takes a substantial amount of time to execute a vast amount of tasks offloaded to edge servers due to their modest computation capacities. Besides, when a huge amount of tasks are offloaded to the cloud server after the congestion of Edge VMs, the service time on the cloud server keeps increasing due to a high WAN delay. As a result, under high workloads, the service time on both edge and cloud servers is quite high and there is no workload balance between cloud and edge servers.

The Hybrid workload orchestration approach offloads tasks to the cloud server if the current WAN bandwidth is greater than the minimum WAN bandwidth (threshold) and the current CPU utilization of VMs running on edge servers is higher than the maximum CPU utilization (threshold). Otherwise, it offloads tasks to edge servers. From Fig. 16c, it can be seen that when the workload is low, this approach offloads all tasks to edge servers since the current CPU utilization of VMs running on edge servers is lower than the CPU utilization threshold. Under low workloads, the service time remains low on edge servers. As the workload increases, it first offloads most tasks to edge servers. When the current CPU utilization of VMs running on edge servers is higher than the CPU utilization threshold and the current WAN bandwidth is greater than the WAN bandwidth threshold, it offloads the remaining tasks to the cloud server. Since it first pushes most tasks to edge servers, the service time on edge servers keeps increasing since it takes a substantial amount of time to execute a vast amount of tasks offloaded to edge servers because of their modest computation capacities. Besides, when a huge amount of tasks are offloaded to the

cloud server after the current CPU utilization of VMs running on edge servers reaches the threshold, the service time on the cloud server keeps increasing due to a high WAN delay. As a result, under high workloads, the service time on both edge and cloud servers is quite high and there is no workload balance between cloud and edge servers.

When it comes to the FLB and BRB workload orchestration approaches, both approaches consider CPU utilization of Local Edge VMs and Remote Edge VMs, MAN delay, WAN bandwidth, task length, and delay-sensitivity of a task to determine the optimal target server (among Local Edge, Remote Edge, and Cloud) for each offloaded task generated by delay-sensitive and delay-tolerant applications. Since computation capacities of edge servers are modest in comparison to cloud servers, Local and Remote Edge servers are usually preferred to process delay-sensitive and moderate tasks, while the cloud server is preferred to process delay-tolerant and resource-intensive tasks so that a balance of workloads between edge and cloud servers and a minimal service time on both edge and cloud servers can be ensured. The BRB workload orchestrator uses disjunctive BRBs for delay-sensitive applications and conjunctive BRBs for delay-tolerant applications. Both disjunctive BRBs and conjunctive BRBs are established in such a way that they can properly distinguish delay-sensitive or delay-tolerant tasks with different lengths and determine the optimal target server for processing each task accordingly. The BRB workload orchestrator mostly distributes delay-sensitive tasks between Local Edge and Remote Edge servers to meet low-latency requirements. In some situations, it also distributes delay-sensitive tasks to the cloud server according to disjunctive belief rules to fulfill end-user demands. The BRB workload orchestrator uses conjunctive BRBs for delay-tolerant tasks, which considers different possible network conditions, computation resources, and task properties in terms of conjunctive belief rules and then determines the optimal target server among Local Edge, Remote Edge, and Cloud servers to fulfill end-user demands as much as possible. As a result, under various workloads, there is a balance of workload between cloud and edge servers and the service time on both edge and cloud servers is quite minimal, as depicted in Fig. 16e.

However, the FLB workload orchestrator uses the same fuzzy rule bases for both delay-sensitive and delay-tolerant applications. Since task requirements for both types of applications are different, it is not feasible to use the same fuzzy rule bases to distinguish tasks generated by delay-sensitive and delay-tolerant applications and determine the optimal target server for processing those tasks. Under low and medium workloads, the service time on both edge and cloud servers is quite minimal. However, under high workloads, the service time on edge servers is quite high since the FLB workload orchestrator sends delay-tolerant resource-intensive tasks to edge servers for processing rather than the cloud server due to incorrectly distinguishing those tasks, as depicted in Fig. 16d.

