information

MDPI

Article

Delay-Tolerant Sequential Decision Making for Task Offloading in Mobile Edge Computing Environments [†]

Ibrahim Alghamdi *, Christos Anagnostopoulos and Dimitrios P. Pezaros

School of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK; christos.anagnostopoulos@glasgow.ac.uk (C.A.); dimitrios.pezaros@glasgow.ac.uk (D.P.P.)

* Correspondence: i.alghamdi.1@research.gla.ac.uk

† This paper is an extended version of our paper published in the proceedings of Wireless Days 2019.

Received: 24 August 2019; Accepted: 7 October 2019; Published: 12 October 2019

Abstract: In recent years, there has been a significant increase in the use of mobile devices and their applications. Meanwhile, cloud computing has been considered as the latest generation of computing infrastructure. There has also been a transformation in cloud computing ideas and their implementation so as to meet the demand for the latest applications. mobile edge computing (MEC) is a computing paradigm that provides cloud services near to the users at the edge of the network. Given the movement of mobile nodes between different MEC servers, the main aim would be the connection to the best server and at the right time in terms of the load of the server in order to optimize the quality of service (QoS) of the mobile nodes. We tackle the offloading decision making problem by adopting the principles of optimal stopping theory (OST) to minimize the execution delay in a sequential decision manner. A performance evaluation is provided using real world data sets with baseline deterministic and stochastic offloading models. The results show that our approach significantly minimizes the execution delay for task execution and the results are closer to the optimal solution than other offloading methods.

Keywords: Mobile Edge Computing; tasks offloading; optimal stopping theory; sequential decision making

1. Introduction

In recent years, there has been a significant increase in the use of mobile devices and their applications. Mobile (edge) devices have become an essential part of our lives, while at the same time they are evolving rapidly in the face of advanced technology and context-aware applications [1]. Recent mobile nodes are not limited to the most advanced smartphones but also include new forms of mobile nodes such as intelligent vehicles [2] and drones [3]. Such devices run very advanced applications and thus require powerful computing resources which are not always available in the mobile nodes. Meanwhile, cloud computing has been considered as the latest generation of computing infrastructure. There has also been a transformation in cloud computing ideas and their implementation so as to meet the demand for the latest applications.

Cloud computing has become an efficient and flexible environment in which the capabilities of mobile devices can be enhanced [1]. This is carried out by sending computing tasks to distant and centralized data centres. Yet, such a solution introduces high latency and places more load on the mobile networks [4]. For example, experiments have shown that the total latency for cloud computing is in the range of 30–100 ms, which is not acceptable for many latency critical mobile applications such as real-time online gaming, virtual sports and autonomous driving [5]. Therefore, new forms of cloud computing architectures on the edge of the network have emerged. This idea has appeared in the literature under various names including: Cloudlets [6], fog computing [7] and mobile edge computing [8]. The common objective of these architectures is to provide cloud computing services closer to the mobile nodes on the edge of the network.

Cloudlets [6], introduced in 2009, are decentralized data-centres placed at different places, such as a coffee shop, closer to the mobile users such that compute cycles and storage resources can be leveraged by nearby mobile users [6]. One disadvantage of the cloudlet is that it is supposed to be primarily accessed by mobile devices over a WiFi connection [4]. Each time the user wants to access the cloudlet, they have to switch from their mobile network, such as 4G/3G, to Wi-Fi whenever the cloudlet services are exploited [4]. The other disadvantage is that the quality of service (QoS) is not guaranteed as the cloudlets are not part of the mobile network and the mobile devices might not be in the coverage range of the cloudlets [4]. Another architecture that was proposed is fog computing [7]. Fog computing was proposed in 2012 by Cisco to provide a platform for the Internet of Things (IoT) as well as big data applications [7]. It is suitable for applications that require low latency as in gaming applications, location awareness applications, geo-distributed or in large number of nodes as in sensor networks [7]. The QoS for mobile users can be hardly guaranteed by using the aforementioned paradigm as the cloudlet and the fog computing paradigm are not integrated into the architecture of the mobile network [4], thus the concept of mobile edge computing or multi-access edge computing (MEC) has been proposed.

The first introduction of MEC was by IBM and Nokia Siemens in 2013 when they released the world's first mobile edge computing system which has the ability to run applications directly within mobile base stations [9,10]. MEC is currently being standardized by an Industry Specification Group (ISG) within the European Telecommunications Standards Institute (ETSI) [4,8] with the name of mobile edge computing or multi-access edge computing. MEC provides an IT service platform and cloud-computing capabilities at the edge of the mobile network in close proximity to mobile subscribers [8]. The primary objectives of having this architecture are improving the QoS of mobile nodes and reducing the traffic (bandwidth consumption) between the network edge and the core network. MEC is a good platform for different scenarios and applications including offloading mobile applications, Internet of Things (IoT), or as a caching entity at the edge [8].

