Research Article

Saleh ALFahad, Qiyuan Wang, Christos Anagnostopoulos*, and Kostas Kolomvatsos

Task offloading in mobile edge computing using cost-based discounted optimal stopping

https://doi.org/10.1515/comp-2023-0115
received August 4, 2023; accepted January 22, 2024

Abstract: Mobile edge computing (MEC) paradigm has emerged to improve the quality of service & experience of applications deployed in close proximity to end-users. Due to their restricted computational and communication resources, MEC nodes can provide access to a portion of the entire set of services and data gathered. Therefore, there are several obstacles to their management. Keeping track of all the services offered by the MEC nodes is challenging, particularly if their demand rates change over time. Received tasks (such as, analytics queries, classification tasks, and model learning) require services to be invoked in real MEC use-case scenarios, e.g., smart cities. It is not unusual for a node to lack the necessary services or part of them. Undeniably, not all the requested services may be locally available; thus, MEC nodes must deal with the timely and appropriate choice of whether to carry out a service replication (*pull action*) or tasks offloading (*push action*) to peer nodes in a MEC environment. In this study, we contribute with a novel time-optimized mechanism based on the optimal stopping theory, which is built on the cost-based decreasing service demand rates evidenced in various service management situations. Our mechanism tries to optimally solve the decision-making dilemma between pull and push action. The experimental findings of our mechanism and its comparative assessment with other methods found in the literature showcase the achieved optimal decisions with respect to certain cost-based objective functions over dynamic service demand rates.

*** Corresponding author: Christos Anagnostopoulos**, Knowledge and Data Engineering Systems, School of Computing Science, University of Glasgow, Glasgow, UK, e-mail: christos.anagnostopoulos@glasgow.ac.uk
Saleh ALFahad: Knowledge and Data Engineering Systems, School of Computing Science, University of Glasgow, Glasgow, UK, e-mail: 2426473A@student.gla.ac.uk
Qiyuan Wang: Knowledge and Data Engineering Systems, School of Computing Science, University of Glasgow, Glasgow, UK, e-mail: q.wang.1@research.gla.ac.uk
Kostas Kolomvatsos: Department of Informatics & Telecommunications, University of Thessaly, Thessaly, Greece, e-mail: kostasks@uth.gr

Keywords: service management, sequential decision making, task offloading, mobile edge computing, optimal stopping theory

1 Introduction

Through several application areas, such as smart homes, smart agriculture, manufacturing, and healthcare, advancements in the Internet of Things (IoT) have a revolutionary effect on society and the environment. To accomplish this, a growing number of diverse IoT devices are being linked to provide real-time monitoring and actuation in various application areas. Big data refers to the massive amounts of information sent to the cloud through IoT devices. Due to the following factors, centralized processing in the cloud is unsuitable for numerous IoT applications. First, the fact that many latency-sensitive applications cannot tolerate the delay introduced by centralized cloud-based deployment [28]. Second, there is an increased likelihood of data being lost and network failure [24]. Third, the IoT devices' battery life may decrease more quickly if it is constantly uploading data to the cloud [17]. Finally, request and response must be tightly coupled in particular situations [3]. In order to solve these problems, many ideas have been presented that bring cloud-like assets to the network edge. In particular, mobile edge computing (MEC), fog computing, and cloudlets have gained much attention. Moreover, all of these approaches are grouped together as edge computing (EC) platforms.

The term "mobile edge computing" (MEC) refers to a novel concept in network architecture that places capabilities for information technology and cloud computing at the periphery of mobile networks [19]. MEC can provide a service environment with extremely low latency, high bandwidth, and direct access to real-time network information because it is located in close proximity to its customers [26]. Mobile cloud computing (MCC) was first used to offload computation, although it relied only on mobile devices and the main cloud server sides to do this [7]. Nevertheless, during offloading operations, the main cloud side is not near the mobile devices side. As a result, this causes a latency issue on the media connection middleware and a

major defect that prevents user mobility over the offloading task [11]. MEC was then launched to assist in fixing the offloading process in MCC's latency issue. The close position of the MEC to the user/devices, as well as the need for low latency and the delivery of high workload capacity, is a defining element of MEC [18]. Due to closeness to end users, MEC has much lower transmission and computation latencies than MCCs that rely on more conventional, far-flung resources for computation offloading [14].

Storage and processing power at MEC nodes are restricted. However, they may provide necessary computational requirements for a number of services that facilitate the execution of a wide range of processing tasks [10]. Data processing tasks, for instance, predictive analytics and explanatory-driven models, are often offered in the form of "requests" in such scenarios [20]. Moreover, in certain scenarios, nodes offload processing tasks onto peers by sharing services or data. Edge nodes can only host some of the available services and data [8]. In order to successfully carry out the processing tasks that have been given by applications or users in the manner of tasks, services are a vital requirement. As a consequence of this, the demand for services is continually adapting in order to satisfy the ever-changing requirements of a variety of activities and duties. It implies that a task may request a service that may or may not be accessible locally depending on its availability [10]. As mentioned in the study by ALFahad et al. [1], the nodes have two options:

- Option 1: Nodes are able to undertake a service replication (pull action) from the cloud or their edge peers; however, this is only possible if the service is adequate for the computational capabilities that they possess;
- Option 2: Nodes have the ability to assign the duty, often known as a "push action" to the peers or the cloud that are currently hosting the service(s).

ALFahad et al. [1] adopted optimal stopping theory (OST) to optimize and regulate the replication of services in an EC environment, leaving the initiative with the EC nodes to increase their autonomy. To ensure the service can do the required activities as efficiently as possible, EC nodes must determine locally when to duplicate the service. Each EC node, in accordance with the requirements of incoming tasks and the node's condition, makes its own decision-making about the replication of services *independently*. This proposal is useful when the nodes receive the same average number of requests for tasks.

In this work, we analyze the pull action when the quantity of received tasks changes, especially decreases due to the MEC's mobility. Furthermore, MEC nodes will become more independent and provide confirmation that the replication of services could be optimized and monitored in the MEC environment by applying the ideas of OST [6]. In this scenario, MEC nodes must determine individually *when* it is optimal to duplicate the service in order to maximize the opportunity to carry out the sort of tasks. It is important to note that every MEC node makes its own judgment about whether or not to replicate services based on its own internal state and the requirements of any arriving task.

