Computers, Materials & Continua
DOI:10.32604/cmc.2022.024012
Article

Fuzzy Control Based Resource Scheduling in IoT Edge Computing

Samah Alhazmi, Kailash Kumar* and Soha Alhelaly

College of Computing and Informatics, Saudi Electronic University, Riyadh, Kingdom of Saudi Arabia
*Corresponding Author: Kailash Kumar. Email: k.kumar@seu.edu.sa
Received: 30 September 2021; Accepted: 12 November 2021

Abstract: Edge Computing is a new technology in Internet of Things (IoT) paradigm that allows sensitive data to be sent to disperse devices quickly and without delay. Edge is identical to Fog, except its positioning in the end devices is much nearer to end-users, making it process and respond to clients in less time. Further, it aids sensor networks, real-time streaming apps, and the IoT, all of which require high-speed and dependable internet access. For such an IoT system, Resource Scheduling Process (RSP) seems to be one of the most important tasks. This paper presents a RSP for Edge Computing (EC). The resource characteristics are first standardized and normalized. Next, for task scheduling, a Fuzzy Control based Edge Resource Scheduling (FCERS) is suggested. The results demonstrate that this technique enhances resource scheduling efficiency in EC and Quality of Service (QoS). The experimental study revealed that the suggested FCERS method in this work converges quicker than the other methods. Our method reduces the total computing cost, execution time, and energy consumption on average compared to the baseline. The ES allocates higher processing resources to each user in case of limited availability of MDs; this results in improved task execution time and a reduced total task computation cost. Additionally, the proposed FCERS m 1m may more efficiently fetch user requests to suitable resource categories, increasing user requirements.

Keywords: IoT; edge computing; resource scheduling; task scheduling; fuzzy control

1 Introduction

The collection of wireless interconnected items such as mobiles, sensors, Radio-Frequency Identification (RFID) tags and others form Internet of Things (IoT). These devices can utilise a unique addressing mechanism, but they have limited computational capacity and battery power. IoT devices can create enormous data in real-time and transfer it to cloud computing data centers located far away. Despite this, there is massive traffic and significant waits. As a result, more processing power and storage resources are needed by the IoT networks. Multi-access Edge Computing (MEC) is a novel technique that addresses these needs by providing a distributed computing model for networking, task processing, and data storage at IoT network edges [1]. MEC can fulfil the resource needs of IoT applications while also lowering the latency in communication. Each MEC server consists of a virtualized environment that includes a computing device, data storage and wireless communication unit. The MEC servers housed several virtual resources that could be used to service stationary IoT or Mobile Devices (MD). A single-hop wireless connection, such as 4G LTE devices, Wi-Fi, Bluetooth, and other wireless interfaces, allows an MD to interact directly with MEC servers. The internet that helps to connect the cloud infrastructure is used by the MEC Servers [2]. Fig. 1 shows the MEC's architecture, including wireless connections connecting IoT devices to the MEC [3,4].

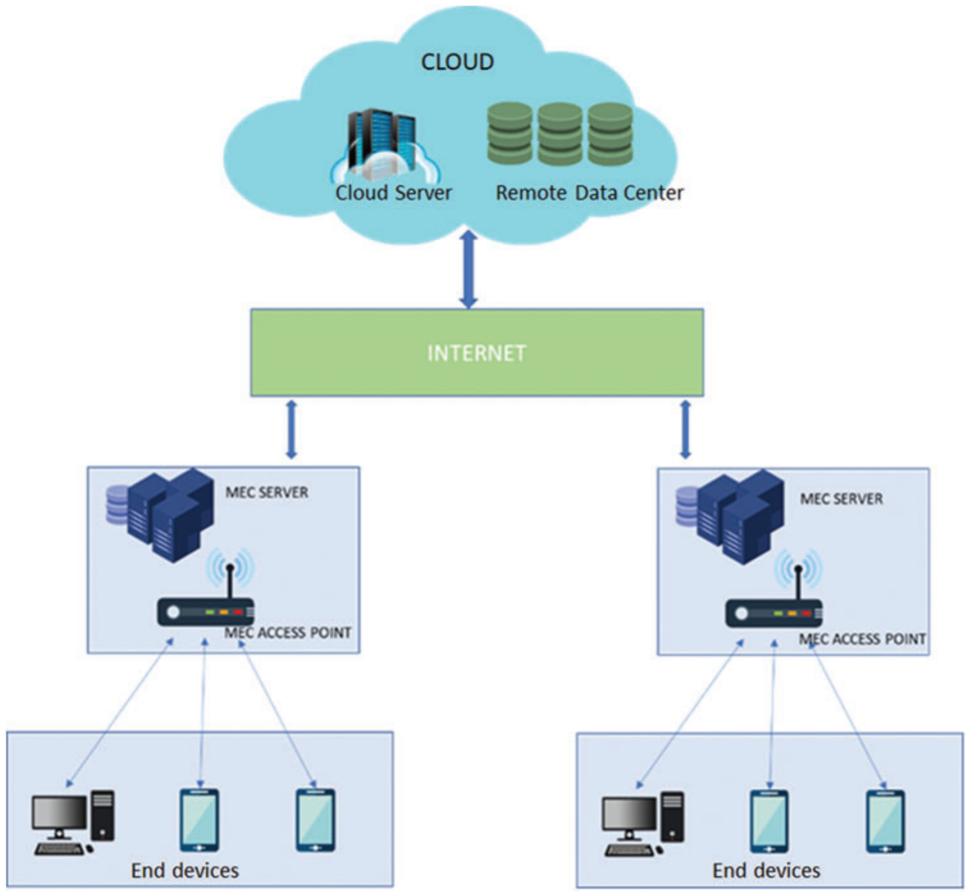


Figure 1: MEC architecture

End-users, objects, and sensors are closer to EC's storage, processing, control, and networking resources. An Edge Node can be as small as a smartphone, an Access Point (AP), a Base Station (BS), or even a cloud. A smartphone links wearable gadgets to the Cloud, a home gateway connects the Cloud with MD and the MDs to the core network. Edge Computing (EC) provides several outstanding features, including location awareness, resource pooling, distributed caching, processing of data analysis, scaling of resources, increased privacy and security, and dependable connection by delivering resource flexibility and intelligence. EC is a critical component of low-latency and good reliability. [5] contain many EC advantages and various use cases (e.g., media advertising, medical, offloading, smart homes, caching, and others).