Computation offloading refers to the process of sending computation tasks and data to a remote server for delegating this computation [11]. Offloading can be a benefit for mobile nodes applications with lower latency and less battery consumption [12]. Examples of applications that can benefit from computing offloading are augmented reality (AR) [13], gaming, IoT applications, data analytics tasks at the edge [14], vehicular network (VN) [2] and unmanned aerial vehicles (UAV) [3]. Computation offloading in the MEC environment faces the challenge of the offloading decision. The offloading decision refers to the spatial decision and the temporal decision. The spatial decision refers to the locations of the offloading, namely, the edge server or the cloud. The temporal decision refers to the decision of selecting the best time to offload [15].

1.1. Challenges and Motivation Example

MEC servers can be deployed in a range of potential locations, including base stations [16] or roadside units [2]. Given the movement of mobile nodes between different MEC servers, the main aim would be the connection to the best server and at the right time in terms of the load of the server. There is significant variation when it comes to the loads of the MEC servers, for example, there are times when many users are connected to the same server, but times when there are few users using the same server [17]. There are many applications that should be offloaded to the MEC servers in order to improve the QoS of the mobile nodes. Consider, for example, the use case where an intelligent vehicle needs to perform a perception task as explained in [18]. Perception tasks refer to object detection and tracking, which can be carried out through the use of powerful deep learning models. In such a case, MEC server assistance is needed and task offload can be beneficial. Taking into account the deployment of MEC servers along the road and the mobility of the vehicles, there will be a number of MEC servers as candidates for task offloading and thus it can be challenging to select which MEC server to choose for offloading. The selection can be made in terms of MEC servers' selection (spatial decision), in the case of having high density deployment of MEC servers, or time selection (temporal decision), when the mobile node remains within the range of the MEC server.

1.2. Contribution

In this work, we tackle this challenge by applying the concept of the optimal stopping theory (OST). In particular, we provide a model by which the mobile nodes can decide which MEC server and which time to select to offload the computing tasks in order to have a minimized total delay. The contributions of this work are as follows:

- We extend our previous work in [19] and provide more evaluations by comparing the proposed model with two other offloading models.
- Different from our previous work, we use more representative data sets for mobiles nodes and for real data centre servers utilization.

The remainder of this paper is organized as follows: We summarize related work in Section 2, while details of the proposed OST-based decision making system are described in Section 3. Performance evaluation results are provided in Section 4. A discussion about the results is provided in Sections 5 and 7 concludes the paper and outlines future research directions.

2. Related Work

Various research has been carried out to deal with the issues of offloading data and computing tasks to an edge node, the majority of which have emphasised if there should be a local processing of the data or task, or whether it should be offloaded externally, for example, to an edge server or the cloud. Previous work has tended to address energy consumption and reducing the delay of execution. The spatial and time computation offloading decision algorithm (ST-CODA) in [15] is related to our work. This work assists in the decision-making of the mobile device in terms of the time and location for offloading tasks by considering the pros and cons of the computation nodes and the different transmission costs in an edge cloud-enabled heterogeneous network. The present work differs from this because the time-optimised sequential decision only offloads tasks to the edge servers and not to the cloud. In [15], the temporal decision refers to deferring the offloading decision until a low cost network is found. In our approach, we defer the offloading decision until a lightly loaded server with low transmission delay is found. By considering the load of the MEC server and the transmission delay, we are more likely to provide higher expected QoS for the users' applications.

The work in [20] puts forward a strategy of computation offloading for a data mining application, namely, activity recognition application for mobile devices. As the user moves, data is obtained from various places and it is held in storage on the mobile device. This data is examined so that a choice can be made as to whether to offload to an edge server or the cloud, or to carry out the process on the device. Should it be decided that offloading to an edge server takes place, the device's communication interface obtains a list of edge servers and connects to the best server. However, during movement, a better server could be present that is not picked up by the communication interface whilst the device's communication interface is scanning. Therefore, with regards to execution delay, it is possible that a better MEC server is available.

The work in [3], for the purpose of reducing the latency of task execution as well as energy consumption, presents the idea of collaborative mobile edge computing. This work looks at UAV applications that uses photos and videos for jobs like object identification or obtaining traffic information. The captured photos/videos are then offloaded to an edge server. When the task is generated by the UAV, a system orchestrator should determine which server should be selected, what data rate ought to be adopted to transmit data to the selected server and how much workload each servers (cooperators) should be allocated. This work [3] makes the assumption that the system orchestrator makes this choice. However, in our work, the decision is made by the mobile node itself as in some situations, there might be heterogeneous/different operators for the MEC servers.

The authors in [21] proposed a code offloading framework for offloading in the mobile fog environment. The proposed method determines which part of the application should be offloaded and takes an offloading decision considering the current state of the edge node resource by modelling the problem as MDP and training it using the Q-learning approach. Also, their algorithm supports the mobility of the user by migrating the offloaded part from one node to another. Their main goal was to minimize the delay of the offloaded applications. In this work, the feasible sites for offloading are the mobile fog in close proximity, the adjacent mobile fog, or the remote public cloud. In our OST approach, we assume that the mobile node can only offload to an edge server and there are a set of feasible MEC servers to offload.