The following sections of this article are laid out as follows: before diving into the details of the proposed OST-based decision-making model in Section 3, we present an overview of prior work, rationale, and novelty in Section 2. Section 4 provides the results of our performance and comparative evaluation, while Section 5 summarizes the work and suggests additional areas for study.

2 Related work and contribution

2.1 Related work

It is advisable to start comparing our current work with the previous work [1], as the strengths that encouraged us to embark on this work will be evident from the comparison. The work in the study by ALFahad et al. [1] focused on decision-making in EC devices, which are mainly static. Through this, EC devices receive a fixed average number of requested tasks, and then, a decision on service replication is made. In contrast, our current work carries the advantage of supporting the MEC devices to make the appropriate decision of pull action for the required services. Moreover, we focused in this study on the case of the MEC devices in the scenario when the number of requested tasks received is decreasing due to mobility. As a result, the MEC devices use our innovative method to make the most appropriate decisions individually.

Chen et al. [5] examined the authors examine the possibility of offloading tasks in a dynamic MEC framework. They proposed a hybrid energy supply strategy in which energy-collecting technology is included in IoT devices. In order to reduce the overall system cost, we coordinate the optimization of local processing and offloading time. Moreover, they provided a randomized optimization-based online dynamic task offloading technique for MEC using a hybrid energy supply. This technique is able to offload tasks based on system cost and queue stability. However, our strategy is distinct from that one because our methodology focuses mainly on the actual advantage of the reward, which reflects the requests existing users have made for a particular service. Offloading decisions are basically formulated in previous studies [9,16] on the basis of a resource scheduling factor.

However, our technique determines whether to decide on a push-pull action based on the total number of newly assigned tasks and the associated costs.

Zhang et al. [27] provided a task offloading strategy that takes into consideration both the categorization of tasks and the selection of offloading nodes. They aimed to reduce the amount of time it takes to complete every task as much as possible. Nevertheless, this research emphasizes making decisions based on the level of demand for the service necessary to accomplish the task. Liu et al. [12] offered a dependent task offloading architecture for multiple MEC environments. MEC devices may offload their compute-intensive duties within this framework to the MEC-cloud system while maintaining dependent restrictions. In order to provide a better experience for the client, it is able to delegate offloaded tasks to the MEC and the cloud dynamically. Since our approach cost-based sequential decision making (COST) has firm deals with limited capacity better than the approach in the study by Liu et al. [12], we concentrate on making the choice of whether to pull push action for a single user to an edge node or a peer in our method.

The authors deal with offloading decisions in the study by Wang et al. [25] based on energy and task causality limitations because of channel oscillations and the arrival of dynamic tasks over time. They do so by jointly improving the transmission power allocation at the energy transmitter (ET) for wireless power transfer (WPT) and the task allocation at the user for local computing and offloading over a particular finite horizon. This allows them to reduce the total transmission power used at the ET while maintaining the successful task execution of the user. This is accomplished over a particular finite horizon. However, the implementation of our solution may vary based on the level of demand for the service and the point of making the decision at the maximum reward. The approach in the study by Alfakih et al. [2] copes with the problem of resource management at the edge server and as a means of determining the most effective offloading decision to lower overall system costs. In addition, Tang and Wong [22] tried to make a choice to offload based on reducing costs to the absolute minimum. Our work, on the other hand, focuses on selecting the route that will result in the most significant reward while also taking into account the extra cost incurred if we skip over the option of push action.

2.2 Rationale and contribution

In Figure 1, we provide a simplified representation of the problem when the number (rate) of incoming tasks is

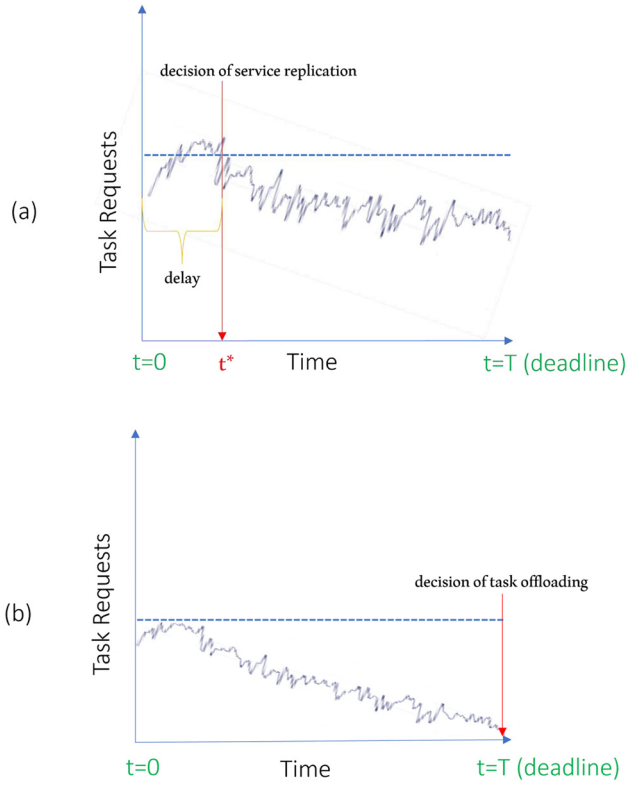

Figure 1: Making an informed decision based on the task request rate at each time instance t to either (a) indicates a high popularity of the service or (b) indicates a low popularity of the service.

