Hindawi
Journal of Mathematics
Volume 2023, Article ID 9760327, 1 page
https://doi.org/10.1155/2023/9760327

Retraction

Retracted: A Computational Offloading Method for Edge Server Computing and Resource Allocation Management

Journal of Mathematics

Received 19 December 2023; Accepted 19 December 2023; Published 20 December 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope

(2) Discrepancies in the description of the research reported

(3) Discrepancies between the availability of data and the research described

(4) Inappropriate citations

(5) Incoherent, meaningless and/or irrelevant content included in the article

(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] M. Al-Razgan, T. Alfakih, and M. M. Hassan, "A Computational Offloading Method for Edge Server Computing and Resource Allocation Management," *Journal of Mathematics*, vol. 2021, Article ID 3557059, 11 pages, 2021.

Hindawi
Journal of Mathematics
Volume 2021, Article ID 3557059, 11 pages
https://doi.org/10.1155/2021/3557059

Research Article

A Computational Offloading Method for Edge Server Computing and Resource Allocation Management

Muna Al-Razgan,[1] Taha Alfakih ,[2] and Mohammad Mehedi Hassan[2]

[1]*Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11345, Saudi Arabia*
[2]*Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11345, Saudi Arabia*

Correspondence should be addressed to Taha Alfakih; talfakih@ksu.edu.sa

Received 24 October 2021; Accepted 22 November 2021; Published 21 December 2021

Academic Editor: Naeem Jan

The emerging technology of mobile cloud is introduced to overcome the constraints of mobile devices. We can achieve that by offloading resource intensive applications to remote cloud-based data centers. For the remote computing solution, mobile devices (MDs) experience higher response time and delay of the network, which negatively affects the real-time mobile user applications. In this study, we proposed a model to evaluate the efficiency of the close-end network computation offloading in MEC. This model helps in choosing the adjacent edge server from the surrounding edge servers. This helps to minimize the latency and increase the response time. To do so, we use a decision rule based Heuristic Virtual Value (HVV). The HVV is a mapping function based on the features of the edge server like the workload and performance. Furthermore, we propose availability of a virtual machine resource algorithm (AVM) based on the availability of VM in edge cloud servers for efficient resource allocation and task scheduling. The results of experiment simulation show that the proposed model can meet the response time requirements of different real-time services, improve the performance, and minimize the consumption of MD energy and the resource utilization.

1. Introduction

There are more issues facing mobile devices such as smart phones, reconnaissance planes (drones), robots, patient monitoring devices, and wireless sensors, due to limited specifications of these devices (like storage, memory, CPU, battery), and intensive applications (e.g., real-time translation, video processing, image processing) require supercomputing. Mobile devices with limited resources are not efficient or opportune to process those applications. Therefore, mobile cloud computing (MCC) emerged as a solution to overcome the resource limitation of mobile devices by using computation offloading.

Computation offloading is a transfer mechanism of software application processes from limited resource devices to resource-rich platforms. Mobile cloud is the well-known module for MD computation offloading. MCC is becoming a popular technique for mobile services (MS), e.g., video streaming, mobile video games, social networking, education, mobile healthcare services, and instant messaging [1].

Wireless communication limitations like disconnection, low bandwidth, security issues, latency, and mobility are considered the key barriers to offloading computation in cloud computing (CC) [2]. Real-time services are extremely latency sensitive and thus require computing data in close proximity to MDs. Therefore, a proximity cloud like MEC, fog computing, and cloudlet can be an efficient and appropriate module for computation offloading.

Edge computing (EC) is an emerging technique, which is currently being standardized in ETSI Industry Specification Group (ISG). EC offers a service of IT environment and capabilities of CC at the mobile network edge, within the Radio Access Network (RAN), and in close proximity to mobile users (MU). The aims are to minimize the services

delay, confirm effective network operation and service delivery, and offer an enhanced user experience [3]. There are some challenges facing the MEC [4], like the service synchronization and orchestration between the cloud servers and edge server (ES) (the central cloud), in addition to "seamless service delivery" as the connectivity at the EC infrastructure may be intermittent with mobility. Additionally, the standard IP based operations become infeasible to address the interactions between MD and servers, considering that the problem is more emphasized in the services of edge computing facility. The other challenges to the services of edge computing may not always assume the availability of the local infrastructure.

MEC is a distributed system that provides features such as low latency, distributed analytics, real-time interaction, geographical distribution, mobility support, and context awareness, which are not supported by centralized CC paradigm [5].

We considered the mobile devices aiming to perform own intensive applications interacting with many edge cloud computing infrastructures, through different wireless connections or Wi-Fi, in order to distribute the computation load to one or more edge computing infrastructures (MEC, cloudlet, and remote cloud) and receive data results, to perform the computation offloading [6]. MEC, e.g., computation offloading of heavy tasks to EC in the communication network (or cellular network), has become a promising technique for the next generation of the cellular networks. Therefore, there is a need for enhancing a model of distributed computation offload, so that not only the computational servers are applied at their superlative capacity, but also the response time of user's restrictions is achieved [7].

The essential components of the computation offloading model are a client running on the client device and a server running on the edge server, regardless of the surrounding devices or remote cloud. The component of client has three main functions: firstly, monitoring and predicting the performance of the MD network; secondly, predicting and tracking the requirements of computation of client services in terms of input and output requirements of data and time of execution on both the client and the ES; thirdly, using such information to select some partition of the computation to perform in the cloud so that the total execution time is reduced. The components of the server immediately execute offloaded portions after receiving them and return back the results to the client components, so the application can be continued on the mobile device. Our objective in this paper is to reduce latency of the processes (i.e., offloading the process and receiving the results) by proposing an evaluation model to evaluate the efficiency of near-end network computation offloading in mobile edge computing (MEC). This model helps in choosing the adjacent edge server from the surrounding edge servers. This helps to reduce the latency and increase the response time. To do so, we use a decision rule based Heuristic Virtual Value (HVV). The HVV is a mapping function dependent on the feature of the edge server like the performance and workload. Additionally, in this paper, we propose a VM resource availability algorithm (AVM) that optimizes resource allocation and task assignment based on VM availability in the edge cloud servers. The simulation results show that the proposed model satisfies the requirements of response time of real-time applications, improves the performance, and minimizes the MD power consumption.