IoT support, Network architecture design, service deployment, Resource Scheduling Process (RSP) and administration, programming models and abstractions, privacy and security, incentive design, edge device dependability, and sustainability are just a few of the additional issues that EC confronts [6,7]. The topic of Edge Computing Resource Allocation (RA) is the focus of this study. Unlike cloud computing, which has essentially infinite computational capacity and high network latency owing to the restricted computation capacity of Edge Nodes (EN), EC has comparatively less network delay but significant delay in processing because of the restricted computation capacity of EN. In addition, there are many small nodes in DC, equivalent to a lesser number of big Distributed Computing (DC). ENs can also be of various sizes (for example, the number of processing elements) and configurations (e.g., computing speed), extending from an MD towards 10's/100's of ES. These nodes are distributed across many sites, resulting in variable network and service delays for end-users. EC systems would most likely be made up of disparate communication, processing resources and Wi-Fi devices (e.g., devices that differ in computing capabilities, battery states, and types of computational tasks). Before adopting compute offloading paradigms, three key issues must be solved.

(a) First, effective TS algorithms must consider the various features of computational tasks and the heterogeneity of communication and computing resources.
(b) Second, efficient administration of heterogeneous communication resources, which will be utilized for data transmission between end users' devices and external computer resources, is required.
(c) Finally, efficient management of heterogeneous computing resources will be necessary to complete the computational tasks offloaded by various MDs.

These three issues must be handled simultaneously, which is a problematic undertaking in extremely diverse EC systems.

Furthermore, Edge Servers (ES) frequently have restricted processing resources in the given deployment's cost benefits and scalability [8]. One rationale is that the number of ES is installed mainly in numbers, demanding consideration of the deployment's financial help. As a result, a particular ES may not require the same number of resources as a CS and cannot offer them. Following that, the ES will be unable to perform all tasks. The work will have a low processing efficiency and a long processing time when every task without any distinction is delivered to the ES. As a result, it's important to find how to offload work based on demand. The act of transferring computing work from a MD to a Cloud Server (CS) or an ES is known as offloading [9–11]. EC's fundamental and crucial issue is developing an offloading technique and offloading work to an ES, resulting in increased latency while consuming less power than local execution. As a result, MD must pick the best offloading technique for their purposes, such as balancing delays and power consumption [12,13], decreasing task processing time [14–16] and energy consumption [17,18].

RA issues have been discovered in much previous work, and the majority of them are concerned with computing or communication of TA for ES. Some researchers looked at how MDs allocate computational resources and control transmission power [19]. In reality, MD-RA and ES-RA will work together to produce an offloading strategy. As a result, the link between offloading decision creation and computing RA must be investigated.

This paper provides a dynamic Resource Scheduling Process (RSP) based on fuzzy control theory. The following are the paper's main contributions:

a) An RSP for EC.

b) A model of allocation of essential resources is created depending on consumers' long-term and ongoing resource requirements.
c) To build the RSP based on fuzzy control theory, the RSP is carried out. The scheme splits user needs and available resources into numerous resource clusters, making it easier to process fuzzy user requirements and improving cloud computing QoS.

The remainder of the work is structured in the following manner. The present RSP and related research efforts are discussed in Section 2. The suggested model based on fuzzy control is presented in Section 3, and a new Fuzzy Control based Edge Resource Scheduling (FCERS) is given in Section 4. The carried out experiments and findings are evaluated in Section 5. To conclude, we make a list of future research work.

2 Related Works

Since the concept of EC was recommended, academics have been interested in computing Task Scheduling (TS) as an essential research topic in EC. There has been a slew of similar study findings recently. The primary objective of the TS offload approach is to find if a task created by a user terminal should be offloaded and where should it be scheduled. By examining TS techniques' present research state, this part recognises the task schedule's research direction. Aazam et al. 2021 [20] propose a compute migration strategy for next-generation networks. A MEC approach based on Software Defined Networking (SDN) and Network Function Virtualization (NFV) technologies, as well as multi-attribute decision making and compute migration, is also presented. The authors used MATrix LABoratory (MATLAB) to conduct their tests, demonstrating multi-attribute decision making based on SDN and NFV that could pick the right MEC center, minimize server response time, and enhance Quality of Service (QoS).

The authors [21] sought to deal with RA on NFV-enabled MECs, to reduce mobile service latency and MEC expenses. They presented a dynamic RA technique based on operational cost, consisting of a quick incremental allocation mechanism. They demonstrated that, compared to stationary MECs, their approach could deploy resources to ensure applications' low latency needs while saving money. Pham's approach aims to improve gateway placement and multi-hop routing in NFV-enabled IoT (NIoT) and service placement at the MEC and cloud levels. To deal with a big NIoT system and optimise routing, RA for service functions, and gateway deployment, created approximation algorithms such as Service Placements Algorithm (SPA-1 and SPA-2), Gateway Placement, and Multi-hop Routing Algorithm (GPMRA). According to the authors, their approximation methods can minimize computing time while achieving near-optimal outcomes.

Literature [22–24] developed a QoS decision engine that efficiently offloads to optimize performance and delay in response to the requirement in EC for real-time optimization. The work [25] presents a game theory technique for calculating unload, explains the nature of the equilibrium distribution, establishes its presence, and calculates equilibrium using a polynomial-time dispersion approach. However, the level of difficulty is higher. A multi-user distributed online task migration system is proposed in the literature [26–28], which considers the user's selfish traits and is centred on the optimization theory presented by Lyapunov and an incentive scheme based on *peer-to-peer* file transfer. It provides *multi-hop* user involvement and common processing activities in the delay network. The present single-hop centralised collaboration method may considerably lower system power usage and enhance system performance.