decreasing per time and a node must decide whether to replicate a service (i.e., pull action) from its neighbors on the network edge or from the cloud and handle the processing locally or to offload the task to the cloud provider (i.e., push action) and let it handle the processing. Suppose that the time when the amount of requests for tasks begins to decrease is $t = 0$, and the time when the task must be completed, which, hereinafter we refer to as the deadline $T > 0$. Two choices with limited time are shown in Figure 1. The situation shown in Figure 1(a) is one in which the pull action choice is made early in time t^* (and before the deadline T) due to the expectation of getting no deep decrease of received tasks seeking the specified service. This indicates that there are quite an acceptable number of tasks interest in this service, suggesting that pulling the service locally to the node is a good option. Moreover, one can observe what would have happened if we postponed making a decision with the help of our growing assurance that interest in the service is low, as shown in Figure 1(b). Assuming that the service request rate was insufficient, the node has the option to offload the requests (push action) to its peers or the cloud after the deadline has passed without a pull action being agreed upon. This means that offloading is

Algorithm 1: Cost-based decaying sequential decision-making (DOST)

Input: Initial mean rate μ; decaying factor $0 < q < 1$; delaying cost $c > 0$.
Output: Optimal rate and time t^*.
$Stop \leftarrow False$; $t \leftarrow 0$; $R_0 \leftarrow 0$;
Repeat
Observe number of requests Y_t;
$R_t \leftarrow R_{t1} + Y_t$ /* update the cumulative sum of requests */ ;
If criterion in (6) is satisfied **Then**
Stop ← True;
$t^* \leftarrow t$ /* activate service replication (pull-action) and start-off a new task's service*/;
Else
$t \leftarrow t + 1$ /* continue with the next received task*/ ;
End
Until $t \leq T$
If $(Stop = False)$
Then call: push-action
End

required since there was a low amount of tasks seeking the specified service. Let us assume that the node made the decision to download (pull) the service early on, and then, after some time, we discovered that only a tiny percentage of users actually wanted it. If we had not performed the pull service action, we would have wasted more resources. Therefore, as shown in Figure 1(b), it would have been beneficial to postpone our decision until the completion of the specified time frame and then carry out the offloading action. However, if the node had chosen to send (offload) the request to the cloud at an earlier moment, the node would not have had sufficient trust that this request was frequently demanded on the node and would have been inefficient as a result. This service would have been expected to be requested in the near future. Hence, a push-based approach is unsuitable.

An early push action may offload tasks to other peers or the cloud prematurely, leading to network congestion if the node later determines that service interest has not decreased significantly. Therefore, as illustrated in Figure 1(a), the correct procedure is to intelligently postpone any choice and execute service replication (pull action) from the neighbors of the node or the cloud to the node at a certain period t^*. Finding the optimal balance between task offloading and service replication in the scenario of receiving a decreased number of tasks is the challenge we face now that we know it has been formulated as t^*. Taking into consideration the latency costs, a creative solution has been provided that guarantees the right option is made at time t^*.

2.2.1 Contribution

Our methodology is flexible enough to be applied to applications that make use of offloading decision-making algorithms in MEC circumstances. The primary technological contribution that we make is:

– In the event that the number of tasks received decreases, we present a unique adaptive approach that increases the possibility of making the choice of service replication (pull action) of the service with the highest (maximal) return. This is shown by a total cumulative of the total

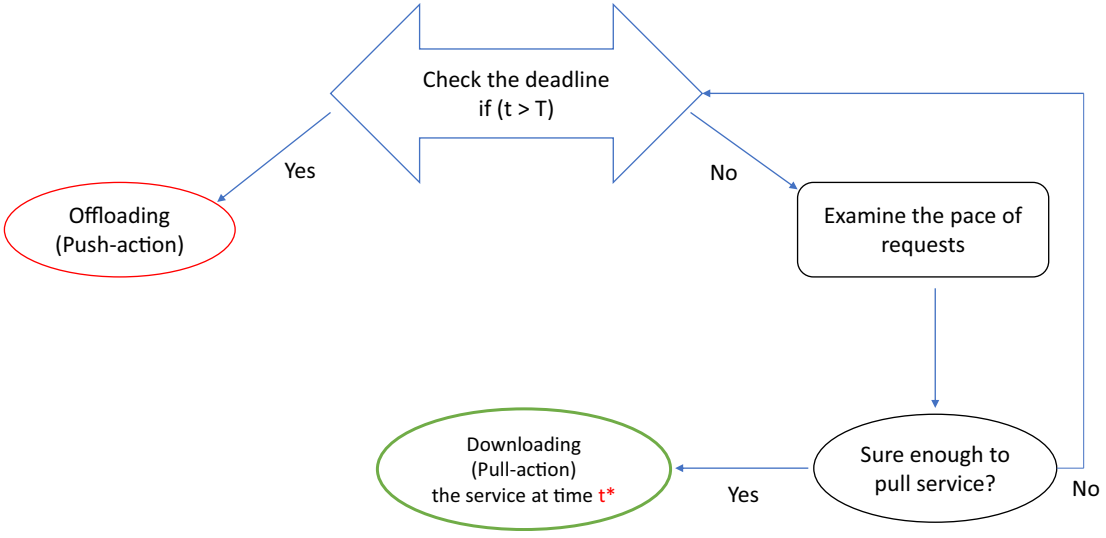

Figure 2: A diagram illustrating the sequential decision-making process for determining the optimal time $t^* < T$ for either task offloading (push action) or service replication (pull action) by the MEC node.

amount of task requests to the node that is being considered while taking notice of the cost that has been got.
- Using the concepts of OST, we theoretically analyze the superiority of our approach.
- We provide a comprehensive analysis of how our OST-based approach compares to our prior work [1], which generates the true optimum reward.

Figure 2 emphasizes on the decision-making state space. In particular, the technique for determining whether to replicate a service (pull action) or offload a task (push action) is shown in Figure 1. The number of tasks received at each time instance $t < T$ for a given service is a significant factor in this choice. The justification is as follows. At each time instance t, the node obtains some received task requests Y_t. The node then has two choices:
- It need to decide to terminate service at time $t \leq t^* < T$, if the number of requests is relatively large enough.
- Or, the node will re-evaluate its decision at time $t + 1$, if there are incoming service requests.

In the event that the given deadline T is met, a choice will be made on whether the task(s) will be passed on to a peer MEC node or to the cloud to be processed further.

3 Problem fundamentals

First, we provide an overview of the system model, and then in Section 3.2, we explore the description of the problem. In Section 3.3, we introduce our COST w.r.t. maximizing the rate of return in detail.