The rest of the paper is organized as follows: In Section 2, we present the related work review. In Section 3, we outline the computation offloading in MEC. In Section 4, resource allocation model is given. Then, in Section 5, we describe our proposed model results and analysis, followed by the conclusion in Section 6.

2. Related Work

In [8], the performance ability or capability is enhanced to leverage the available computing of edge servers and capacity, by the proposing a system which provides a collection of colocated devices as cloud service at the edge and enables leveraging multiple clients into a coordinated cloud computing service despite churn in participation of mobile devices. On the other hand, the study in [9] proposed a framework and a model used a holistic approach to bond context adaptation and computation offloading (cloud aware), to help app's developers to design scalable and flexible edge depending on mobile applications.

The cloud RAN (C-RAN) service is exploited to propose a decision algorithm of offloading that makes decisions about computation offloading from the client to the cloud remote radio heads (RRHs), to save power consumption and keep a satisfying user QoE by reducing app's time of response [10]. Lyapunov optimization algorithm is proposed to take the decisions of the offloading, frequencies of the cycle of CPU for mobile application execution, and energy of the communication for computation offloading [11].

A scheme of opportunistic computation offloading for the MECC systems is provided to execute tasks of data mining in client devices and edge servers (ES), to reduce time of execution and energy consumption [12]. In [13], a distributed computation offloading model that can achieve a Nash equilibrium is proposed to fulfil the superior performance of computation offloading and scale well as the user size increases. In study [7], the authors used minority games theory to analyze the statistical characteristics of the offloading delay for the users' requests and channel quality by using a distributed algorithm to solve efficient servers selection issues. In [14], the authors assess a performance and investigate the computation offloading from the MD's to the small cell cloud (SCC). In the scenario of a cooperative MEC server [15], the authors analyze the problem of joint task offloading and resource distribution. Furthermore, IoT resource fairness should begin by taking the user experience into account. Additionally, the authors propose a two-level heuristic: the first, inspired by evolutionary algorithms, searches for superior offloading schemes globally; the second, considering fairness among all tasks, generates resource allocation modules, making use of the server's resources as efficiently as possible [15].

The proposed method is formatted as an optimization problem to reduce the consumption of MD energy, due the overhead of the MEC capacity. The priority of offloading for each device depends on its power consumption and channel gain. Complete offloading is performed for a high priority, while minimum offloading is performed for a low priority [16].

A modern framework for computation offloading from a mobile device as a client to an edge server as a host, with availability of highest CPU, is presented. The main idea is to estimate RTT value between mobile device and edge server according to signal quality of the RAN as an application programming interface (API) to make the mobile device decision to offload or not computing tasks for application. Additionally, a novel algorithm of computation is proposed; it depends on the estimated RTT connected with other parameters (e.g., consumption of power) to take a decision as to when to offload application computation tasks of mobile device to the mobile edge computing server [17].

Proposing computation offloading in a multicell system, the authors considered multiple users requests with multiple inputs and multiple outputs (MIMO). Additionally, the researchers expressed the problem using the radio joint optimization and presence of the intercellular interference with the resources of computational for computation offloading in an applications intensive deployment [18]. The preceding research works did not deal with the process of evaluation pre-decision to the unloading process. When used, this technology needs to be handled with caution, and more research should be done on such issues such as confidentiality, authenticity, and integrity [19]. In this study, an evaluation model is proposed to evaluate the efficiency of near-end network computation offloading in MEC.

Max–Min algorithm [20] is one of the popular, very simple, and easy to implement cloud scheduling algorithms, because it has very few control variables, and we use it in the edge cloud computing. All the small tasks are allocated to faster resources, and large tasks are allocated to slower resources. Hence, it minimizes the waiting time average of shorter jobs, by assigning them to faster resources, and our large tasks are executed by slower resources. Therefore, this algorithm improves simultaneous implementation of tasks on resources. Therefore, algorithm of Minimum Completion Time (MCT) allocates offloaded tasks to resources or VMs based on the best expected completion time for these tasks in random order. Each task is allocated to the resource or VM that has the closest completion time. In the MCT algorithm, some tasks are assigned to resources or VMs that do not have minimum processing time [21].

Round Robin algorithm (RR) [22] is considered one of the easiest process scheduling algorithms. It gives equal time for each process, dealing with all processes without priority for any of them. The RR scheduling is characterized by its simplicity and ease of implementation, as well as being free from starvation, which means that the process does not have the resources needed to complete it at all or after a long period.

In the study, the vertical collaboration of mobile devices, mobile edge servers, and mobile cloud servers, as well as the horizontal cooperation of edge nodes in computation, is examined in relation to cooperative computation task offloading and resource assignment in the MEC. It is formulated to make decisions regarding computation offloading, cooperative selection, power allocation, and CPU cycle frequency assignment. In order to minimize latency, energy consumption, transmission power, and CPU cycle frequency should be reasonably constrained [23].

3. Computation Offloading in ES

The computation offloading is the technique used to raise the performance of the mobility of devices and minimize the power consumption by offloading the intensive tasks which need urgent and accurate processing to the remote resources that are able to compute these tasks and return the results immediately, for example, real-time and intensive applications like image processing.