A Task Scheduling Algorithm (TSA) by Literature [29] proposed a TS method for task and resource features, and then recommends a Genetic Algorithm (GA) firework detection mechanism and uses the concepts of explosion radius, that allow load balancing and decrease TS time. Noghani et al. [30] proposed the allotment of resources for multi-user mobile EC environments regarding the Time Division Multiple Access (TDMA) and Orthogonal Frequency Division Multiple Access (OFDMA); they solved the mixed-integer issue multi-objective optimization problem and successfully minimized the consumption of total weighted mobile energy. Yang et al. [31] introduced an energy-optimizing offloading technique with guaranteed latency for global optimization. This method uses an artificial algorithm based on a fish swarm while also looking at the interaction between the fronthaul and backhaul network states [32].

A work classification and resource usage-based task consolidation approach for a cloud computing environment was invented in 2016 [33]. They also devised a method of VM aggregation to balance power usage and TS processing time. A TS method in mobile cloud resources by dynamic clustering and the improved FCM method to reduce the number of corresponding criteria in the search was proposed in 2016 by Qiang et al. [34]. The experiment demonstrates that the matching strategy may be proactively modified depending on the matching value and reinforcement training. Advanced Network Credit Scheduler (ANCS), proposed by Ni et al. 2017 [35], is a new approach for maintaining QoS in virtualization using dynamic network allocation of resources. Proportional sharing based on weight, reserving the minimum bandwidth and restricting the maximum bandwidth limitation are among the performance standards that ANCS aims to offer at the same time to meet the demands of the diverse network and cloud users. EC provides services to MDs who are close to each other. EC allows widely dispersed edge computing clusters to share resources or cache data. Following that, we feature a panel of specialists that specialize in resource management and EC.

The deployment of the Steiner tree to examine the resource caching technique in EC is proposed by Su et al. [36]. When an EC server stores resources to decrease overall route weight, it first builds a Steiner tree; the tree helps to lower resource cache costs. The paper results demonstrate that the Steiner tree approach outperforms the shortest path strategy. A dynamic RA was proposed in 2017 by Rahbari et al. [37]. Its aim is resource management while screening, preparing, and encrypting data. For this aim, in an EC environment, the study provided resource management. Many factors that lead a client to quit, such as the service type, the cost of service, and the various waiver likelihood modifications, are considered in this article. The experiment results show that with the help of these characteristics, the service provider can estimate the number of resources needed [38].

Traditional TS tool is straightforward to use. For data sharing in the TS of multi-layer deadline-constrained scientific procedures, few Virtual Machines (VM), which are heterogeneous in nature, are provided with remote storage services by each supplier. One of the TS techniques that optimizes cost and makespan objectives is the earliest finish time. The authors considered the cloud service as commercial infrastructure. The Pareto front is used as a decision-making tool to assess the merits and demerits of several alternatives. The TS costs were lowered in half, but the turnaround time was increased by 5%. For spot VMs and on-demand instances, a Fault-tolerant TS [39] is addressed by bidding mechanism. This technique is inefficient due to the low-priced spot VM. The authors suggested a scheduler for real-time workflow [40]. Backward shifting, resource scaling up, and shrinking facilitation were three phases in their technique that improved resource use over prior baseline algorithms.

To solve the problems, heuristic approaches employ a set of rules. Traditional heuristics include first, best, and worst fit. Fog, Cloud, and Edge providers may use heuristic approaches to run large-scale applications and processes. Mtshali et al. [41] use a heuristic method with an objective function that includes the task's makespan and execution cost to schedule the tasks. The results show that this strategy is more efficient and has a lower needed cost when compared to other approaches. Based on cluster formation, Rodrigues et al. [42] present a coalitional game for radio access networks in the Fog. To increase network throughput, it employed an approach for dispersed TS of users. Other dynamic cloud TS approaches for vehicular are based on queue length and reaction time parameters. A stochastic Petri net and OpenStack were used to schedule activities and simulate them using the Markov single server system. In order to save energy, a heuristic method is employed in Xiaojun et al. [43] to plan simultaneous real-time activities in a heterogeneous network. In this work, nonlinear programming is utilised to choose frequencies and assign threads.

Cloud-Fog investigated the TS and suggested a solution based on heuristics for striking a compromise between maximum completion time and Cloud resource monetary costs. Kao et al. [44] examined primary TS issues in a software-defined embedded system to support fog computing. To enhance users' experience, difficulties were framed as a mixed-integer nonlinear programming problem and provided an efficient heuristic TS technique. Nath et al. [45] introduced a fog computing RA method based on Priced Timed Petri Nets (PTPN). PTPN task model was created using fog resource characteristics. In terms of efficiency, the proposed technique beats static allocation strategies. In addition, heuristic approaches were used to achieve RSP in certain studies.

3 The Proposed Approach

3.1 System Model

As indicated in Fig. 2, we examine a network architecture with Edge Server and Node Mobile Devices. The MD $u_i (i \in \{1, 2, \ldots, N\})$ and the ES have a channel bandwidth of w. Because the distance in the middle of u_i and the edge server is generally 1-hop, the propagation latency among u_i and the ES 'c' is insignificant. The MD u_i has only a limited amount of energy E_i^{max} and the network generates M tasks that must be performed during each time slot. Therefore, for each time slot t $(t \in \{1, 2, \cdots, T\})$ device u_i, generates a task $j (j \in \{1, 2, \ldots, M\})$ which is represented as below, Eq. (1)

$$T_{i,j,t} = (d_{i,j,t}, C^{U}_{i,j,t}, C^{E}_{i,j,t}, t^{deadline}_{i,j,t}, S_{i,j,t}) \tag{1}$$

where $d_{i,j,t}$ is the task's data volume. The task is performed locally in a heterogeneous ES that specifies the required number of CPU cycles. $C^{E}_{i,j,t}$ is the necessary number of CPU cycles. The $t^{deadline}_{i,j,t}$, 't' is the maximum execution time that may be tolerated. $S_{i,j,t}$ represents the task offloading strategies. $S_{i,j,t} = 1$, if the task is completed locally for the task $\tau_{i,j,t}$; otherwise, $S_{i,j,t} = 0$.