A context-sensitive offloading system using machine-learning classification algorithms was proposed in [22]. The proposed system integrates middleware, machine learning classification algorithms, and a robust profiling system. The authors considered whether a task should be done locally or at the edge node. Our proposed work can help such a system to decide which server to be used and what time the offloading should occur once the decision is made by such algorithms.

The work in [23] proposed an offloading decision algorithm for vehicles. The proposed algorithm decides which part of the application should be done locally or in the cloud based on the task requirements. A heuristic mechanism for partitioning and scheduling the application between the vehicular and the cloud is proposed. This work is designed for cloud-based architecture and focused on the decision regarding which part of the application should be offloaded.

In [19], we proposed a time-optimized task offloading decision making in MEC to minimize the total delay when offloading task/data. We compared the proposed method to the optimal solution. The proposed model was very close to the optimal. In this work, we extend such model by comparing it to other offloading models and use a real server utilization data set along with a real car trace.

3. Delay-Tolerant Sequential Decision Making for Task Offloading in MEC

OST is about deciding when to carry out an action on the basis of random variables observed in sequence for the purpose of increasing the potential payoff or reducing the potential costs [24]. The secretary problem (SP), the house selling (HS) problem or the fair coin problem are some of the models with varying aims that can be classified under the OST [24]. Our model sees the problem of sequential offloading decision-making as a finite horizon OST problem because the offloading decision must be made within n observations, as is the assumption in [2].

Based on the OST principles, we propose a delay-tolerant time-optimized task offloading decision making rules to minimise the total delay the mobile node experiences when offloading a computing task. In particular, we consider a MEC system as shown in Figure 1, where a mobile node can offload data to perform a computing task on a specific MEC server. The offloaded tasks can be perception tasks as mentioned earlier, computing tasks over offloaded data e.g., image recognition, image processing, data correlation analysis, inferential and predictive analytics [25], statistical learning models building and/or model selection [14,26]. The mobile node can be a smart vehicle as proposed in [2] or smart phone used by the passenger of the vehicles. For each MEC server, at each time instance, there is a temporal load associated with it. Such a load refers to the number of users' requests the server is processing.

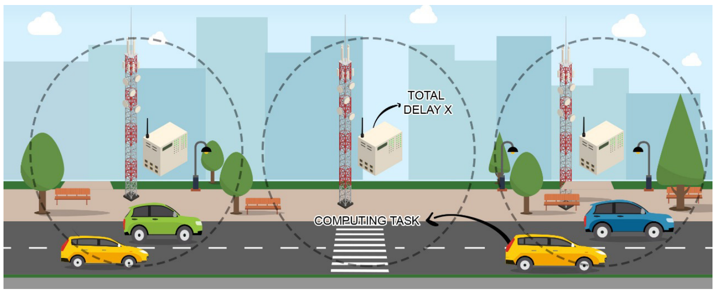

Figure 1. Vehicular network (VN) with mobile edge computing (MEC) servers.

For each server, there is also a transmission delay from the mobile node to the server, which represents the expected time for uploading the data to the MEC server and receiving the processed data/analytics results back.

The execution delay for a task (hereinafter is referred to as **total delay**) on the MEC server, X_o, incorporates, as stated in [4]:

1. Transmission duration of the offloaded data to the server X_{ot}.
2. Processing time at the MEC server X_{op}.
3. Time spent to receive the processed data from the MEC X_{or}.

We consider the case when the expected total delay at the MEC server $\mathbb{E}[X_o]$ is lower than the expected delay when executing the tasks locally on the mobile device $\mathbb{E}[X_l]$, i.e., $\mathbb{E}[X_o] < \mathbb{E}[X_l]$. In other words, the mobile node wishes to offload tasks and data for processing to a MEC server as it does not have the computational capabilities to do so and/or sufficient energy for such tasks. Initially, the mobile node is observing the total delay for each server and it can either offload or postpone the offloading in light of having a minimized total delay. We then formalize our problem as follows:

Problem 1. *The mobile node should find the best time instance t^* (server) such that the expected total delay $\mathbb{E}[X_o]$ is minimized, i.e., the optimal stopping time t^* such that the following infimum is attained:*

$$ess \inf_{t} \mathbb{E}[X_{ot}]. \tag{1}$$

Hence, our expected minimum delay is $\mathbb{E}[X_{ot}]$ with $t = \inf\{t : X_{ot} < \mathbb{E}[X_{ot+1}]\}$.