### 3) AVERAGE VM UTILIZATION ON EDGE SERVERS

The average Virtual Machine (VM) utilization on edge servers (%) for all workload orchestration approaches regarding the number of UE is depicted in Fig. 17.

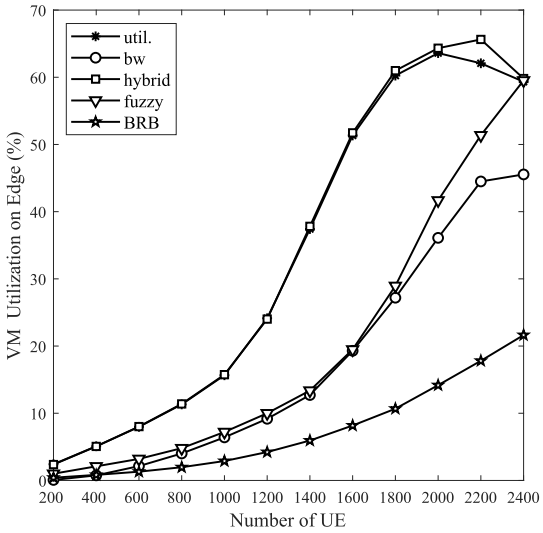



**FIGURE 17. Average VM utilization on edge servers for different workload orchestration approaches.**

As discussed in section IV-A, 14 edge servers are considered in the simulation experiment, where each edge server has 8 VMs. From Fig. 17, it can be observed that as the number of UE increases, the average VM utilization on edge servers for all workload orchestration approaches rises due to an increase in workloads. For different types of workloads, the average VM utilization on edge servers for the proposed BRB workload orchestrator is consistently the lowest compared to other approaches. It means the BRB workload orchestrator performs better compared to other approaches by efficiently distributing each offloaded task to the optimal edge server under various workloads and there is no waste of computation resources when the BRB workload orchestrator is used since all VMs are utilized efficiently on edge servers.

## VI. CONCLUSION

Multi-access Edge Computing (MEC) is a standard network architecture of edge computing that has emerged to handle a massive amount of diverse data traffic, enhance computation capabilities, and reduce communication latency. However, there exist several critical challenges during workload orchestration in MEC. Firstly, a task generated by delay-sensitive or delay-tolerant applications should appropriately be distinguished when determining the optimal edge/cloud server for processing that task since task requirements for delay-sensitive and delay-tolerant applications are different. Secondly, due to the high mobility of end-users, unknown numbers of offloading requests from different types of UE, and intermittent data traffic, the state of the network condition keeps changing, and computational capacities of edge and cloud servers keep fluctuating, which arises a

rapidly changing dynamic environment at the network edge due to uncertainty. Finally, if offloaded tasks from UE are distributed unevenly and randomly within the overall MEC infrastructure, it will lead to imbalance workloads between edge and cloud servers, increase task latency due to improper task offloading, and impair the overall performance and efficiency of MEC. In order to address these workload orchestration challenges in MEC, a BRB workload orchestrator is proposed in this study. The proposed BRB workload orchestrator uses belief rules to specify the required workload orchestration operations with regard to network conditions, computational resources, and properties of offloaded tasks from UE and decide the optimal execution location for each offloaded task within the overall MEC infrastructure. The performance of the proposed BRB workload orchestrator is compared with four workload orchestration approaches from the literature under various workloads in regard to three performance metrics. According to the result of simulation experiments, the proposed workload orchestrator outperforms other approaches in aspects of task failure rate, average service time, and average VM utilization on edge servers. Under various workloads, it reduces overall service time, minimizes task failures, and provides lower average VM utilization by efficiently utilizing all VMs on edge servers and balancing the workload among edge servers.