The task information and the RA scheme for local computing are communicated to the ES. Still, after every time slot, the MD produces tasks and assigns them to the computing resources available locally. Depending on the offloading policy, the task is done locally or offloaded. The offloading policy is calculated by the ES and returned to the MD.

3.2 A User Model

Based on the service level, the user is the one who rents cloud services and submits the tasks. The user set $U_i = \{U_1, U_2, \ldots, U_m\}$, which is a service entity with six tuples as defined below, Eq. (2)

$$U_i =< U_{id}, U_{names}, USL, U_{jobs,} U_{pos}, U_{\sec} > \tag{2}$$

U_{id} is the unique ID of the user in cloud services.

U_{name} is the name of the user.

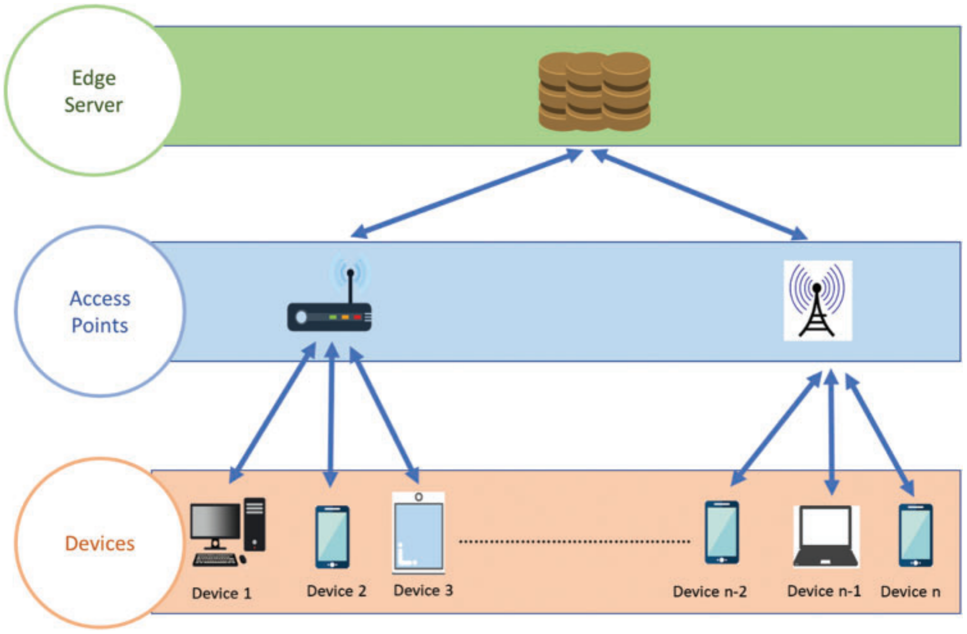


Figure 2: Network system model

The user is provided with a resource type given by SL (Service Level). USL provides the user with SL. $SL = \{Tier_I, Tier_{II}, Tier_{III}\}$, for the account of the users, these parameters are considered. U_{jobs} is the set of tasks submitted by the user. The data regarding the security of users are given by $U_{\sec}$.

3.3 B Task Model

Multiple tasks are carried out in parallel, and they must continually interact with one another. Each task monopolizes VM resources. To the feasible extent, it must be ensured that all tasks in the RSP execute in parallel, preventing task block and reducing operation run time. The act of sending employment applications is demonstrated in Eq. (3).

$$tasks = \{task_1, task_2, \ldots task_n\} \tag{3}$$

$task_t$ is expressed as below Eq. (4), which has 7 tuples

$$< t_{id}, t_{type}, t_{len}, t_{res}, t_{art}, t_{awt}, t_{tasks} > \tag{4}$$

The unique ID for $task_t$ is t_{id} and t_{type} gives the type of tasks, small and large, as shown in Eq. (5).

$$t_{type} = \{small|t_{len} \in [0, \max -1]\}, t_{type} = \{l\arg e|t \in [\max]\} \tag{5}$$

$t_{len} = |t_{tasks}|.t_{len}$, gives the length of tasks;

t_{tasks} is the set of tasks in one task $t_{tasks} = \{Task - 1, Task - 2, \ldots Task - k\}$

To run a task, the minimum set of resources required is given by t_{res}.

The parameter t_{art} gives the Average Response Time for a task, which is calculated by averaging the response time of each task to evaluate operation cost and performance. The time that it takes for a task is to be added to the TS queue and start running is the average wait time.

3.4 C Resource Model

The resources such as (services, applications, server, data store etc.,) are shared. Resource model shows $\mathrm{Re}source = \{r_1, r_2, \ldots, r_p\}$, total resource number$p = |\mathrm{Re}source, r_k|$, r_k which is the eight tuples: Computing resources that are shared and adjusted are referred to as resources (e.g., networks, applications, data center, servers and services, etc.). The resource model demonstrates the total number of resources. , r_k is a 9 tuple set shown below, Eq. (6)

$$r_k =< r_{id}, r_{\sup ply}, r_{type}, r_{comp}, r_{mem}, r_{stor}, r_{io}, r_{net}, r_{pos} > \tag{6}$$

where

r_{id} is the unique ID of the resource

$r_{\sup ply}$ is the provider of the resource

r_{type} is the type of the resource

r_{comp} is the capacity of resources to process information

r_{mem} is resources available memory

r_{stor} is the capacity of the resource storage

r_{io} is the capacity of IO in resources

r_{net} is the resource bandwidth

r_{pos} is the geographical resource position

3.5 Edge Computing

The data uplink transmission rate is required when the offloading of the task $\tau_{i,j,t}$ to the ES is chosen by the MD u_i, Eq. (7)

$$r_{i,j,t} = w\log_2\left(1 + \frac{p_i h_{i,j,t}}{\sigma + \sum_{k\in N, k\neq i}(1-s_{k,j,t})p_k h_{k,j,t}}\right) \tag{7}$$