Table 1: Notations of parameters

Notation	Definition
$\mathcal{M}$	Set of MEC nodes
T	Deadline
t	Time instance
t^*	Optimal stopping time
Y_t	Number of tasks received at time t
λ	Arrival rate of tasks Y
θ	Tolerance threshold
c	Cost of the decision delay for the DOST model
b	Cost of the decision delay for COST model
R_t	Cumulative sum of the number of tasks Y_t up to t

3.1 System model

As shown in Figure 3, we take into account that the network environment is made up of a distant cloud data center as well as an MEC system [4]. Each MEC base station is equipped with certain MEC servers, and the MEC system is built up of MEC base stations. Table 1 includes the notations used in this article. The set of MEC nodes will be referred to as $\mathcal{M} = \{1, 2, ..., M\}$. In general, we operate under the concept that a number of different service providers each install their own applications on each MEC

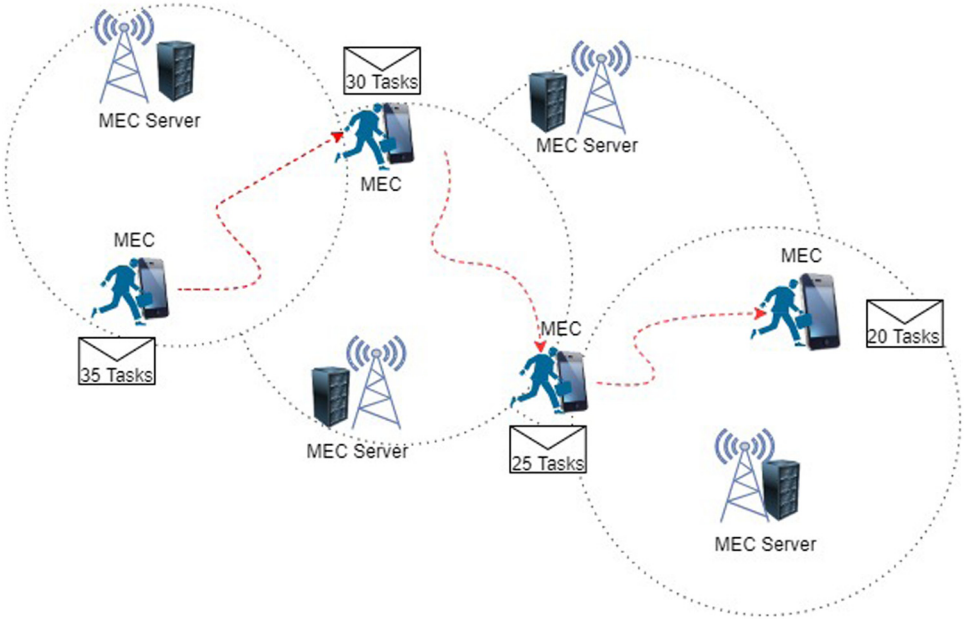

Figure 3: Example of MEC environment.

node. Each MEC device has the capability to assign tasks to an MEC node that is provided with suitable computing services. Furthermore, MEC devices have the opportunity to offload tasks to the cloud, provided that the cloud has all of the necessary services as well as an adequate number of computing resources [13]. The highest possible reward value is taken into consideration when MEC devices make a choice about task service replication (pull action). The period of time is split into discrete intervals that are represented by $t = 1, 2, \ldots, T$. At a certain point in time t^*, the MEC device will, based on local information (for instance, task information that includes the number of services that have been received) come to a conclusion on whether or not service replication (pull action) should occur. We build our studies on the assumption that the actual position of the MEC device and its associated network environment is unchanging from one-time instance to the next. Because of this, we are able to provide a guarantee that choosing the task of replication (pull action) will take place at the same moment as instance t.

3.2 Problem definition

The task model assumes that the number of requests for a particular task at a given time instant, denoted by Y_t, follows a Poisson distribution with a rate of λ [21]. As the MEC device is presumed to be in motion, fewer tasks may be received. Assuming an exponential decay, the number of tasks received after a certain amount of time t is Poisson with mean μq^{t-1}, where q is the decay factor, which is a scalar between 0 and 1, while μ represents the initial mean rate. In addition, since MEC devices are always in motion, the number of MEC nodes that an individual MEC device is able to sense and receive tasks from at any single instance t. As an outcome, the accumulative total of the number of requests made up to t is applied to every task that arrives on the MEC device. We define the cumulative distribution function of the demand rate at t as:

$$P(Y \le y;\lambda) = \sum_{k=0}^{y} \frac{\mathrm{e}^{-\mu} \cdot \mu^k q^{k(t-1)}}{k!}. \tag{1}$$

Consider now the cumulative sum of the reduced number of requests R_t, $0 \le t \le T$ as:

$$R_t = \sum_{k=0}^{t} Y_k. \tag{2}$$

Problem 1. Find a stopping rule t^* that maximizes the expected rate of return per time instance t in equation (3), where $R_t = Y_1 + \cdots + Y_t$:

$$t^* = \arg\max_t \frac{\mathbb{E}[R_t - c]}{\mathbb{E}[t+1]}. \tag{3}$$

Let $c > 0$ denote the cost incurred each time instance t that the MEC does not make the decision of a pull action. Our problem is to determine the ideal timing t to stop in order to maximize the total amount of rewards R_t.

The rationale behind Problem ?? lies in the fact we expect the rate of service requests per time t, excluding the latency cost, to be considerably not deep decreasing. Therefore, this exemplifies the possibility that the pull action will prove to be stochastically a more advantageous option than the push action would be. This is due to the fact that we have become more certain that in the (not-too-distant) future, we will be receiving a sufficient number of requests for a service in a node despite the fact that the number of requests is continuing to decrease. Because of this, our goal is to maximize this rate of return.