In this study, tasks generated by the four types of applications are considered independent tasks or stateless jobs. Besides, full computation offloading is considered, which means a whole task from UE is offloaded to an edge or cloud server for processing without any split of the task. In future studies, the dependency between tasks and partial computation offloading can be considered, where after splitting a task into sub-tasks with an optimal task-splitting strategy, some sub-tasks can be executed locally by UE if it has enough processing capacity, while the remaining sub-tasks can be offloaded to an edge and/or cloud servers for further processing. The proposed BRB workload orchestrator can be enhanced by considering partial computation offloading, which can further minimize task failures and reduce overall service time for delay-sensitive and resource-intensive applications through a collaboration among UE, Local Edge, Remote Edge, and Cloud servers and improve the overall performance and efficiency of MEC.

## REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[2] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[3] H. Li, G. Shou, Y. Hu, and Z. Guo, "Mobile edge computing: Progress and challenges," in *Proc. 4th IEEE Int. Conf. Mobile Cloud Comput., Services, Eng. (MobileCloud)*, Mar. 2016, pp. 83–84.

[4] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 369–392, 1st Quart., 2014.

[5] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[6] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[7] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[8] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[9] 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.

[10] L. Abdullah, "Fuzzy multi criteria decision making and its applications: A brief review of category," *Proc. Social Behav. Sci.*, vol. 97, pp. 131–136, Nov. 2013.

[11] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1818–1831, Jun. 2017.

[12] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[13] A. Al-Dulaimy, J. Taheri, A. Kassler, M. R. HoseinyFarahabady, S. Deng, and A. Zomaya, "MultiScaler: A multi-loop auto-scaling approach for cloud-based applications," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2769–2786, Oct. 2022.

[14] 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.

[15] W. Alasmary and S. Valaee, "Crowd sensing in vehicular networks using uncertain mobility information," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11227–11238, Nov. 2019.

[16] S. K. U. Zaman, A. I. Jehangiri, T. Maqsood, Z. Ahmad, A. I. Umar, J. Shuja, E. Alanazi, and W. Alasmary, "Mobility-aware computational offloading in mobile edge networks: A survey," *Cluster Comput.*, vol. 24, no. 4, pp. 2735–2756, Dec. 2021.

[17] J.-B. Yang, J. Liu, J. Wang, H.-S. Sii, and H.-W. Wang, "Belief rule-base inference methodology using the evidential reasoning approach-RIMER," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 36, no. 2, pp. 266–285, Mar. 2006.

[18] J. Liu, L. Martinez, A. Calzada, and H. Wang, "A novel belief rule base representation, generation and its inference methodology," *Knowl.-Based Syst.*, vol. 53, pp. 129–141, Nov. 2013.

[19] S. Wang, M. Zafer, and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.

[20] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.

[21] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2146–2153, Jun. 2018.

[22] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "Cost-effective processing for delay-sensitive applications in cloud of things systems," in *Proc. IEEE 15th Int. Symp. Netw. Comput. Appl. (NCA)*, Oct. 2016, pp. 162–169.

[23] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-aware resource allocation for edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jun. 2017, pp. 47–54.

[24] V. Scoca, A. Aral, I. Brandic, R. De Nicola, and R. B. Uriarte, "Scheduling latency-sensitive applications in edge computing," in *Proc. 8th Int. Conf. Cloud Comput. Services Sci.*, 2018, pp. 158–168.

[25] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 1222–1228.

[26] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 26–35, Mar. 2017.

[27] T. Zheng, J. Wan, J. Zhang, and C. Jiang, "Deep reinforcement learning-based workload scheduling for edge computing," *J. Cloud Comput.*, vol. 11, no. 1, p. 3, Dec. 2022.