We define a discrete-time dynamic system, which expresses the evolution of a scalar variable, hereinafter referred to as the system's "state" z_k, under the influence of decisions made at discrete instances of time associated with the kth observation. A state z_k summarizes past information that is needed for future optimization. By writing that the system is at state $z_k = X^*_{k-1}$ at $k \leq n$, we mean that the decision maker has not offloaded the data to a MEC server. By writing that the system is at state $z_k = z_T$, we mean that the decision maker has already offloaded the data to a MEC server, $k \leq n$, where z_T is defined as the terminating state. We take $z_1 = 0$ (a fictitious state). With these conventions (adopted from [27] and applied in [28]), the system equation (the mechanism by which the system is updated) has the form:

$$z_{k+1} = \begin{cases} z_T, & \text{if } z_k = z_T \text{ (stop).} \\ X^*_k, & \text{otherwise (continue).} \end{cases} \tag{2}$$

Let $J_k(z_k)$ be the optimal server to offload data/task to. The Bellman's equation for this system is then:

$$J_n(z_n) = z_n \tag{3}$$

for $k = n$, and

$$J_k(z_k) = \min\left[(1+r)^{n-k} z_k, \mathbb{E}[J_{k+1}(X^*_k)]\right] \tag{4}$$

for $k = 1, \ldots, n-1$.

Note that $\mathbb{E}[J_{k+1}(X_k^*)] = \mathbb{E}[J_{k+1}(z_{k+1})]$. The $r \in (0,1)$ parameter is a delay factor, which prompts the decision maker to delay its optimal decision. In our model, a smaller r value denotes that the decision maker will skip a relatively small number of observations before proceeding with a offloading decision. The term $(1+r)^{n-k} z_k$ in (4) denotes the risk if the offloading happens at k and $\mathbb{E}[J_{k+1}(X_k^*)]$ denotes the expected risk if the decision maker continues the observation process. Hence, it is optimal to stop at stage k if

$$z_k \leq a_k = \frac{\mathbb{E}[J_{k+1}(X_k^*)]}{(1+r)^{n-k}} \tag{5}$$

else, it is optimal to continue. The optimal stopping rule is determined by the scalar values $a_1, a_2, \ldots, a_n$ through which the mobile node decides either to offload or not. Specifically, the optimal offloading rule of the mobile node is:

Optimal Task Offloading Rule: *stop the observation and offload the data at the k-th MEC server if* $z_k \leq a_k$*; otherwise continue the observation if* $z_k > a_k$*.*

In our context, the optimal stopping rule states that the offloading decision (stopping) should happens right after receiving the k-th observation for which the expected total delay $X_{ok} \leq a_k$. Note that, X_{ok} can refer to the expected total delay of the server K. This also can mean the expected total delay of the server at time k. This depends on the deployment density of MEC servers. For example, if the user is passing by a set of servers (high mobility), X_{ok} can refer to the total delay of server K. On the other hand, if the user is moving within the range of one server, X_{ok} refers to the expected total delay of the server at time k. The scalar variable a_k values are calculated as shown in Lemma 1.

Lemma 1. *The scalar values $a_1, a_2, \ldots, a_n$ can be calculated once through the method of backward induction for $k = n$ to 1 from*

$$a_k = \frac{1}{1+r}\left(a_{k+1}(1 - F(a_{k+1})) + \int_0^{a_{k+1}} u dF(X)\right) \tag{6}$$

$$a_n = \frac{1}{1+r}\int_0^1 u dF(X) = \frac{1}{1+r}\mathbb{E}[X] \tag{7}$$

where $F(X) = P(X^* \leq X)$ is the cumulative distribution function of X^*.

Proof. Consider the function:

$$F_k(z_k) = \frac{J_k(z_k)}{(1+r)^{n-k}}, z_k \neq z_T$$

and then $\mathbb{E}[F_{k+1}(X)] = \mathbb{E}[J_{k+1}(X)]/(1+r)^{n-k-1} = a_k/(1+r)^{-1}$ or $a_k = (1+r)^{-1}\mathbb{E}[F_{k+1}(X)]$. Hence, we have $F_n(z_n) = J_n(z_n) = z_n$ and by placing $\mathbb{E}[J_{k+1}(X)]$ with $\mathbb{E}[F_{k+1}(X)](1+r)^{n-k-1}$ in (4), for $k = 1, \ldots, n-1$, we get

$$\begin{aligned} F_k(z_k) &= \min\left[z_k, (1+r)^{-1} E[F_{k+1}(X)]\right] \\ &= \min(z_k, a_k). \end{aligned}$$

The a_k value is recursively calculated as follows:

$$\begin{aligned} a_k &= (1+r)^{-1}E[F_{k+1}(X)] \\ &= (1+r)^{-1}E[\min(X, a_{k+1})] \\ &= \frac{1}{1+r}\left(\int_0^{a_{k+1}} u dF(X) + \int_{a_{k+1}}^{1} a_{k+1} dF(X)\right) \\ &= \frac{1}{1+r}\left(a_{k+1}(1 - F(a_{k+1})) + \int_0^{a_{k+1}} u dF(X)\right). \end{aligned}$$

The scalar values $a_1, a_2, \ldots, a_n$ can be inductively obtained for $k = n$ down to 1, and the terminal condition is $a_n = \frac{1}{1+r}\mathbb{E}[X]$. $\square$

Before calculating the stopping rules, we need to know the probability distribution of the random variable (total delay X_o in our case). For example, if the total delay X_o, including X_{ot}, X_{op} and X_{or}, is uniformly distributed, we would first get the cumulative distribution using:

$$F(c \leq X \leq d) = \int_c^d f(x)dx = \frac{1}{b-a}dx = \frac{d-c}{b-a} \tag{8}$$

with $a \leq c < d \leq b$. After that, we calculate the expected delay of the load using:

$$\mathbb{E}(X) = \int_a^b f(x)dx = \int_a^b \frac{x}{b-a}dx = \frac{b-a}{2}, \tag{9}$$

with $a \leq X \leq b$.