where channel bandwidth is represented by w, the MD u_i's computation energy is given as p_i, where h_i and $\sigma$are the network gain and noise power, respectively. The inter-channel interference is given in denominator as cumulative term. Therefore, the uplink transmission time is, Eq. (8)

$$t_{i,j,t}^{Trans} = \frac{d_{i,j,t}}{r_{i,j,t}} \tag{8}$$

The heterogeneous ES that takes time $\tau_{i,j,t}$for the task is given in Eq. (9)

$$t_{i,j,t}^{Execute^i E} = \frac{c_{i,j,t}^E}{f_{i,t}^E} \tag{9}$$

where $f_{i,t}^E$ is the computing resource assigned to the task in the time slot 't' from u_i the ES; computing resources are allocated regularly, much as communication resources. The overall time spent on EC is given below as, Eq. (10):

$$t_{i,j,t}^{E} = t_{i,j,t}^{trans} + t_{i,j,t}^{Execute^{i}E} \tag{10}$$

Many expensive computation applications have a considerably lower quantity of data than the input and neglect the return time.

4 Resource Scheduling Algorithm Design

Initially, computer RA is done for all tasks. The offloading of all tasks cannot be considered the ultimate offloading choice owing to timeline and energy restrictions, and it is not fair to use the default offloading to allocate the computation. As a result, we present a Fuzzy Control based Edge Resource Scheduling (FCERS) approach in this part to arrive at appropriate offloading options. The first step is to apply a constraint-based filter. The new offloading technique is denoted by s′, and its initial values are $s' = s$. It should be noticed that one of the requirements for filtering is a time limitation. We may retrieve the entire duration of edge execution $t_{i,j,t}^{E}$ after getting the RA result $f_{i,t}^{E}$. And **If** $t_{i,j,t}^{E} > t_{i,j,t}^{deadline}$, **Then** s' (i, j, t) is set to 1. Furthermore, a local computational RSP is presented with energy restrictions, even though edge computations consume less energy than local computations. As a result, $s'(i, j, t) = 1$.

Algorithm: 1 of Constraint-Based Filtering

```
Step 1.   Input $\tau, s$
Step 2.   Output $s'$
Step 3.   Begin
Step 4.   $s' \leftarrow s$
Step 5.   While $(Any\_Task, \tau_{i,j,t})$ $not\_processed$ Do
Step 6.   Calculate $t_{i,j,t}^{E} = (\frac{d_{i,j,t}}{r_{i,j,t}}) + (\frac{c_{i,j,k}^{E}}{f_{i,t}^{E}})$
Step 7.   Calculate $e_{i,j,t}^{E} = p_i t_{i,j,t}^{t\ rans}$ using $p_i t_{i,j,t}^{trans} = p_i(\frac{d_{i,j,t}}{r_{i,j,t}})$
Step 8.   End While
Step 9.   If $t_{i,j,t}^{E} > t_{i,j,t}^{deadline}$ or $e_{i,j,t}^{deadline} > e_{i,j,t}^{U}$ Then
Step 10.  $s'_{i,j,t} = 1$
Step 11.  End If
Step 12.  End
```

User needs may be classified into several categories. The user's demands are matched with the resources in the class after locating the suitable resource category. The resource scale in the RSP is lowered after a fair split of resources. Simple weight matching is used to complete the RSP in this article. The following is the weight matching formula, Eq. (11):

$$Rank = Total(\mathrm{Re}sult(REQ_i - R_{Set_i}) \times \frac{Weight_i}{Total(Wight_i)} \tag{11}$$

where REQ_I denotes the attribute of the user's demands, $R_{set}.i$ represents the resource sets, and $Weight_i$ means the attribute weights.

The user's demand varies in different volumes of resources. Computing needs, bandwidth and storage requirements may be split into three categories for distinct task preferences. Each task, like resources, has three characteristics, each of which has a distinct weight. The user-required attribute and the resource attribute are combined in the formula mentioned earlier, and the highest score achieved is returned to the user as the outcome of the RSP. The following is the pseudo-code for the RSP algorithm Fuzzy Control based Edge Resource Scheduling (FCERS).

Algorithm: 2 of Fuzzy Control based Edge Resource Scheduling (FCERS)

```
Step 1.  Input: Resource Set $\{R_{Set_1}, R_{Set_2}, \ldots R_{Setm}\}$; Task Set $\{T_{Set_1}, T_{Set_2}, \ldots T_{Setn}\}$ $n, \varphi, m$
Step 2.  Output Ranking $rank$, similarity result $S$
Step 3.  On the resource, set Do clustering
Step 4.  $RConRS(M, L) = User$ & Re$source\_Clusters$
Step 5.  $R(s) = Size(R)$
Step 6.  While( Test < 1)
Step 7.  $Test = Test + 1$
Step 8.  $x^j = x$
Step 9.  Process $xn$
Step 10. $Computed\ t^E_{i,j,t}$
Step 11. $l=1$
Step 12. While $(l <= l)$ Do
Step 13. l=l+1
Step 14. j=j+1
Step 15. While $j = < m$ Do
Step 16. j=j+1
Step 17. Space $(k, i) = S_p$ $(Information\ (j, :)$
Step 18. End While
Step 19. End While
Step 20. Measure $x^J$
Step 21. If Result$(x^J - x$, Data$) < \varphi$ Then
Step 22. Break
Step 23. End If
Step 24. $x \leftarrow x^J$
Step 25. End While
Step 26. Obtain Resource Clusters
Step 27. Measure Rank (MS) = $REQ_I$, $R_{set}i$, $\omega_i$
Step 28. $MR = \dfrac{Total_{\text{Resourece}(\text{Re}sult(REO_1 - R_{set_i}))}}{Total_{Weight_i}}$
Step 29. Return Similarity Measure $S$
Step 30. End If
Step 31. End
```