3.3 Cost-based discounted sequential decision-making (DOST)

In order to discover a solution to our problem in maximizing (3), we first focus on finding a stopping rule that maximizes $\mathbb{E}[W_t - \lambda D_t]$, where $W_t = R_t - c$ and $D_t = t + 1$, with rate $\lambda > 0$. We first check whether the one-stop look ahead policy (1-sla) is the optimal one. Specifically, if we quit after a period of time t, we will have gained reward:

$$R_t - c - \lambda(t+1). \tag{4}$$

Moreover, if we proceed to the next time instance $t + 1$ and then stop, we would anticipate that we will have gained

$$R_t + \mathbb{E}[Y_{t+1}] - c - \lambda(t+2) = R_t + \mu q^t - c - \lambda(t+2). \tag{5}$$

Since the difference between (4) and (5) is monotone, the optimal policy is 1-sla.

Theorem 1. *The OST rule at which an MEC node stops at the first time* T_1 [15], *which maximizes* (3) *is provided by*:

$$T_1 = \min\{t > 0 : \lambda \ge \mu q^t\} = \min\left\{t > 0 : t \ge \frac{\log(\mu/\lambda)}{\log(1/q)}\right\}. \tag{6}$$

Proof. The 1-sla policy in Theorem 1 is the optimal criterion that can maximize the rate of return. We proceed with estimating this time T_1 as follows. We can re-write the 1-sla given the initial mean μ as [23]:

$$T_1 = m, \text{ where} \begin{cases} m = 1, & \text{if } \lambda \ge \mu \\ m = \dfrac{\log(\mu/\lambda)}{\log(1/q)}, & \text{if } \lambda < \mu. \end{cases} \tag{7}$$

The expected return anticipated from (7) is a function of λ:

$$\begin{aligned} V(\lambda) &= \mathbb{E}[R_m - c - \lambda(m+1)] \\ &= \mu(1 + q + \cdots + q^{m-1}) - c - \lambda(m+1) \\ &= \frac{\mu(1-q^m)}{(1-q)} - c - \lambda(m+1). \end{aligned} \tag{8}$$

In order to find λ, we first make this equation equal to zero; thus, we obtain

$$\lambda = \frac{\frac{\mu(1-q^m)}{(1-q)} - c}{(m+1)}. \tag{9}$$

Based on a λ value from (9), we iteratively feed it to (7), where we obtain an updated $T_1 = m$ value. This value is then fed back to equation (9) to obtain a new value of λ. We proceed with this recursion until convergence. The outcome is the optimal (maximal) expected rate of return λ along with the optimal stopping time m, where an MEC node should stop within the horizon $\{1, \ldots, T\}$ in order to maximize the rate of return in equation (3). □

Our cost-based decaying sequential decision-making process is provided in Algorithm 1.

4 Experimental evaluation

4.1 Experimental setup

We test two distinct models in our experiments that show evidence of our approach: (i) the DOST model, which is determined by the initial rate μ; $0 < q < 1$ incurred cost $c > 0$. In our proposed model, the decision to perform a pull action is made if the accumulation value, R_t, at the time instance, t, meets the criteria outlined in equation (7). (ii) The COST model in the study by ALFahad et al. [1], which calculates the cumulative total of the received tasks R_t based on the stopping criterion is as close as feasible to a budget threshold θ. After this, it makes the choice of which service replication (pull action) will result in the biggest reward while also taking into consideration the cost that will be incurred b for each time instance t, as shown in equation (10). In every other case, the decision to offload is taken if the deadline elapses.

In the experiments that were designed to provide a comparative analysis in Section 4.2, we conducted the test using both the DOST and the COST models. In addition, we determine the values for each of the variables *mu*, q, and *theta*, as well as the cost c and cost-per-time b. Furthermore, we adjusted $\mu = 10$, $c = 0.1\mu$, and $b = 0.1\mu$. Moreover, we split the experiments into two groups. The first group included an unchanged value for budget $\theta = 50$. After that, we conducted the comparative tests with respect to the payoff of the DOST and COST for $q \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$. Finally, we analyzed the results of the experiments. Furthermore, our assessment involves fixing a value for $q \in \{0.7, 0.8, 0.9\}$ and running comparison experiments between the payoff of DOST and COST for various values of $\theta \in \{10, 30, 50, 100\}$. All parameters are defined to simulate different scenarios that examine the models in different restrictions, for instance, capacity that is denoted by θ and decay factor q. In the second group, we carried out experiments where we compared the amount of time required to stop the DOST and COST models along with the amount of time required to stop with the associated payoff.

4.2 Comparative assessment

Choosing the moment to stop and maximizing cumulative reward using OST is the foundation of the strategy proposed in the study by ALFahad et al. [1], which we term here as cost-based sequential decision making (COST). The goal is to prevent more tasks from being accepted when the cumulative total of those tasks, denoted by the variable R_t, is getting close to a value θ, which represents the tolerance threshold (budget). In addition, taking into consideration the total amount of incurred cost $b > 0$ due to delays up to that moment t. As a result, the optimal stopping rule with the highest payoff achieved in the study by ALFahad et al. [1] is provided in equation (10) for reasons of completion:

$$t^* = \inf\left\{ t \geq 0 : \sum_{y=0}^{\theta - R_t} (R_t + y) - b(t+1)P(Y=y) \leq R_t - bt \right\}. \tag{10}$$

The parameters for the first iteration of the comparative evaluation experiment are as follows: $\mu = 10$, $c = 0.1\mu$, and $b = 0.1\mu$. Furthermore, we left $\theta = 50$ and $q \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$ unaltered in their respective values. The results of our experiment are mostly influenced by the different values of q, c, and b. Along with this, the experiment determines the choice that resulted in the greatest payoff at the time indicated by t^*. In addition, the experiment was carried out using both of the models.