The major objective of our research is the concurrent offloading of the offloadable processes (partitioned tasks) to proximate resources EC (L-ESs) as shown in Figure 1, to make the appropriate decision based on the results of the profile examination of each resource (i.e., L-ESs, ECC, or RC). In other words, we decide the computation offloading of the process to EC resources (L-ESs, ECC, and remote cloud), mobility management, and effective allocation of the computation resources, for exploiting the moving users within each resource.

3.1. Edge Server Computing Architecture. Figure 2 portrays the high-level overview of the edge server and the mobile devices, such as smart phones, tablets, robots, and drones. The base station communicates with the network of the local ES before entering the Internet. At the ES, the components of network having super computation and storage capabilities are collected to create a virtual server (VS) offering mobile ES. If a workload demands resources beyond what the ES can support, the request can be redirected over the core network until reaching the cloud services on the other side of the network [24].

ME is the complementary edge computing model especially prototyped for unified and latency-aware MS. The offloading decision depends on the following parameters: performance, energy consumption, latency, and cost. The proposed model automatically selects the computing source based on the following parameters: performance, signal strength, radio bandwidth, and workload. We relied on these criteria to choose the suitable edge server.

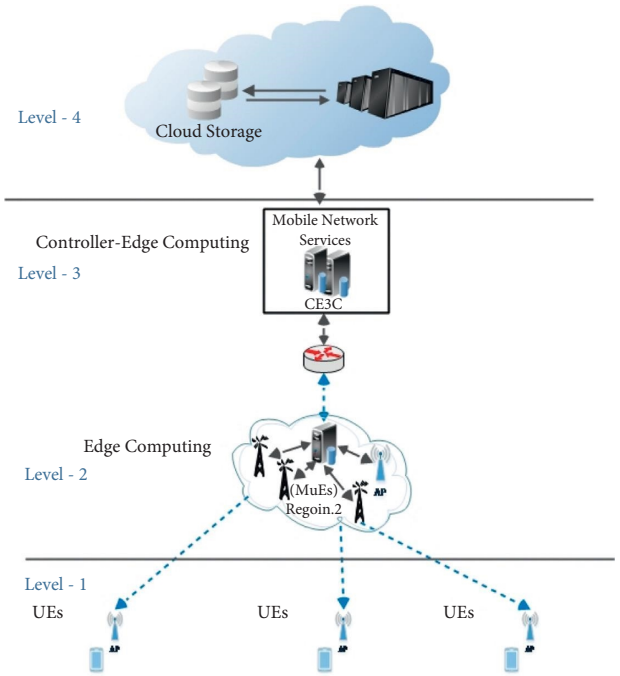

Figure 1: A high-level overview of edge server computing.

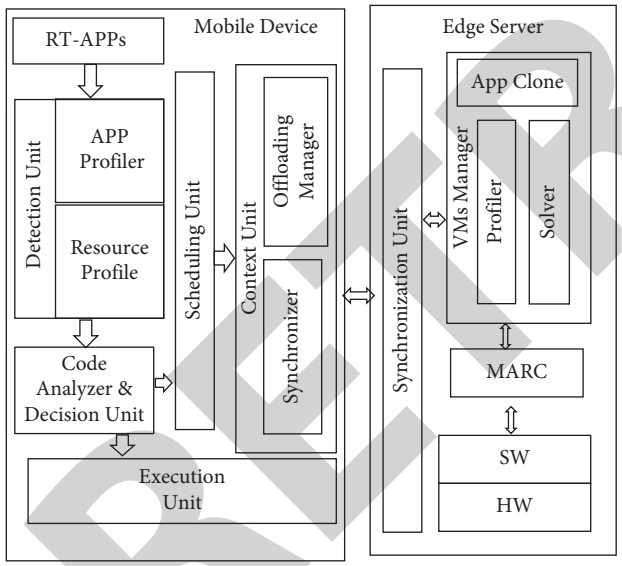

Figure 2: Mobile edge computing system architecture.

Figure 2 illustrates the MECS architecture. The MECS on the edge server consists of the synchronization unit, which is responsible for maintaining the synchronization between mobile devices and edge server; management allocation resource control (MARC) unit; and virtual management unit, which consists of profiler to monitor the VM workload and edge server performance. Additionally, the solver executes the task or the application clone.

MECS architecture on the mobile device consists of five units: the detection unit detects the tasks and the available resources; the code analyzer divides the task into subtasks and determines whether a subtask is offloadable or nonoffloadable; the scheduling unit places subtasks in the waiting queue if they are offloadable and assigns them to local execution otherwise; the context unit is responsible for synchronization between the client and edge server and management of offloading to distribute the subtasks.

3.2. Offloading Decision Rule. Since computation offloading migrates the intensive tasks to more resourceful computing, it involves decision making as to whether and what computation should be migrated. The decisions of offloading to remote resources are divided into improving performance and saving energy [25]. However, other problems emerged, especially for the sensitive computational tasks, including latency, mobility, bandwidth bottleneck, resource management, privacy, and security.

We consider an ECC consisting of a pool of edge servers (L-ES) denoted by **s** that is connected through the LTE and AP, edge server control (ESC) denoted by S, and a set of mobile devices (MDs) denoted by M in each zone Z. Each mobile device has some sensitive computational processes $\mathscr{C}$ to be accomplished in successive periods (time) $\mathscr{T}$. Each process P may be partitioned into several tasks c, according to the tasks performed by this process. T is the time consumed to compute the task on the MD, while T' is the time consumed to complete task on the edge server. The consumed energy to compute the task on the MDs is denoted by E, and the consumed energy to process the task on the edge server is denoted by E'.The workload for each edge server is denoted by $\mathbf{s_W}$.