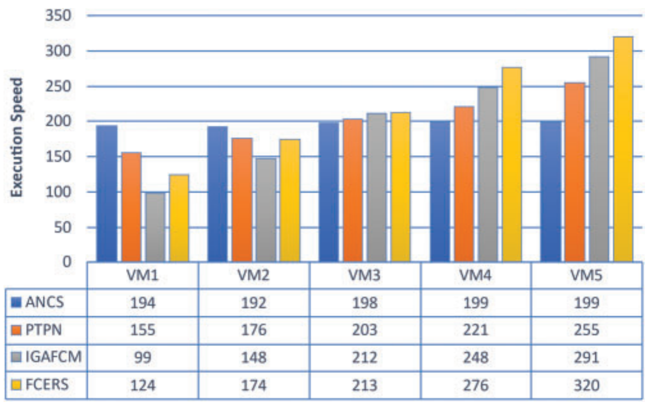
5 Experimental Outcomes for Fuzzy Control Based Edge Resource Scheduling

To assess FCERS's resource allocation efficiency, we evaluated several methods, including their distribution of resources methods. We assign tasks of varying lengths to distinct Virtual Machines (VMs). The proposed FCERS model is compared with three well-developed models presented in Tab. 1 below:

In various VM, the execution speed of the Central Processing Unit (CPU) is analyzed and given in Fig. 3. It shows that the proposed FCERS algorithm performs efficient RSP among corresponding VM based on the duration of tasks. When task durations are increased, the VM's execution speed increases. The trend is more apparent than other algorithms. Improved Genetic Algorithm for Fuzzy c-means Clustering Algorithm Method (IGAFCM) and PTPN cannot adapt fast to task duration as FCERS. Additionally, the Advanced Network Credit Scheduler (ANCS) has a limited ability to respond to task characteristics.

Table 1: Various models for RA

Model name	Proposed by authors
ANCS	Zhang et al. 2017
PTPN	Ni et al. 2017
IGAFCM	Hong-Qiang et al. 2016



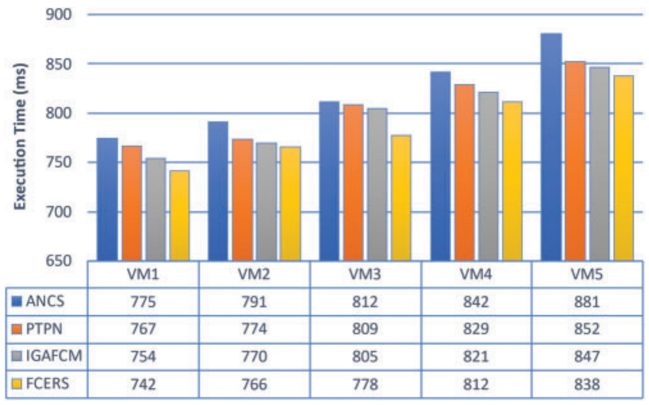
	VM1	VM2	VM3	VM4	VM5
🟦 ANCS	194	192	198	199	199
🟧 PTPN	155	176	203	221	255
⬜ IGAFCM	99	148	212	248	291
🟨 FCERS	124	174	213	276	320

Figure 3: CPU execution speed among different VMs

Fig. 4 illustrates the time required to complete a task using various techniques, and there are five VMs. The execution speed is lower in the ANCS, and the tasks are assigned to VM in an ordered fashion, resulting in resource shortages and insufficient utilization. The task execution efficiency is higher in PTPN, and the execution time is comparatively less when ANCS is used. However, tasks are not categorized, resulting in high-configuration VMs performing trivial activities and underutilized resources. When IGAFCM is used, each processor unit optimizes the available resources based on the task duration and overall execution time, decreasing the implementation latency between various VMs. When FCERS is used, process execution time is minimized. When FCERS is used, resource availability across VMs is constantly changed in response to task demand, execution time is effectively decreased, and the processing time-interval amongst processing units is considerably minimized.

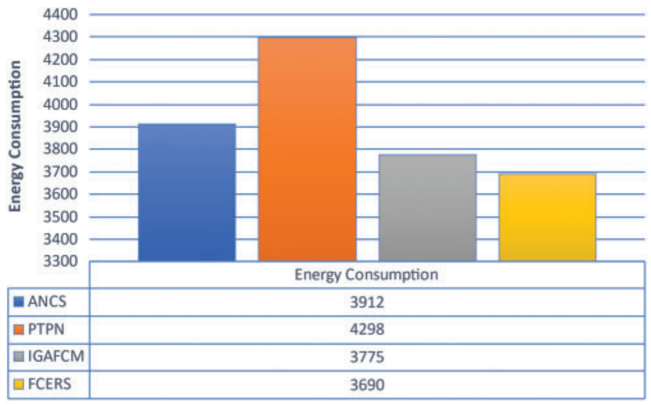
As seen in Fig. 5, the algorithm's overall energy consumption to accomplish all operations, including computation and transmission energy utilization, is primary since the algorithm completes tasks at the quickest. The lowest general energy usage is determined using CloudSim's built-in energy consumption model.

As illustrated in Fig. 6, the usage of memory by the proposed FCERS algorithm is comparable to that of the IGAFCM, PTPN and ANCS algorithms, completely fulfilling the memory space constraint for EC nodes. As a result, the proposed method may be implemented without compromising performance on EC nodes.



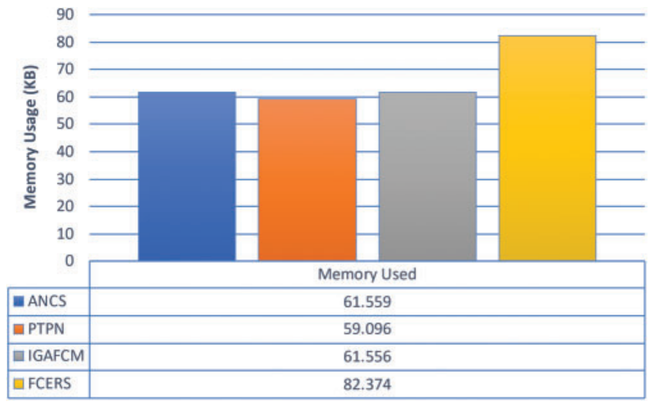
	VM1	VM2	VM3	VM4	VM5
■ ANCS	775	791	812	842	881
■ PTPN	767	774	809	829	852
■ IGAFCM	754	770	805	821	847
■ FCERS	742	766	778	812	838

Figure 4: Comparing task execution time for different VMs



	Energy Consumption
■ ANCS	3912
■ PTPN	4298
■ IGAFCM	3775
■ FCERS	3690

Figure 5: Total task energy consumption



	Memory Used
■ ANCS	61.559
■ PTPN	59.096
■ IGAFCM	61.556
■ FCERS	82.374

Figure 6: Memory usage

For Figs. 7–9, we consider alternative combinations of offloading strategies and their impacts on overall computation cost, time consumption, and energy consumption, all of which vary by the number of MDs used.