For example, if we have an idea that the X_o in a specific time interval is uniformly distributed between $a = 1$ and $b = 20$ s by studying the previous X_o of the same servers at similar time, we start obtaining the scalar variable a_n and a_k by the backward induction method using Equations (6) and (7).

The scalar decision values $\{a_k\}_{k=1}^n$ are illustrated in Figure 2. Now, it is optimal to offload at time k, i.e., on the k-th MEC, if the total delay $X_o \leq a_k$; otherwise, continue. In other words, it is optimal to stop if the value of the total delay is under the curve shown in Figure 2. By doing this, we are minimizing the expected total delay.

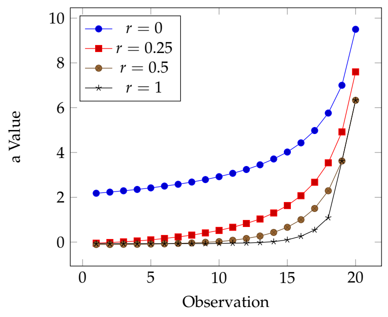

Figure 2. The values of the decision scalars $\{a_k\}_{k=1}^n$ for $n = 20$ observations based on a uniform distribution of the load for different delay factors r.

Figure 3 shows the a values when the random variable, i.e., X_o, is normally distributed.

When the random variable is normally distributed with known mean and standard deviation, we follow the same steps as we did with the uniform distribution in order to get the a values. We get the cumulative distribution function of the normal distribution using:

$$F(X) = \int_{-\infty}^{x} f(x)dx = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2}. \tag{10}$$

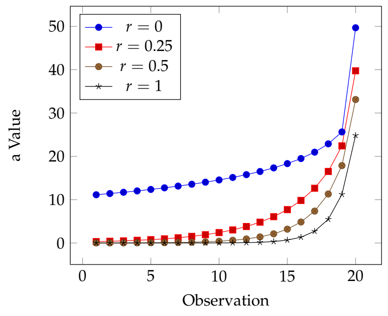

Figure 3. The values of the decision scalars $\{a_k\}_{k=1}^{n}$ for $n = 20$ observations based on a normal distribution of the load for different delay factors r.

To summarize, the first procedure of the proposed model is to obtain the scalar values $\{a_k\}_{k=1}^{n}$ considering the above-mentioned model. In a real world scenario, the values of a can be calculated and distributed by the operator of the MEC servers to the mobile nodes. After that, the mobile node will run the lightweight process shown in Algorithm 1. The mobile node observes the total delay X_o of MEC servers, which is provided by each of them upon request. Then, the mobile node offloads their tasks/data to the first MEC server k which has total delay X_o less than or equal to the variable a_k. Based on this optimal offloading rule, the mobile node is more likely to minimize the total delay X_o. If no offloading decision is made after observing n MEC servers, the mobile node offloads the tasks/data to the n-th MEC server, since no recall is allowed in the work of this paper.

Algorithm 1 Optimal tasks/data offloading rule.

Input: Decision scalar values $a_1, a_2, ..., a_n$

Output: Decision of which MEC server to offload

```
Offload ← FALSE
for $k = 1 : n$ do
  if $X_{o,k} \leq a_k$ then
    MEC-Server ← $k$ ;
    Offload ← TRUE; break;
  end if
end for
if Offload == FALSE then
  MEC-Server ← $n$;
end if
Offload tasks/data to the MEC-Server;
```

4. Performance Evaluation

4.1. Data Sets

To simulate the movements of the mobile nodes, we used the real data set of taxi cabs' movements in Rome [29]. The data set contains GPS coordinates of 320 taxis collected over 30 days. For each row in this data set, we have the cab-id, date/time and GPS coordinates of the current location. Figure 4 shows the movements of the cars on a map of Rome. The focus was on 50 cars for one minute of movements. To simulate the delay of the MEC servers, we used a real data set for server utilisation in a data centre taken from the Alibaba Cluster Trace Program [30]. The data set contains the CPU and memory utilisation for more 1000 machines at different times. The CPU and memory utilisation have been summed to represent the delay the user may experience when offloading a task. Thus, for each movement in the mobility trace, CPU and memory utilisation is checked in the servers data set. The goal was to select the server with minimum utilisation.