The execution time of the task on the edge server (**S**) is

$$T = \frac{\mathscr{C}}{M_p}, \tag{1}$$

where $\mathbf{M_p}$ the performance of the mobile device and $\mathbf{s_p}$ represents the performance of the edge server. The offloading process will improve the response time in processing tasks, as it reduces the response time due to the ES close to the user. The total time of responsiveness is calculated using the following:

$$\text{Total time} = \text{time of communication} + \text{time of computation},$$

$$T' = \frac{c_i}{b} + \frac{\mathscr{C}}{s_p}, \tag{2}$$

The offloading performance (latency) improves if the following condition holds:

$$\begin{aligned} T > T' &\Rightarrow \frac{\mathscr{C}}{M_p} > \left(\frac{c_i}{b} + \frac{\mathscr{C}}{s_p}\right) \\ &\Rightarrow \mathscr{C}\left(\frac{1}{M_p} - \frac{1}{s_p}\right) > \frac{c_i}{b}, \end{aligned} \tag{3}$$

where c_i indicate the size of task that needs to be sent and b is the bandwidth.

The consumed energy to execute the task within the mobile device system is given by

$$E = e_m \times \frac{\mathscr{C}}{M_p}, \tag{4}$$

where e_m represents the energy on the mobile system. The total consumed power considering the computation and transmission is determined by

$$E' = e_w \times \frac{c_i}{b} + e_c \times \frac{\mathscr{C}}{s_p}, \tag{5}$$

where e_c indicates the power required for communication between mobile device and server edge over the network, e_w refers to the power required to wait for the result, c_i is the size of task that needs to be offloaded, and b is the bandwidth. The offloading power is saved if the following condition holds:

$$\begin{gathered} E > E' \Rightarrow e_m * \frac{\mathscr{C}}{M_p} > e_w * \frac{c_i}{b} + e_c * \frac{\mathscr{C}}{s_p}, \\ \mathscr{C}\left(\frac{e_m}{M_p} - \frac{e_c}{s_p}\right) > e_w * \frac{c_i}{b}. \end{gathered} \tag{6}$$

Saving offloading power will be computation process $\mathscr{C}$, and light communication of the each task c_i is considered.

3.3. Decision Rule. We have several edge servers and we need to offload tasks to one of them (where). In other words, we need to choose the optimal edge to compute the offloadable task depending on our proposed Heuristic Virtual Value (HVV) and signal strength (s_{ss}) of the edge servers. The HVV is a mapping function dependent on the features of the edge server like the performance and workload. Each server independently decides whether to be in active mode (to accept computation task, $s_M = 1$) or inactive mode (not to accept computation task, $s_M = 0$). Therefore, we proposed the Heuristic Virtual Value (HVV) for each server, and we need to calculate it. The HVV depends on some parameters (like performance $\mathbf{s}_\mathrm{p}$, workload s_W, mode $\mathbf{s_M}$) for each edge server $\mathbf{s} \in \mathbf{S}$. The edge server depends on the HVV to connect to the mobile devices. Let $\mathbf{s_i}(\mathbf{t})$ be the server that decides to be ready to receive the task at the time $\mathbf{t}$ if the server mode is active ($\mathbf{s_M} = 1$), so that the HVV for each server is given by

$$\mathrm{HVV}_{s_i}(t) = \mathscr{F}\left[s_p, s_W\right]. \tag{7}$$

The $\mathbf{HVV}_{\mathbf{s_i}}(\mathbf{t})$ is a mapping function that receives the values of $\mathbf{s_p}$ edge server performance and $\mathbf{s_w}$ edge server workload as percentage and converts them into categorical values between one and four as shown in Table 1.

$$\begin{gathered} \mathscr{F}\left[s_p, s_W\right] = \frac{S_p}{S_w}, \\ \mathrm{HVV}_{s_i}(t) = \frac{S_p}{S_w}, \\ S_M = \begin{cases} 0, & 0 < \mathrm{HVV}_{s_i}(t) < 1, \\ 1, & \text{Otherwise.} \end{cases} \end{gathered} \tag{8}$$

The resulting potential values of such HVV (t) function receive the result of division edge server performance $\mathbf{s_p}$ over edge server workload $\mathbf{s_W}$. The edge server mode $\mathbf{s_M}$ depends on the value of the $\mathbf{HVV}_{\mathbf{s_i}}(t)$; for instance, in the best cases the value of $\mathbf{HVV}_{\mathbf{s_i}}(t)$ is 4, which makes the edge server in active mode $\mathbf{s_M} = 1$; in contrast, in the worst cases the $\mathbf{HVV}_{\mathbf{s_i}}(t)$ is 25, which makes the edge server in inactive mode $\mathbf{s_M} = 0$, as illustrated in Algorithm 1.

Figure 3 illustrates the processes of the computation offloading; before making a computation offloading decision, each client detects the signals for each surrounding edge server and makes an ordered list of signal strength degrees for edge servers. The client receives the strength of signal as a percentage and converts it into a categorical value between one and four like the values of $\mathbf{s_p}$ as shown in Table 1. The client selects the edge server that has a maximum S_{ss} and starts to communicate with it by sending acknowledgment ack (t). $\mathbf{T_n}$ represents the threshold value for the offloadable task. If the edge server receives the ack(t), the server calculates the $\mathrm{HVV}_{s_i}(t)$; if $\mathbf{s_M} = 1$, the edge server replies to the client that it is active and assigns each mobile device a unique ID. Then, the client starts to offload the task $c(t)$ to the edge server and receive the result.

In contrast, if the edge server returns $S_M = 0$, this means that the server is unable to serve the client and meet time requirements; in this situation, the client finds an alternative server by checking the order list $[S_{ss}]$, choosing the Next Max$[S_{ss}]$, starting to communicate with it, and so on. Otherwise, it is the client. If the client fails to communicate with the alternative server, they have to communicate with the remote cloud.