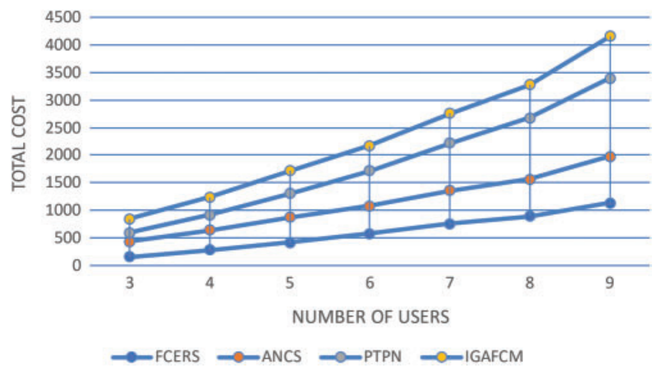


Figure 7: Total computation cost against increase in users

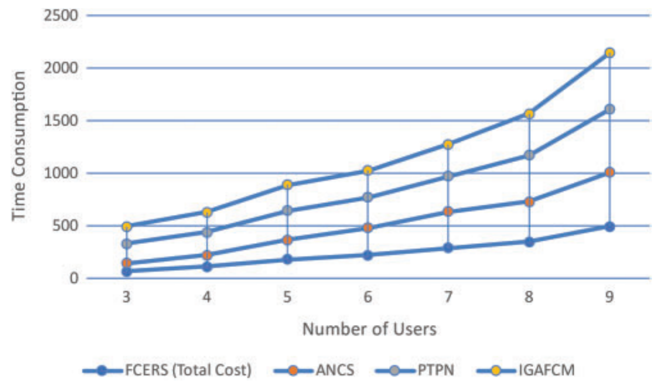


Figure 8: Time consumption against increase in users

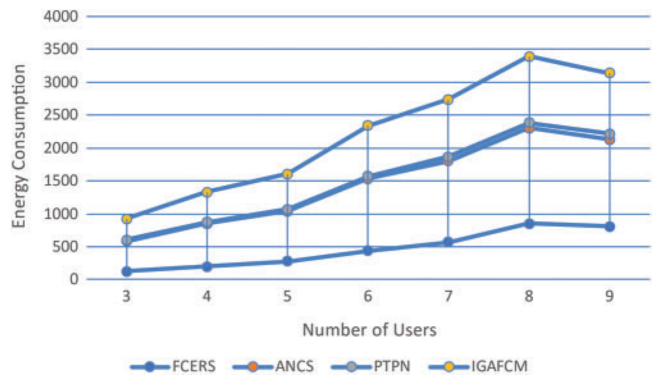


Figure 9: Energy consumption against number of users

To demonstrate, we evaluated two baselines. Our method reduces the total computing cost, execution time, and energy consumption on average compared to the baseline. The ES allocates higher processing resources to each user in case of limited availability of MDs; this results in improved task execution time and a reduced total task computation cost. As for EC, the energy consumption is ordered a few times or ten times that of local execution, which results in additional total computation costs.

6 Conclusion and Future Work

The purpose of this paper was to investigate the resource scheduling challenges in fog computing. First, we grouped the fog resources, which significantly reduces the range of user needs for resource matching. Additionally, we propose the Fuzzy Control based Edge Resource Scheduling algorithm for resource allocation. Finally, an experimental study revealed that the suggested FCERS method in this work converges quicker than the other methods. Additionally, the proposed FCERS algorithm may more efficiently fetch user requests to suitable resource categories, increasing user requirements.

In future studies, we will address dynamic resource changes and offer a novel RSP for optimizing resource use and ensuring user requirements.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] X. Chen and L. Wang, "Exploring fog computing-based adaptive vehicular data scheduling policies through a compositional formal method—PEPA," *IEEE Communications Letters*, vol. 21, no. 4, pp. 745–748, 2017.

[2] J. J. Durillo and R. Prodan, "Multi-objective workflow scheduling in Amazon EC2," *Cluster Computing*, vol. 17, no. 2, pp. 169–189, 2014.

[3] P. Gupta and S. P. Ghrera, "Trust and deadline aware scheduling algorithm for cloud infrastructure using ant colony optimization," "*Int. Conf. on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*," Noida, India, pp. 187–191, 2016.

[4] D. T. Nguyen, L. B. Le and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 302–317, 2021.

[5] D. Zeng, L. Gu, S. Guo, Z. Cheng and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, 2016.

[6] S. Wang, T. Zhao and S. Pang, "Task scheduling algorithm based on improved firework algorithm in fog computing," *IEEE Access*, vol. 8, pp. 32385–32394, 2020.

[7] Y. Wang, T. Uehara and R. Sasaki, "Fog computing: Issues and challenges in security and forensics," *IEEE 39th Annual Computer Software and Applications Conference*, Taichung, Taiwan, pp. 53–59, 2015.

[8] X. Q. Pham and E. N. Huh, "Towards task scheduling in a cloud-fog computing system," *18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Kanazawa, Japan, pp. 1–4, 2016.

[9] D. Poola, K. Ramamohanarao and R. Buyya, "Fault-tolerant workflow scheduling using spot instances on clouds," *Procedia Computer Science*, vol. 29, pp. 523–533, 2014.