It is worthwhile mentioning that the use of mobility trace is not for studying the mobility of users. It was used in this experiment to use each time movement as a location or time to check for a server/time to offload. In other words, it is about the time instances for each user in the mobility trace which is used to check the server load. In real world applications, one might divide the time into intervals and for each interval, check the delay. Additionally, the aim of using the data from real servers is to have an idea of how the proposed model can be applied to real servers' load, as the proposed model is based on the data of the servers' load or delays and assuming the MEC servers will be similar to the servers deployed in large centralized data centres.

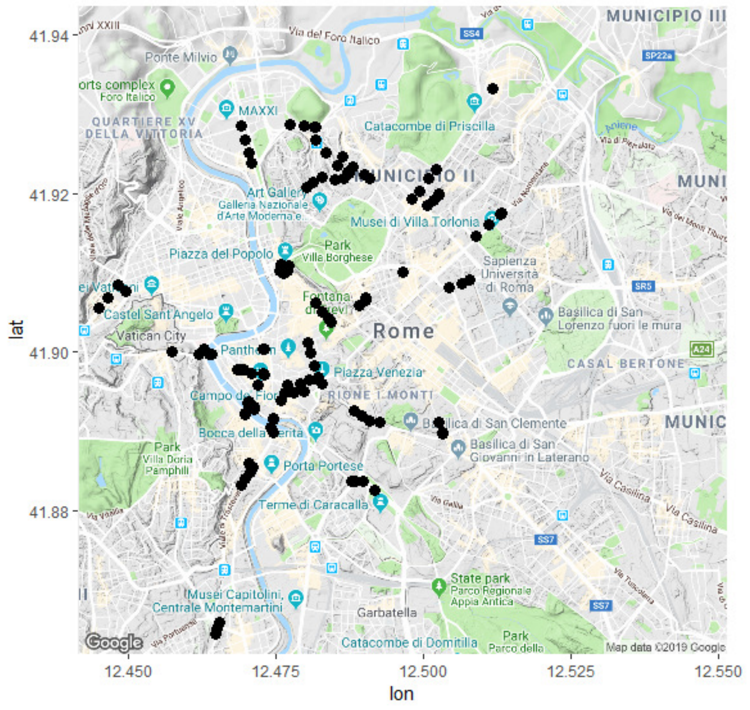

Figure 4. Taxis trajectories in Rome.

Since the server data set contains a large number of rows, we make mapping between the cab's data set and the server data set. As aforementioned, the aim was to have a server utilisation for each movement in the mobility trace. Therefore, two types of mapping are used, namely, (1) random mapping, and (2) consecutive time-based mapping. In the random mapping, for each movement in the mobility trace, a server and its utilisation were randomly selected, thus obtaining a data set that contains different servers for different time utilisation. This mapping is representative of situations such as the high-density deployment of MEC servers, high variations of MEC servers' load or in high-speed movements. It was attempted to optimise the MEC server selection for this type of mapping. In the second mapping method, for each car movement, we select a consecutive time-based utilisation from one server. This is representative of when there are fewer MEC servers or when the mobile node is slower. In other words, the mobile node might be moving within the range of the MEC server and trying to select the best time to offload. For example, Tables 1 and 2 demonstrate the movements of the cab with the id number 156 in both settings, that is, random mapping and consecutive time-based mapping. It is clear that in the random mapping in Table 1, there are different servers at different times. In Table 2, we can see that the same server was checked (m_1934) but at different times.

Table 1. A sample of the data set in the random mapping.

Cab-ID	Location	Machine Name	Time	CPU and Memory Utilization
156	(41.88367, 12.48778)	m_1939	03:24:20	(41,92)
156	(41.88360, 12.48844)	m_1936	01:06:50	(43,92)
156	(41.88376, 12.48968)	m_1941	16:20:00	(44,92)
156	(41.88251, 12.49179)	m_1941	13:23:50	(46,49)

Table 2. A sample of the data set in the onsecutive time-based mapping.

Cab-ID	Location	Machine Name	Time	CPU and Memory Utilization
156	(41.88367, 12.48778)	m_1934	00:42:40	(34,89)
156	(41.88360, 12.48844)	m_1934	00:43:00	(35,89)
156	(41.88376, 12.48968)	m_1934	00:44:00	(27,88)
156	(41.88251, 12.49179)	m_1934	00:46:40	(24,87)

4.2. Performance Assessment

We compare our OST-based offloading model with the random selection model (Random), and the p-stochastic model (p-model) for different probabilities p. In Random, for each user, we randomly select a server to offload the task. In the p-model, for each server (time if we have the same server), we assign a probability of offloading $p \in \{0.1, 0.3, 0.5, 0.8\}$. In each user's movements, each server has probability p of being selected for task offloading (not selected with probability $1 - p$). If a server is selected, we stop the process and consider that server for offloading. If there was no server selected, we select the last server. For example, when p = 0.8, the user will offload early at the start of each period. This is to simulate the situation when the user offloads the computing task to the first server/time. When p = 0.1, on the other hand, the user is more likely to delay the offloading or offload at the last server/time. We compare the results from all models with the ground truth, i.e., the optimal model, in which we select the server with the minimum total delay for each user. For example, the optimal in Table 2 is to offload at 00:46:40 with total delay (simulated as server utilization) 111 $(24 + 87)$ ms. The closer a model is to optimal, the better the model performs in terms of the task offloading decision. We run all models on each car (user) for evaluation. In short, for each car, we select a server for offloading as suggested by each model. We then take the average total delay for all selected servers per model.