4. Resource Allocation Management in the MEC Based on VM Resource Availability

Figure 4 illustrates the system model, which consists of the following units: The monitoring and migration unit as a core unit is responsible for the resource management of ES. The profiling module unit works to acquire the tasks features and its computing requirements to compare them with the ES capabilities and determine whether it can serve them. Then, the VMs control unit works to create VMs according to the offloaded tasks' computation requirements. The scheduling unit distributes these tasks to the available VMs. In the UE mobility, if the UE is out of coverage service of ES, the aggregation unit, in collaboration with the other units in the system model, distributes the tasks to adjacent ES, as well as balancing the resources between VMs.

4.1. VM Availability Evaluation. VM availability directly impacts the scheduling tasks in the edge cloud platform; therefore, the evaluation of VMs availability is considered when managing task scheduling. Additionally, VM resource availability is the capacity of providing functional services within required time after the task is offloaded to VMs.

We can measure the capacity of available task processing $\mathbf{q}_{ij}$ of VMs using completion time (τ), completion rate (t_c), and task arrival rate (t_r). We calculate the capacity of VMs

Table 1: Values of server's performance and workload.

Range (%)	Mapping value (s_P, s_W)
0–25	1
26–50	2
51–75	3
76–100	4

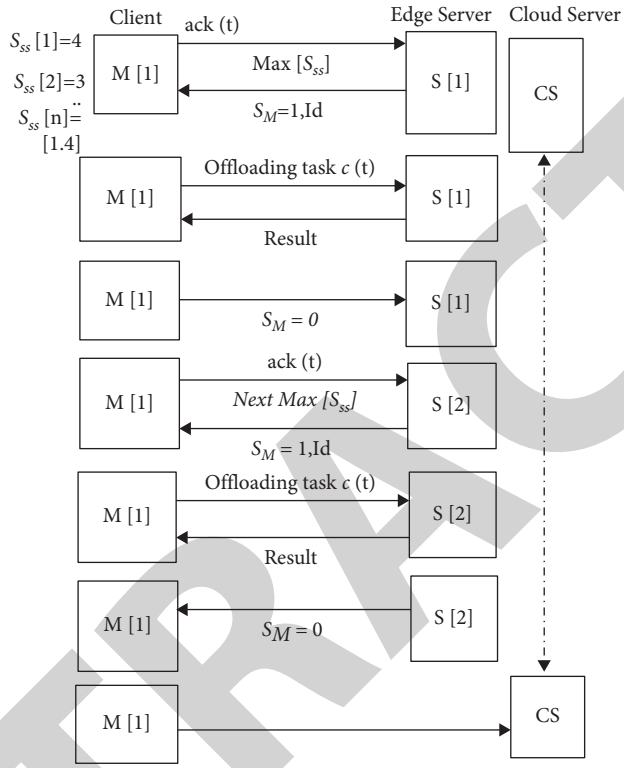

Figure 3: Processes of the computation offloading to the edge server.

(1) Input: s_P, s_W
(2) Output: s_M
(3) Begin
    //Mapping s_P, and s_W into their corresponding category.
(4) if s_P & s_W, in [0% –25%] then
(5) $s_P = 1$; $s_W = 1$;
(6) elseif s_P & s_W, in [26% –50%] then
(7) $s_P = 2$; $s_W = 2$
(8) elseif s_P, in [51% –75%] then
(9) $s_P = 3$; $s_W = 3$;
(10) else
(11) $s_P = 4$; $s_W = 4$;
(12) End if
(13) $\mathrm{HVV}_{s_i}(t) = S_p/S_w$
(14) if $(0 < \mathrm{HVV}_{s_i}(t) < 1)$ then
(15) $s_M = 0$;
(16) else
(17) $s_M = 1$;
(18) Return s_M
(19) End

Algorithm 1: Heuristic Virtual Value algorithm-based decision rule.

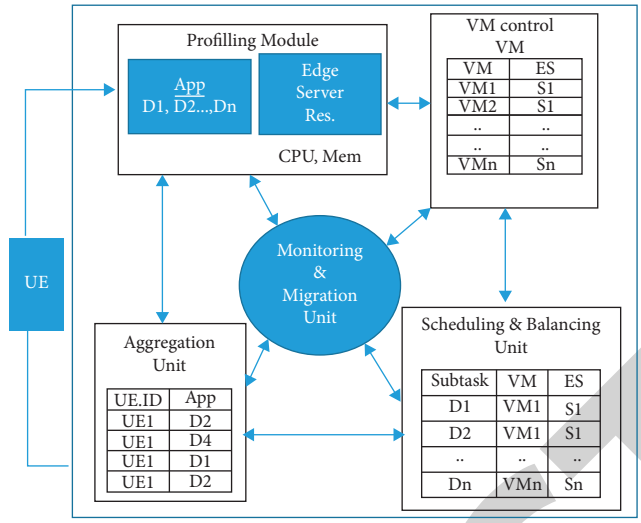

Figure 4: System model of resource allocation management.

available to task processing (q_{ij}^{V}) of the VM_j in the S_i edge server after receiving the task $\{c_i, i = 1, \ldots, n\}$, as follows:

$$q_{ij}^{V} = \frac{P_{ij}\alpha}{t_c}, \tag{9}$$

where P_{ij} is the probability that the VM_j in the S_i edge server receives a task and α is rate of task arrival in the edge cloud platform.