The COST used to be compared with our DOST approach. The many effects that q has on both models in terms of payoff

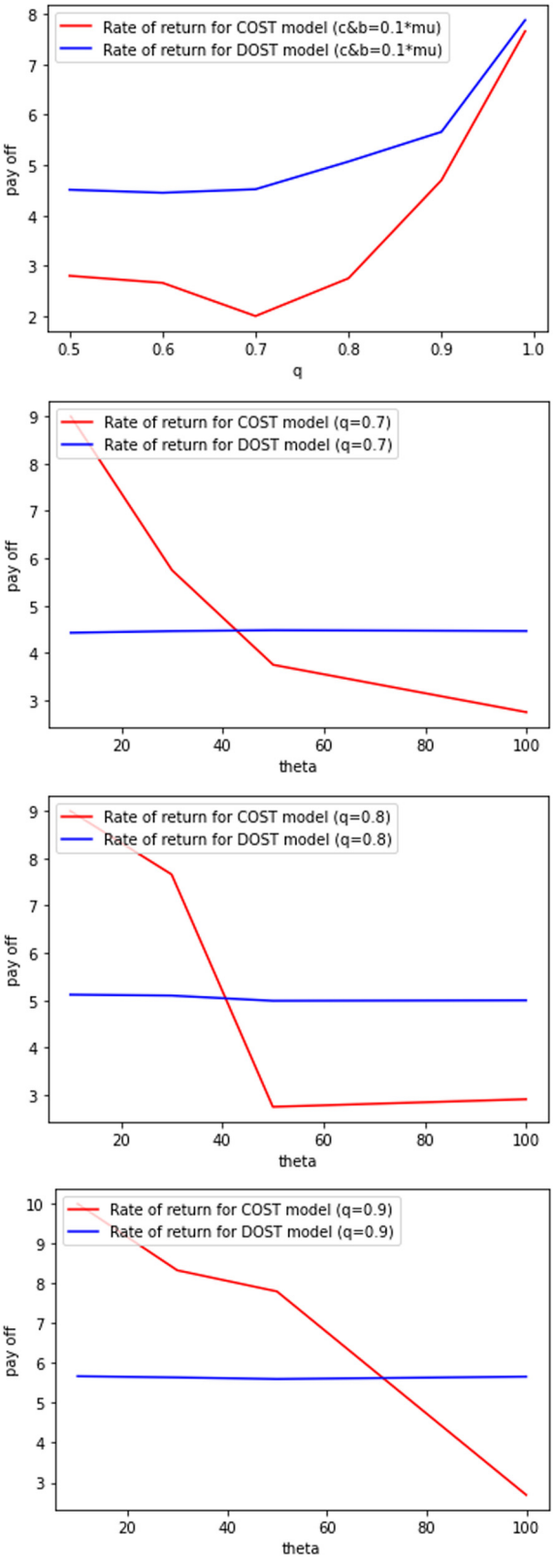

Figure 4: Expected payoff vs q for fixed delay cost value c & $b = 0.1*\mu$ for DOST and COST models; Expected payoff vs θ for fixed $q = 0.7$ value for DOST and COST models; Expected payoff vs θ for fixed $q = 0.8$ value for DOST and COST models; Expected payoff vs θ for fixed $q = 0.9$ value for DOST and COST models.

are shown in Figure 4. Also, from Figure 4, we note that there are different values for the q, which means that $q = 0.5$, there is a sharp decline in the number of tasks received, and as the value of q increases, the decrease in the number of tasks received decreases. In this scenario, we note the red line, which represents the value of payoff across the different q values of the COST model. On the other hand, we also note the blue line, which represents the value of the payoff for our new approach, the DOST model. Furthermore, it is noted that the DOST model has a high return on payoff compared to our previous work COST, regardless of the severity of the decline in the number of requests received. However, our previous work was clearly affected by the sharp drop in the number of tasks received, as it shows a low value of payoff compared to our DOST model, and these payoffs improve as the value of q increases. Despite that and on all q values, the effectiveness of the performance of our DOST model has a higher return of payoff than our previous work COST model, and this demonstrates the strength of the principle that we have chosen to work with the challenge of decreasing the number of tasks within the specified time T.

The following are the parameters that should be used for the comparative assessment experiment's first group as shown in 4: $\mu = 10$, $c = 0.1\mu$, and $b = 0.1\mu$. In addition, we adjusted the variables for different values of q. we experimented when the value of q set to 0.7 and the $\theta \in \{10, 30, 50, 100\}$. The outcomes are mostly determined by the varying values of the variables θ, q, c, and b, respectively. In addition, the experiment found the option that led to the biggest payoff at the time indicated by the symbol t^*. Beyond that, the experiment was conducted with both of the models, i.e., DOST and COST. Figure 4 demonstrates the effects of θ on the COST model in terms of payoff when q is set to 0.7, illustrating various impacts. A further observation that can be made from Figure 4 is that there is a wide range of values for the parameter θ, which stands for the diversity of MEC device capacities. If θ has a significant number, it indicates that the capacity is also large. The red line, which reflects the payoff value across the many θ values of the COST model, is something we observe in this particular circumstance. On the other hand, we have also paid attention to the blue line, which depicts the value of the payoff for our recently developed strategy, which is the DOST model. On top of that, it has been observed that the DOST model has a fixed return of payoff in comparison with our previous COST, despite the fact that θ may take on many different values. However, it is evident that the various values of θ influenced our earlier study, as it demonstrates a low value of payoff that has a deep decreasing parallel with the expanding of θ values. In spite of this, and for all θ values, the effectiveness of the performance of our DOST model has a higher return of payoff than our previous work COST, particularly with MEC devices that required high capacity

volume. This shows the power of the concept that we decided to work with.

Moreover, we modified the parameters by setting $q = 0.8$. The experiment also determined the time (t^*) at which the choice with the highest payment was available. Furthermore, both models were considered throughout the duration of the investigation, i.e., DOST and COST. Some of the payoff-related effects by θ on the COST model are shown in Figure 4. Figure 4 further shows that the parameter θ, which represents the variety of MEC device capabilities, may take on a large set of values. If θ is considerable, then the capacity is likewise substantial. In this case, we see the red line, which represents the payoff value for all θ variables in the COST model. Meanwhile, we have also been keeping an eye on the blue line, which shows the payoff value of our newly devised technique, which is the DOST model. Although θ may take on a wide range of values, it has been noted that the payoff from the DOST model is fixed relative to our prior COST and is hence larger than the value reported when $q = 0.7$. COST shows that different θ values have an effect since the payoff at low values follows a deep falling parallel with increasing θ values. Despite this, for all θ values, our DOST model beats our earlier work COST, especially when it comes to MEC devices that need a lot of storage space. This demonstrates how effective the idea was that we settled on.