4.3. Results

Figure 5a shows the average total delay (server utilisation) suggested by each model for all users. It can be seen that the OST model is the closest model to the optimal. The proposed model is only 3 ms different from the optimal and the proposed model is higher than the optimal by only 2.8% as shown in Figure 5b,c. We also have similar results when we adapt the time based mapping as shown in Figure 6a–c. In the consecutive time-based mapping in Figure 6a, it can be observed that the total average delay obtained by the optimal, that is, the minimum total delay is close to other models as the data used for this experiment is for one server for different times. However, a lower difference is still achieved compared to the optimal when applying the OST-based model.

In reality, the mobile node would normally offload to the first server or at the first time. A simulation for such a case is the p-model with $p = 0.8$. This is clear in Figures 5a and 6a where the p-model when $p = 0.8$ has the lowest offloading times (offload earlier than other models). We can see from the results in both experiments that the p-model with $p = 0.8$ is too far from the optimal. In other words, our results show that going with the first server (time) is not a good idea. Moreover, which server/time is optimal is unknown and not provided to the mobile node, meaning that having the OST-based model implemented in the mobile node can achieve a total delay close to the optimal.

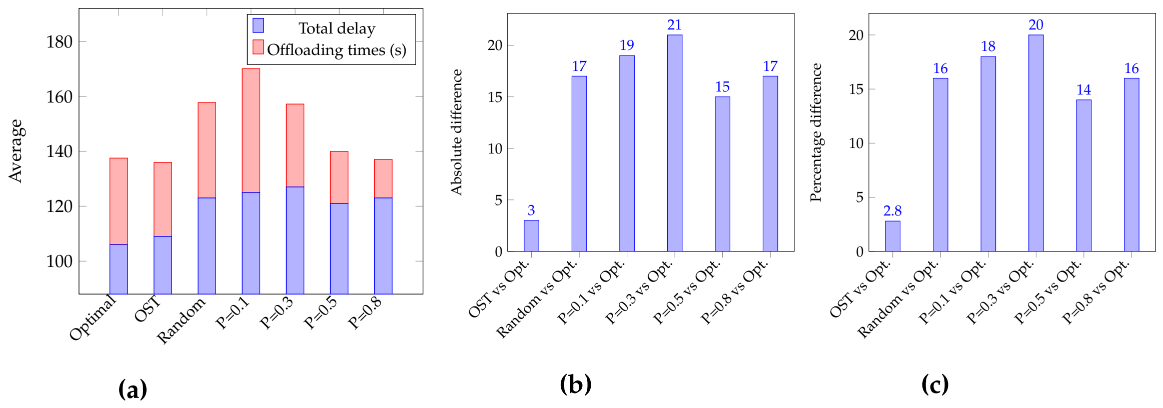

Figure 5. (**a**) Average total delay and average offloading times suggested by each model; (**b**) absolute difference between the Optimal and all models; (**c**) percentage difference between the Optimal and all models in Random mapping results.

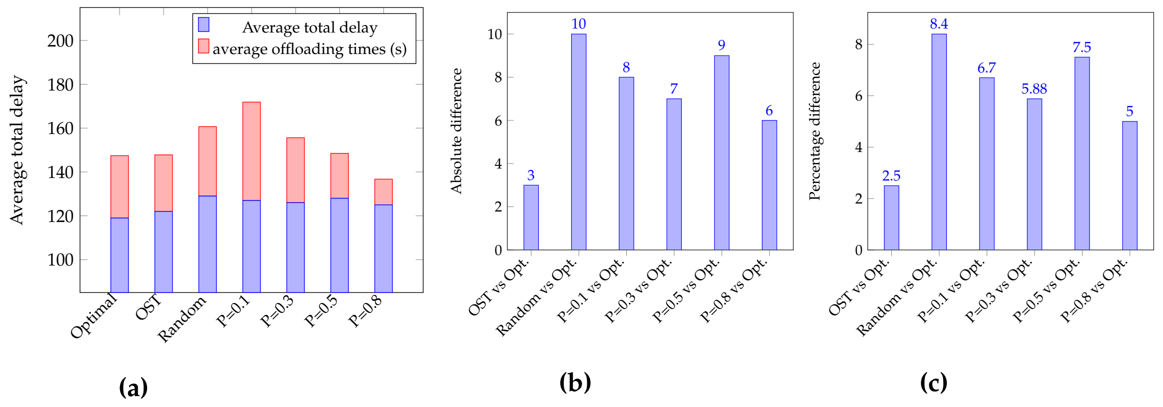

Figure 6. (**a**) Average total delay and average offloading times suggested by each model; (**b**) absolute difference between the Optimal and all models; (**c**) percentage difference between the Optimal and all models in Consecutive time-based mapping.