Task completion time τ refers to the difference between task time of arrival t_r and task completion rate t_c, namely,

$$\tau = t_r - t_c. \tag{10}$$

Therefore, the capacity of available task processing q_{ij}^{S} of each S_i edge server is the sum of processing capacities of all VMs in the edge server.

$$q_{ij}^{S} = \frac{\sum_{j=1}^{n} P_{ij}\alpha}{t_{cij}}. \tag{11}$$

The available capacity of S to tasks processing ($\mathbf{q}_{ij}$) of the VM_j in the S_i edge server gives the model strength and ability to compute the most significant number of tasks. Therefore, we can calculate the resource utilization ($\mathcal{R}_u$) as follows:

$$\mathcal{R}_u = \frac{q_{ij}^{S}}{q_{ij}^{V}}, \quad \text{where } q_{ij}^{V} \subset q_{ij}^{S}. \tag{12}$$

4.2. Scheduling Algorithm. We assume that the task requirements are already acquired. We measure each VM workload (V_W) and workload of edge server (S_W) as follows:

$$V_W = \frac{f_{Vi}}{T}, \tag{13}$$

where

$$S_W = \sum_{i=1}^{n} V_W \quad , \forall V_i \in S, \tag{14}$$

where T is the execution time of the task on the edge server ($\mathbf{S}$).

Measurement of the VM availability and allocation of VMs to accommodate the offloaded tasks constitute one of the most important issues that edge servers face in order to be able to schedule tasks for computation and meet time requirements [26]. We proposed a task scheduling algorithm based on the availability of VMs (abbreviated to AVM), Algorithm 2, and their dynamic evaluation. The choice is made among the most available resources, which avoids slowing down the tasks computing. The module of task scheduler manages the scheduling of tasks based on requests of task resources $\{c_i, i = 1, \ldots, n\}$ The purpose of the task scheduling is to maximize available differential of VMs, and VMs with relatively small workloads on the edge server where they are located are selected, which can improve task computation time as shown in Algorithm 2.

5. Simulation and Result Analysis

This study used the Cloudsim tool [27] to evaluate the proposed decision rule, VM resources availability, and resource allocation management. The CloudSim is an open-source package that is obtainable for public use. In this section, we will verify two aspects: (1) offloading decision making; (2) resource allocation management and task scheduling based on availability of VM.

5.1. Offloading Decision Making. There are some factors that influence offloading decision making: First (when to offload), predict execution time and power consumption in the mobile devices and remote cloud and compare them. If the e_m, $t_m > e_c, t_c$, the offloading is active, where e_m indicates the mobile energy consumption, t_m mobile time execution, e_c cloud energy consumption, and c_m cloud time execution.

(1) **Input**: A set of tasks $\{c_i, i = 1, \ldots, n\}$, $S = \{S_1, S_2, \ldots, S_n\}$, and VM = $\{v_1, v_2, \ldots, v_n\}$
(2) **Output**: Scheduling tasks and resource allocation
(3) Compute available of the VM for computing capabilities to meet processing time requirement based on (9)
(4) Initialize VM available capabilities of task processing
(5) for all tasks $\{c_i, i = 1, \ldots, n\}$ do
(6) for all VMs = $\{v_1, v_2, \ldots, v_n\}$ do
(7) Assigned tasks c_i into v_i
(8) Compute v_i available task processing capabilities based on (9)
(9) end for
(10) for all tasks $\{c_i, i = n + 1, \ldots\}$ do
(11) Assigned tasks c_i into v_i
(12) add a new VM
(13) end for

Algorithm 2: Task scheduling algorithm of VM available differential maximization (AVM).

Second, the decision depends on network features like the bandwidth and signal strength. Third (where to offload and which edge server is selected), selecting the appropriate server depends on the performance and workload of the edge server. In this research, we focus on the third factor. We suppose that we have offloadable task, it is divided into subtasks, and some of the subtasks (blocks) are offloadable but the others are executed locally. Additionally, we suppose that the feature of the network is appropriate to offloading. We suppose that we have several edge servers and we need to offload tasks to one of them (where to offload). Each server independently decides whether to be in active mode (to accept computation task $s_M = 1$) or in inactive mode (not to accept computation task $s_M = 0$) based on the Heuristic Virtual Value (HVV) for each server. The edge server depends on the HVV to connect to the mobile devices. Let $\mathbf{s_i}$ (**t**) be the server that decides to be ready to receive the task at the time **t** if the server mode is active ($\mathbf{s_M} = 1$); otherwise, the server mode is inactive ($\mathbf{s_M} = 0$).

We can compare the performance of the sample topologies in many aspects, such as the service time and power consumption. However, we want to highlight the results, which can be only provided by our simulation. Figure 5 shows that the average of service time (performance) is shown with respect to the number of offloadable subtasks (blocks). In the simulation, we note that the execution time when offloading subtasks to the edge server in case it is active is much less than that in case it is inactive, assuming that the offloading processes are done whether the edge server is active or inactive.

Figure 6 shows the average of power consumption with respect to the number of offloadable subtasks (blocks). We notice that the power consumption of subtasks when offloading is active is much less than that when offloading process is inactive, under the assumption that offloading processes are done whether the edge server is active or inactive.

5.2. Task Scheduling Based on VM Availability. To evaluate the effectiveness of the algorithm, we configure the simulation environment setting by computer configuration as follows: CPU: 1,500–3,000 MIPS, octa-core; memory: 32,000–64,000 MB; storage: 2 TB. Experimental parameters are set as follows: 2 edge servers, 40 VMs, range of offloaded tasks from 20 to 300. We simulated many offloaded tasks using LCG data for VMs in each ES, assuming that the ES provides edge cloud services to the VMs [28].

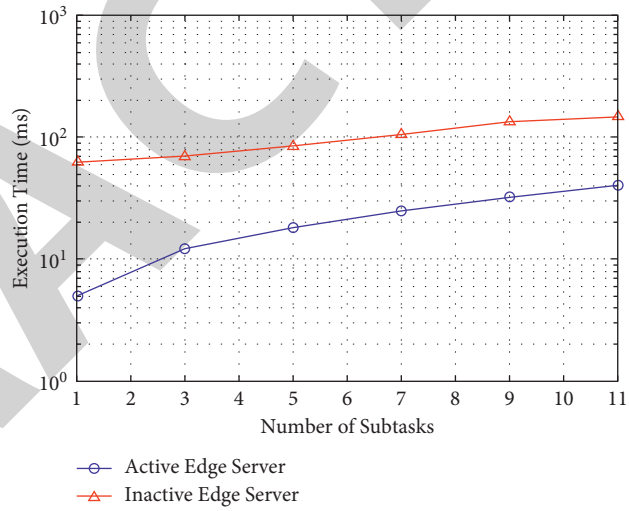

Figure 5: Average of execution time.