[10] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222–235, 2104.

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

[12] Z. Li, B. Chang, S. Wang, A. Liu, F. Zeng *et al.,* "Dynamic compressive wide-band spectrum sensing based on channel energy reconstruction in cognitive internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2598–2607, 2018.

[13] X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang *et al.,* "Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 12, pp. 3501–3517, 2016.

[14] Y. Sun, T. Dang and J. Zhou, "User scheduling and cluster formation in fog computing-based radio access networks," in *IEEE Int. Conf. on Ubiquitous Wireless Broadband (ICUWB)*, Nanjing, China, pp. 1–4, 2016.

[15] W. Sun, J. Liu, Y. Yue and H. Zhang, "Double auction-based resource allocation for mobile edge computing in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4692–4701, 2018.

[16] L. M. Vaquero and L. Rodero Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[17] C. Li, J. Bai and J. Tang, "Joint optimization of data placement and scheduling for improving user experience in edge computing," *Journal of Parallel and Distributed Computing*, vol. 125, pp. 93–105, 2019.

[18] H. E. Zahaf, A. E. H. Benyamina, R. Olejnik and G. Lipari, "Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms," *Journal of Systems Architecture*, vol. 74, pp. 46–60, 2017.

[19] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu *et al.,* "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282–3299, 2020.

[20] M. Aazam, Z. Sherali and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities,*" Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.

[21] H. Zhang, J. Guo, L. Yang, X. Li and H. Ji, "Computation offloading considering fronthaul and backhaul in small-cell networks integrated with MEC," in *IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS)*, Atlanta, GA, USA, pp. 115–120, 2017.

[22] H. K. Apat, "An optimal task scheduling towards minimized cost and response time in fog computing infrastructure," in *Int. Conf. on Information Technology (ICIT)*, Bhubaneswar, pp. 160–165, 2019.

[23] W. Zhang, X. Li, L. Zhao, X. Yang, T. Liu *et al.,* "Service pricing and selection for IoT applications offloading in the multi-mobile edge computing systems," *IEEE Access*, vol. 8, pp. 153862–153871, 2020.

[24] A. Samanta, Z. Chang and Z. Han, "Latency-oblivious distributed task scheduling for mobile edge computing," in *IEEE Global Communications Conf. (GLOBECOM)*, Abu Dhabi, pp. 1–7, 2018.

[25] S. Josilo and G. Dan, "Decentralized scheduling for offloading of periodic tasks in mobile edge computing," in *IFIP Networking Conf. (IFIP Networking) and Workshops*, Zurich, Switzerland, pp. 1–9, 2018.

[26] G. Zhang, W. Zhang, Y. Cao, D. Li and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4642–4655, 2018.

[27] J. Zhang, X. Hu, Z. Ning, E. Ngai, L. Zhou *et al.,* "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, 2018.

[28] M. Luo, K. Zhang, L. Yao and X. Zhou, "Research on resources scheduling technology based on fuzzy clustering analysis," in *9th Int. Conf. on Fuzzy Systems and Knowledge Discovery*, Chongqing, China, pp. 152–155, 2012.

[29] Y. Fang, F. Wang and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing," in *Proc. of Lecture Notes in Computer Science*, Springer, Berlin, Germany, vol. 6318, pp. 271–277, 2010.

[30] K. A. Noghani, H. Ghazzai and A. Kassler, "A generic framework for task offloading in mmWave MEC backhaul networks," in *Proc. of IEEE Global Communications Conf. (GLOBECOM)*, Abu Dhabi, UAE, pp. 1–7, 2018.

[31] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros and G. Pavlou, "Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 475–488, 2018.

[32] R. Mamoon and W. Umer Iqbal, "Role of fog computing platform in analytics of internet of things-issues, challenges and opportunities," in *Fog, Edge, and Pervasive Computing in Intelligent IoT Driven Applications*, Hoboken: Wiley-IEEE Press, pp. 209–220, 2021.

[33] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[34] W. H. Qiang, L. X. Yong, F. B. Xing and W. Y. Ping, "Resource scheduling based on improved FCM algorithm for mobile cloud computing," in *22nd IEEE Int. Conf. on Parallel and Distributed Systems (ICPADS)*, Wuhan, China, pp. 128–132, 2016.

[35] L. Ni, J. Zhang, C. Jiang, C. Yan and Y. Kan, "Resource allocation strategy in fog computing based on priced timed petri nets," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1216–1228, 2017.

[36] J. Su, F. Lin, X. Zhou and X. Lu, "Steiner tree based optimal resource caching scheme in fog computing," *China Communications*, vol. 12, no. 8, pp. 161–8, 2015.

[37] D. Rahbari and M. Nickray, "Scheduling of fog networks with optimized knapsack by symbiotic organisms search," in *21st Conf. of Open Innovations Association (FRUCT)*, Helsinki, Finland, pp. 278–283, 2017.

[38] L. Guangshun, X. Shuzhen, W. Junhua and D. Heng, "Resource scheduling based on improved spectral clustering algorithm in edge computing," *Scientific Programming*, vol. 2018, Article ID 6860359, pp. 1–13, 2018.

[39] P. Qi and L. S. Li, "Task scheduling algorithm based on fuzzy quotient space theory in cloud environment," *Journal of Chinese Computer Systems*, vol. 34, no. 8, pp. 1793–1797, 2013.

[40] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[41] M. Mtshali, H. Kobo, S. Dlamini, M. Adigun and P. Mudali, "Multi-objective optimization approach for task scheduling in fog computing," in *Int. Conf. on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, Winterton, South Africa, pp. 1–6, 2019.

[42] 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 Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2017.

[43] W. Xiaojun, W. Yun, H. Zhe and D. Juan, "The research on resource scheduling based on fuzzy clustering in cloud computing," in *8th Int. Conf. on Intelligent Computation Technology and Automation (ICICTA)*, Nanchang, China, pp. 1025–1028, 2015.

[44] Y. H. Kao, B. Krishnamachari, M. R. Ra and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3056–3069, 2017.

[45] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 181–198, 2020.