[28] B. Yamansavascilar, A. C. Baktir, C. Sonmez, A. Ozgovde, and C. Ersoy, "DeepEdge: A deep reinforcement learning based task orchestrator for edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 1, pp. 538–552, Jan. 2023.
[29] G.-Y. Hu and P.-L. Qiao, "Cloud belief rule base model for network security situation prediction," *IEEE Commun. Lett.*, vol. 20, no. 5, pp. 914–917, May 2016.
[30] B.-C. Zhang, G.-Y. Hu, Z.-J. Zhou, Y.-M. Zhang, P.-L. Qiao, and L.-L. Chang, "Network intrusion detection based on directed acyclic graph and belief rule base," *ETRI J.*, vol. 39, no. 4, pp. 592–604, Aug. 2017.
[31] S. Li, J. Feng, W. He, R. Qi, and H. Guo, "A new health prediction model for a sensor network based on belief rule base with attribute reliability," *Sci. Rep.*, vol. 11, no. 1, pp. 1–10, Feb. 2021.
[32] W. He, C.-Q. Yu, G.-H. Zhou, Z.-J. Zhou, and G.-Y. Hu, "Fault prediction method for wireless sensor network based on evidential reasoning and belief-rule-base," *IEEE Access*, vol. 7, pp. 78930–78941, 2019.
[33] W. He, P.-L. Qiao, Z.-J. Zhou, G.-Y. Hu, Z.-C. Feng, and H. Wei, "A new belief-rule-based method for fault diagnosis of wireless sensor network," *IEEE Access*, vol. 6, pp. 9404–9419, 2018.
[34] M. S. Hossain, K. Andersson, and S. Naznin, "A belief rule based expert system to diagnose measles under uncertainty," in *Proc. World Congr. Comput. Sci., Comput. Eng., Appl. Comput. (WORLDCOMP'15) Int. Conf. Health Inform. Med. Syst.* Las Vegas, NV, USA: CSREA Press, 2015, pp. 17–23.
[35] M. S. Hossain, A. A. Monrat, M. Hasan, R. Karim, T. A. Bhuiyan, and Md. S. Khalid, "A belief rule-based expert system to assess mental disorder under uncertainty," in *Proc. 5th Int. Conf. Informat., Electron. Vis. (ICIEV)*, May 2016, pp. 1089–1094.
[36] M. S. Hossain, F. Ahmed, Fatema-Tuj-Johora, and K. Andersson, "A belief rule based expert system to assess tuberculosis under uncertainty," *J. Med. Syst.*, vol. 41, no. 3, pp. 1–11, Mar. 2017.
[37] M. S. Hossain, S. Rahaman, A.-L. Kor, K. Andersson, and C. Pattinson, "A belief rule based expert system for datacenter PUE prediction under uncertainty," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 140–153, Apr. 2017.
[38] R. Ul Islam, M. S. Hossain, and K. Andersson, "Inference and multi-level learning in a belief rule-based expert system to predict flooding," in *Proc. Joint 9th Int. Conf. Informat., Electron. Vis. (ICIEV) 4th Int. Conf. Imag., Vis. Pattern Recognit. (icIVPR)*, Aug. 2020, pp. 1–10.
[39] Q. Xiong, G. Chen, Z. Mao, T. Liao, and L. Chang, "Computational requirments analysis on the conjunctive and disjunctive assumptions for the belief rule base," in *Proc. Int. Conf. Mach. Learn. Cybern. (ICMLC)*, vol. 1, Jul. 2017, pp. 236–240.
[40] L. Chang, X. Ma, L. Wang, and X. Ling, "Comparative analysis on the conjunctive and disjunctive assumptions for the belief rule base," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery (CyberC)*, Oct. 2016, pp. 153–156.
[41] D.-L. Xu, J. Liu, J.-B. Yang, G.-P. Liu, J. Wang, I. Jenkinson, and J. Ren, "Inference and learning methodology of belief-rule-based expert system for pipeline leak detection," *Exp. Syst. Appl.*, vol. 32, no. 1, pp. 103–113, Jan. 2007.
[42] M. N. Jamil, M. S. Hossain, R. U. Islam, and K. Andersson, "Belief rule-based adaptive particle swarm optimization," in *Data Science and Big Data Analytics in Smart Environments*. Boca Raton, FL, USA: CRC Press, 2021, pp. 88–107.
[43] Y.-M. Wang, J.-B. Yang, and D.-L. Xu, "Environmental impact assessment using the evidential reasoning approach," *Eur. J. Oper. Res.*, vol. 174, no. 3, pp. 1885–1913, Nov. 2006.
[44] R. Ul Islam, M. S. Hossain, and K. Andersson, "A novel anomaly detection algorithm for sensor data under uncertainty," *Soft Comput.*, vol. 22, no. 5, pp. 1623–1639, Mar. 2018.
[45] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, pp. 1–17, 2018.
[46] C. Sonmez. *EdgeCloudSim*. Accessed: May 1, 2022. [Online]. Available: https://github.com/CagataySonmez/EdgeCloudSim
[47] M. Silva, D. Freitas, E. Neto, C. Lins, V. Teichrieb, and J. M. Teixeira, "Glassist: Using augmented reality on Google glass as an aid to classroom management," in *Proc. XVI Symp. Virtual Augmented Reality*, 2014, pp. 37–44.
[48] C. Tunca, N. Pehlivan, N. Ak, B. Arnrich, G. Salur, and C. Ersoy, "Inertial sensor-based robust gait analysis in non-hospital settings for neurological disorders," *Sensors*, vol. 17, no. 4, p. 825, Apr. 2017.
[49] J. Guo, B. Song, Y. He, F. R. Yu, and M. Sookhak, "A survey on compressed sensing in vehicular infotainment systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2662–2680, 4th Quart., 2017.
[50] A. Kumar, P. Srikanth, A. Nayyar, G. Sharma, R. Krishnamurthi, and M. Alazab, "A novel simulated-annealing based electric bus system design, simulation, and analysis for dehradun smart city," *IEEE Access*, vol. 8, pp. 89395–89424, 2020.
[51] S. Kamouskos, "Supporting nomadic users within virtual private networks," in *Proc. IEEE Globecom Workshop IEEE Service Portability Virtual Customer Environments*, Jul. 2000, pp. 128–133.