To evaluate the proposed algorithm AVM, we compare its results with other scheduling algorithms, namely, Round Robin (RR) and Minimum Completion Time (MCT) algorithm. We adopted three important parameters for comparison, which are awaiting time, resource utilization, and response time. The experimental results show that the AVM is superior to the other two algorithms. In the experiment, we measure the response time in milliseconds. Figure 7 illustrates the differential of the response time of each task. The number of tasks ranges between 20 and 300. Therefore, the results of the AVM are better than those of the other algorithms, as AVM improved the response time for each task.

In Figure 8, we measure the resource utilization by calculating the amount of resources remaining after serving

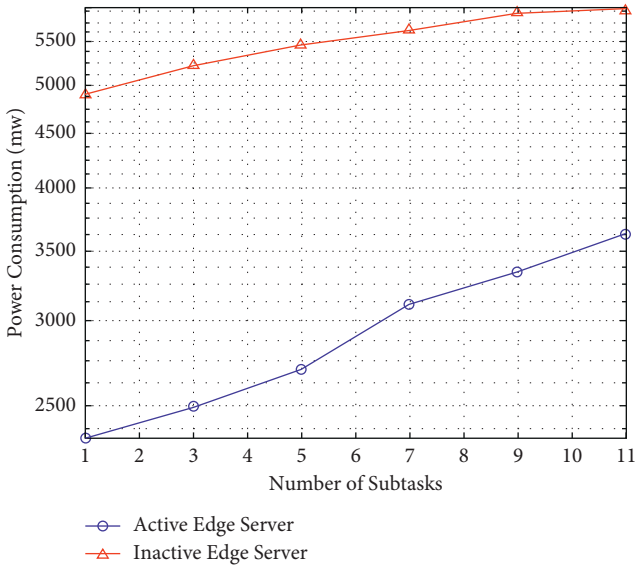

Figure 6: Average of power consumption.

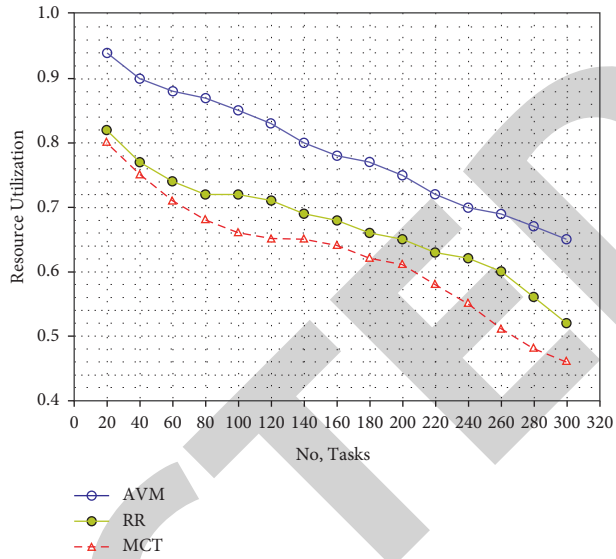

Figure 8: VM resource utilization.

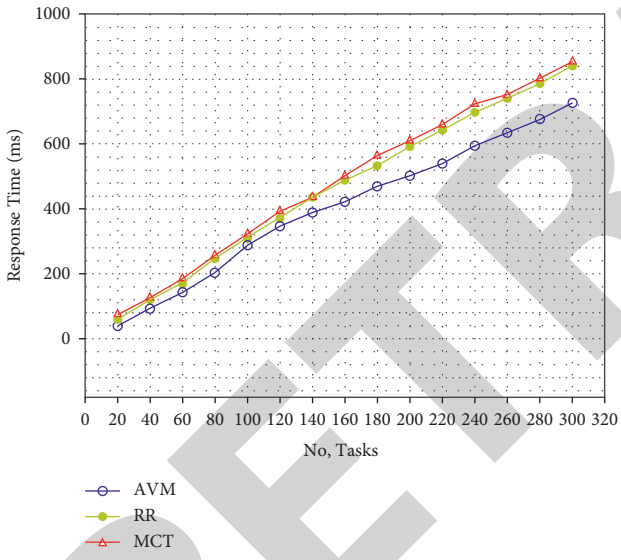

Figure 7: Response time per task.

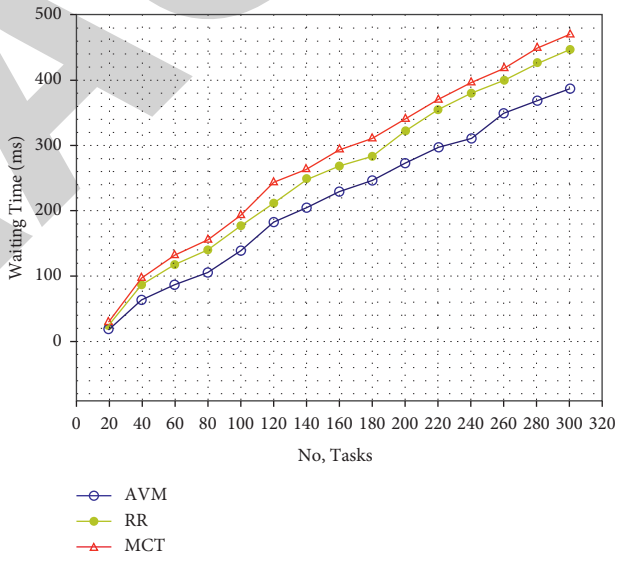

Figure 9: Task waiting time.

the tasks; for example, in our experiment, based on (14), the percentage of the remaining resources is measured.