Also, we set $q = 0.9$ as shown in Figure 4. In the present scenario, we see the payoff value throughout a wide range of θ in the COST model, shown by the red line. We have, however, also paid close attention to the blue line, which represents the payoff value of the DOST model. Moreover, it has been discovered that while θ may take on many different values, the DOST model has a fixed return of payoff compared to our prior COST, and this return is larger when compared to both values when $q = 0.7$ and $q = 0.8$. The COST was influenced by this finding because it shows a low value of payoff that has a deep decreasing parallel with the expanding of θ values, and $q = 0.9$ is a key factor in making the decrease in received tasks insignificant, leading to good results in terms of payoff at small values of θ. However, for all θ values, our DOST model surpasses our earlier work COST, especially when it comes to MEC devices that need a lot of storage space.

The following are the parameters that should be used for the comparative assessment experiment's second group as shown in Figure 5: $\mu = 10$, $c = 0.1^*\mu$, and $b = 0.1^*\mu$. The settings for the second group, in which q was set to 0.9 and θ was varied over the ranges $\in \{10, 30, 50, 100\}$, were likewise adjusted by us. The findings of the second group are significantly affected by shifts in the values of θ, q, c, and b, accordingly. The experiment also calculated the time (t^*) at which the option with the biggest potential payoff was

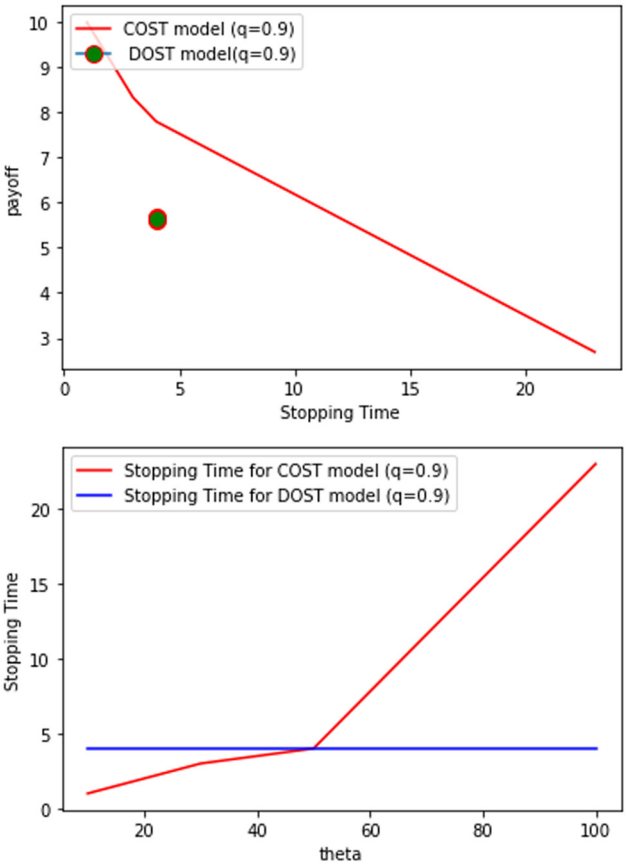

Figure 5: Stopping time vs expected payoff for fixed $q = 0.9$ value for DOST and COST models.

shown. Furthermore, the experiment was conducted while keeping into consideration both of the models. The COST was often evaluated in comparison with the DOST model, which served as the primary foundation of our strategy. The effects of varying theta values throughout the stopping time are shown in Figure 5, which was created using the COST and DOST models. In the current situation, we see that the stopping time in the COST model varies throughout a broad spectrum of values for θ, as shown by the red line. Nevertheless, particular attention should be given to the blue line representing the DOST model, which indicates the stopping time for the model discussed. In addition, it was found that the DOST model has a constant stopping time in comparison with our earlier COST, despite the fact that the value of θ might take on many distinct forms. The COST was impacted by late stopping time in tandem with the growing of θ values. Moreover, $q = 0.9$ is a critical component in making the drop in received tasks minimal, leading to positive findings in terms of early stopping time for tiny values of θ. Our DOST model is superior to our previous work COST, particularly when it comes to MEC devices that need a significant amount of storage space.

Moreover, the experiment's second group details a range of stopping timings for a number of payoff values, as shown in Figure 5. The COST and DOST models were used in its development. As indicated by the red line, the stopping time and payoff in the COST model exhibit increasing harm across a wide range of values for θ. However, you should also focus on the green dot within a thick red color, as this represents the moment at which our DOST model will stop and with the highest return payoff. Even if the value of θ may assume a variety of forms, it was discovered that the DOST model has a set stopping time and payoff compared to our prior COST. Positive results in terms of early stopping time and the high payoff for insignificant values of θ are also seen when $q = 0.9$ is used to keep the decline in received tasks to a minimum. As the θ value increases, the situation deteriorates. Therefore, when it comes to MEC devices that need a large amount of capacity for storage, our DOST approach is better than our prior work COST. This demonstrates the uniqueness of our solution.

5 Conclusions

We offer end-users/applications in MEC a time-optimized decision-making strategy that is based on the OST and specifically in maximizing the rate of return. This decision-making technique is particularly useful in the situation in which users receive tasks with variable demand rates. We provide a full evaluation of the procedure when it comes to utilizing and implementing the DOST mechanism for service replication (pull action) in MEC systems. According to the findings of our experimental evaluations, the performance of DOST is superior to that of COST, an OST-based relevant work. These results are appropriate for use in MEC nodes and are not influenced by the total amount of resources that are available. Furthermore, the best results are achieved by our model, which is superior to baseline solutions in terms of payoff.

In the future, our goal is to develop a new model that adopts artificial intelligence (AI) methods for selecting the best set of MEC nodes to offload the tasks relying on multi-armed bandits algorithms.