**MOHAMMAD NEWAJ JAMIL** received the B.Sc. (Eng.) degree in computer science and engineering from the University of Chittagong, Bangladesh, and the erasmus mundus joint master's degree in GrEen NetworkIng And cLoud computing (GENIAL). His research interests include artificial intelligence, machine learning, data sciences and analytics, cloud computing, and the Internet of Things.

**MOHAMMAD SHAHADAT HOSSAIN** (Senior Member, IEEE) received the M.Phil. and Ph.D. degrees in computation from the Institute of Science and Technology (UMIST), University of Manchester, U.K., in 1999 and 2002, respectively. He is currently a Professor in computer science and engineering with the University of Chittagong, Bangladesh, and a Visiting Professor with the Luleå University of Technology, Sweden. His current research interests include e-government, the modeling of risks and uncertainties using evolutionary computing techniques, investigation of pragmatic software development tools and methods, information systems in general, and expert systems.

**RAIHAN UL ISLAM** received the M.Sc. degree in mobile systems and the Ph.D. degree in pervasive and mobile computing from the Luleå University of Technology, Sweden. Previously, he was a Software Engineer with the NEC Laboratories Europe, Context-Aware Services (CAS) and Smart Environments Technologies Group. His research interests include expert systems, machine learning, smart cities, M2M communication, mobile systems, and pervasive and ubiquitous computing.

**KARL ANDERSSON** (Senior Member, IEEE) received the M.Sc. degree in computer science and technology from the Royal Institute of Technology, Stockholm, Sweden, and the Ph.D. degree in mobile systems from the Luleå University of Technology, Sweden. After pursuing postdoctoral research with the Internet Real-Time Laboratory, Columbia University, New York City, USA, and the National Institute of Information and Communications Technology, Tokyo, Japan, he is currently a Professor in pervasive and mobile computing with the Luleå University of Technology. His research interests include green and mobile computing, the Internet of Things, cloud technologies, and information security.