Average waiting time is measured by computing the difference between the task offloaded time to the edge server and the starting time of the execution of the task. Figure 9 shows that the waiting time when applying the proposed algorithm was less compared to the other algorithms. Figure 9 depicts the average waiting time depending on the number of tasks when the number of VMs ranges between 20 and 300. The waiting time was affected by the number of offloaded tasks at the edge server (ES), as the waiting time gradually increased based on the number of tasks.

6. Conclusions

The edge computing technology promises an opportunity to overcome the constraints of mobile devices by offloading resource intensive and time sensitive applications to a nearby server. Until now, the concept of edge computing is not standardized, and it is hard to develop different and various architectures or scenarios of application. In this research, we proposed an evaluation module to evaluate the efficiency of the close-end network computation offloading in MEC. This model helps in choosing the adjacent edge server from the surrounding edge servers. This helps to decrease the latency and increase the response time. To do

so, we use a decision rule based Heuristic Virtual Value (HVV). The HVV is a mapping function based on the features of the ES like the performance and workload. Additionally, in this study, we proposed AVM algorithm to address the resource balancing, resource allocation, and task scheduling. The results of simulation show that the proposed model satisfies the latency requirements of time sensitive applications, enhances the performance, and minimizes the energy consumption of mobile devices. Furthermore, the experiment of the proposed AVM algorithm showed improvement in task response time and efficiency in resource utilization compared to the other similar algorithms.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by King Saud University (Project no. RSP-2021/206), Riyadh, Saudi Arabia.

References

[1] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: a survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[2] G. H. Forman and J. Zahorjan, "The challenges of mobile computing," *Communications of the ACM*, vol. 36, no. 7, pp. 75–84, 1993.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[4] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: a survey, use cases & future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, 2017.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, pp. 13–16, Helsinki, Finland, August 2012.

[6] D. Mazza, D. Tarchi, and G. E. Corazza, "A cluster based computation offloading technique for mobile cloud computing in smart cities," in *Proceedings of the IEEE International Conference on Communications (ICC), 2016*, pp. 1–6, Kuala Lumpur, Malaysia, October 2016.

[7] S. Ranadheera, S. Maghsudi, and E. Hossain, "Computation offloading and activation of mobile edge computing servers: a minority game," *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 688–691, 2018.

[8] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: leveraging mobile devices to provide cloud service at the edge," in *Proceedings of the IEEE 8th International Conference on Cloud Computing (CLOUD), 2015*, pp. 9–16, Nice, France, March 2015.

[9] G. Orsini, D. Bade, and W. Lamersdorf, "Computing at the mobile edge: designing elastic android applications for computation offloading," in *Proceedings of the 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 112–119, Munich, Germany, June 2015.

[10] O. Chabbouh, S. B. Rejeb, Z. Choukair, and N. A. Ueve, "Computation offloading decision algorithm for energy saving in 5G/HetNets C-RAN," in *Proceedings of the 2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*, pp. 284–289, Marrakech, Morocco, September 2016.

[11] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[12] M. H. ur Rehman, C. Sun, T. Y. Wah, A. Iqbal, and P. P. Jayaraman, "Opportunistic computation offloading in mobile edge cloud computing environments," in *Proceedings of the 17th IEEE International Conference on Mobile Data Management (MDM)*, vol. 1, pp. 208–213, Porto, Portugal, June 2016.

[13] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[14] J. Dolezal, Z. Becvar, and T. Zeman, "Performance evaluation of computation offloading from mobile device to the edge of mobile network," in *Proceedings of the 2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 1–7, Berlin, Germany, October 2016.

[15] J. Zhou and X. J. Zhang, "Fairness-aware task offloading and resource allocation in cooperative mobile edge computing Z," *IEEE Internet of Things Journal*, 2021.

[16] C. You and K. Huang, "Multiuser resource allocation for mobile-edge computation offloading," in *Proceedings of the Global Communications Conference (GLOBECOM)*, pp. 1–6, Washington, NJ, USA, March 2016.

[17] F. Messaoudi, A. Ksentini, and P. Bertin, "On using edge computing for computation offloading in mobile network," in *Proceedings of the Globecom 2017-IEEE Global Communications Conference*, Singapore, December 2017.

[18] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[19] T. M. Alfaqih and J. J. Al-Muhtadi, "Internet of things security based on devices architecture," *MEC Compation*, vol. 133, no. 15, pp. 19–23, 2016.

[20] M. Derakhshan and Z. Bateni, "Optimization of tasks in cloud computing based on MAX-MIN, MIN-MIN and priority," in *Proceedings of the 2018 4th International Conference on Web Research (ICWR)*, pp. 45–50, Tehran, Iran, April 2018.

[21] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. i. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," *PLoS One*, vol. 12, no. 5, Article ID e0176321, 2017.

[22] M. Sanaj and P. J. Prathap, "An enhanced Round robin (ERR) algorithm for effective and efficient task scheduling in cloud environment," in *Proceedings of the 2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA)*, pp. 107–110, Cochin, India, July 2020.

[23] Z. Kuang, Z. Ma, Z. Li, and X. Deng, "Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing," *Journal of Systems Architecture*, vol. 118, Article ID 102167, 2021.
[24] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," *IEEE Access*, vol. 8, pp. 54074–54084, 2020.
[25] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.
[26] Z. Li, L. Liu, Z. J. J. Tong, and T. Review, "Task scheduling algorithm based on virtual machine," *Availability Awareness in Cloud Platform*, vol. 11, no. 5, 2018.
[27] 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.
[28] H. Li, M. Muskulus, and L. Wolters, "Modeling job arrivals in a data-intensive grid," in *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 210–231, Springer, Malo, France, June 2006.