Acknowledgement: The authors would like to thank the Saudi Prime Minister (Crown Prince Mohammad Bin Salman), the Royal Saudi Air Force, Major General Abdulmonaim ALharbi, Senior Engineer Mohammad Aleissa, Saudi Arabia, and the Saudi Arabian Cultural Bureau in the UK for their support and encouragement.

Funding information: The authors state that no funding is involved.

Author contributions: All authors have accepted responsibility for the entire content of this manuscript and approved its submission.

Conflict of interest: Dr. Christos Anagnostopoulos is the Editor-in-Chief of Open Computer Science but was not involved in the review process of this article.

Data availability statement: The datasets used in this article are publicly available.

References

[1] S. ALFahad, C. Anagnostopoulos, and K. Kolomvatsos, "Time-optimized sequential decision making for service management in smart city environments." *Sustain. Cities Soc.* pp. 1–23, 2023 (Preprint).

[2] 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.

[3] H. AlMajed and A. AlMogren, "A secure and efficient ECC-based scheme for edge computing and internet of things," *Sensors*, vol. 20 no. 21, p. 6158, 2020.

[4] S. Chen, H. Chen, J. Ruan, and Z. Wang, "Context-aware online offloading strategy with mobility prediction for mobile edge computing," In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–9, 2021, IEEE.

[5] Y. Chen, F. Zhao, Y. Lu, and X. Chen, "Dynamic task offloading for mobile edge computing with hybrid energy supply," *Tsinghua Sci Technol.*, vol. 28, no. 3, pp. 421–432, 2022.

[6] P. V. Gapeev and L. Li, "Optimal stopping problems for maxima and minima in models with asymmetric information,". *Stoch.*, vol. 94, no. 4, pp. 602–628, 2022.

[7] K. Gasmi, S. Dilek, S. Tosun, and S. Ozdemir, "A survey on computation offloading and service placement in fog computing-based IoT," *J. Supercomput.*, vol. 78, no. 2, pp. 1983–2014, 2022.

[8] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, "The role of edge computing in internet of things," *IEEE Commun. Mag.*, vol. 56, no. 11, pp. 110–115, 2018.

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

[10] K. Kolomvatsos and C. Anagnostopoulos, "A proactive statistical model supporting services and tasks management in pervasive applications," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 3, pp. 3020–3031, 2022.

[11] M. Li, N. Xiong, Y. Zhang, and Y. Hu, "Priority-mece: a mobile edge cloud ecosystem based on priority tasks offloading," *Mob. Netw. Appl.*, vol. 27, no. 4, pp. 1768–1777, 2022.

[12] J. Liu, J. Ren, Y. Zhang, X. Peng, Y. Zhang, and Y. Yang, "Efficient dependent task offloading for multiple applications in mec-cloud

system," *IEEE Trans. Mob. Comput.*, vol. 22, no. 4, pp. 2147–2162, 2023.

[13] H. Lu, C. Gu, F. Luo, W. Ding, and X. Liu, "Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning," *Future Gener. Comput. Syst.*, vol. 102, pp. 847–861, 2020.

[14] M. Maray and J. Shuja, "Computation offloading in mobile cloud computing and mobile edge computing: survey, taxonomy, and open issues," *Mob. Inf. Syst.*, vol. 2022, pp. 1–17, 2022.

[15] G. Peskir and A. Shiryaev, *Optimal stopping and free-boundary problems*, Lectures in Mathematics, Birkhäuser Basel, ETH Zürich, 2006.

[16] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4192–4203, 2019.

[17] N. Saranya, K. Geetha, and C. Rajan, "Data replication in mobile edge computing systems to reduce latency in internet of things," *Wirel. Pers. Commun.*, vol. 112, pp. 2643–2662, 2020.

[18] S. L. Shah, I. A. Abbasi, A. Bashier Gism Elseed, S. Ali, Z. Anwar, Q. Rajpoot et al., "Tamec: trusted augmented mobile execution on cloud," *Sci. Program.*, vol. 2021 pp. 1–8, 2021.

[19] Z. Song, X. Qin, Y. Hao, T. Hou, J. Wang, and X. Sun, "A comprehensive survey on aerial mobile edge computing: Challenges, state-of-the-art, and future directions," *Comput. Commun.*, vol. 191, pp. 233–256, 2022.

[20] M. Soula, A. Karanika, K. Kolomvatsos, C. Anagnostopoulos, and G. Stamoulis, "Intelligent tasks allocation at the edge based on machine learning and bio-inspired algorithms," *Evol. Syst.*, vol. 13, no. 2, pp. 221–242, 2022.

[21] F. Sun, F. Hou, N. Cheng, M. Wang, H. Zhou, L. Gui, and X. Shen, "Cooperative task scheduling for computation offloading in vehicular cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11049–11061, 2018.

[22] M. Tang, and V. W. Wong, , "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mob. Comput.*, vol. 21, no. 6, pp. 1985–1997, 2020.

[23] P. Van Khanh, "Optimal stopping time to buy an asset when growth rate is a two-state Markov chain," *Am. J. Oper. Res.*, vol. 4, no. 3, pp. 1–10, 2014.

[24] M. Villari, M. Fazio, S. Dustdar, O. Rana, D. N. Jha, and R. Ranjan, "Osmosis: The osmotic computing platform for microelements in the cloud, edge, and internet of things," *Computer*, vol. 52, no. 8, pp. 14–26, 2019.

[25] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 4, pp. 2443–2459, 2020.

[26] C. Zhan, H. Hu, Z. Liu, Z. Wang, and S. Mao "Multi-uav-enabled mobile-edge computing for time-constrained IoT applications," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15553–15567, 2021.

[27] R. Zhang, L. Wu, S. Cao, X. Hu, X. Xue, D. Wu, and Q. Li, "Task offloading with task classification and offloading nodes selection for mec-enabled iov," *ACM Trans. Internet Technol. (TOIT)*, vol. 22, no. 2, pp. 1–24, 2021.

[28] B. Zheng, Z. Mei, L. Hou, and S. Qiu, "Application of internet of things and edge computing technology in sports tourism services," *Secur. Commun. Netw.*, vol. 2021, pp. 1–10, 2021.