5. Discussion

With the recent advances in mobile nodes and the applications including smartphones, smart vehicles and drones, and the envision deployment of MEC servers in 5G technology, the proposed model can be adapted and implemented in mobile nodes. The model can be adapted by applications developers or MEC server operators. Such a model can be transferred to a real model for MEC applications with the cooperation between the MEC server operators and the user. From the operator's perspective , there can be historical data about what probability distributions the load of server and the total delay will be in different locations within a city. Based on this, the threshold of when to offload or which server to offload to can be estimated and provided to the mobile nodes.

Moreover, with different requirements of mobile nodes' applications, delay is only one instance of the requirements of some applications. There might be other requirements that need to be considered apart from the delay. In other words, the proposed model can also work well for other requirements. For example, in some applications, the mobile node might be looking for higher bandwidth such as offloading collected data. In such a case, the proposed model can be modified and adapted easily. In addition, the proposed model can run efficiently in the mobile nodes as it has linear computational time. The mobile node needs to do a liner search for the server/time that is less than or equals the threshold a_k.

6. Materials and Methods

We used the R Programming Environment to combine the data sets, implement the proposed model and apply it to the data sets. As mentioned in the Performance Evaluation section, we used two data sets in order to have one combined data set [29,30]. The data sets used for experiments after the mapping between the data sets and the code are available in the link http://dx.doi.org/10.5525/gla.researchdata.896.

7. Conclusions

In this work, we have extended our previous work [19] and compared the OST-based model proposed previously with two offloading methods: Random and P-model. The results show that the proposed model is the closest model to the optimal. A limitation of this work is that the number of MEC servers or times the user passes by has to be known. Thus, in the future work, we will consider the case where the mobile node does not have an idea about the number of MEC servers and will also consider the case where there is no information about the load of the MEC servers.

Author Contributions: The authors contributed equally to this work.

Funding: This research has been supported in part by Al-Baha University, Saudi Arabia, and the Saudi Arabian Cultural Bureau in the UK; by the UK Engineering and Physical Sciences Research Council (EPSRC) projectsEP/N033957/1 and EP/P004024/1; and by the European Cooperation in Science and Technology (COST) ActionCA 15127: RECODIS – Resilient communications and services.

Acknowledgments: The authors would like to thank Al-Baha University, Saudi Arabia, and the Saudi Arabian Cultural Bureau in the UK for their support and encouragement.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MEC	Mobile edge computing
OST	Optimal stopping theroy
QoS	Quality of service
IoT	Internet of Things
ETSI	European Telecommunications Standards Institute
ISG	Industry Specification Group
AR	Augmented reality
VN	Vehicular network
UAV	Unmanned aerial vehicles

References

1. Dinh, H.T.; Lee, C.; Niyato, D.; Wang, P. A survey of mobile cloud computing: Architecture, applications, and approaches. *Wirel. Commun. Mob. Comput.* **2013**, *13*, 1587–1611. [CrossRef]
2. Zhang, K.; Mao, Y.; Leng, S.; He, Y.; Zhang, Y. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading. *IEEE Veh. Technol. Mag.* **2017**, *12*, 36–44. [CrossRef]
3. Zhu, S.; Gui, L.; Chen, J.; Zhang, Q.; Zhang, N. Cooperative Computation Offloading for UAVs: A Joint Radio and Computing Resource Allocation Approach. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 74–79.
4. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 1628–1656. [CrossRef]
5. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [CrossRef]
6. Satyanarayanan, M.; Bahl, V.; Caceres, R.; Davies, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Comput.* **2009**, *8*, 14–23. [CrossRef]

7. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 13–16.
8. Hu, Y.C.; Patel, M.; Sabella, D.; Sprecher, N.; Young, V. Mobile edge computing—A key technology towards 5G. *ETSI White Pap.* **2015**, *11*, 1–16.
9. IBM and Nokia Siemens Networks Announce World's First Mobile Edge Computing Platform. 2013. Available online: https://www-03.ibm.com/press/us/en/pressrelease/40490.wss (accessed on 24 September 2019).
10. Roman, R.; Lopez, J.; Mambo, M. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Gener. Comput. Syst.* **2018**, *78*, 680–698. [CrossRef]
11. Akherfi, K.; Gerndt, M.; Harroud, H. Mobile cloud computing for computation offloading: Issues and challenges. *Appl. Comput. Inform.* **2016**, *14*, 1–16. [CrossRef]
12. Dolezal, J.; Becvar, Z.; Zeman, T. 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), Berlin, Germany, 31 October–2 November 2016; pp. 1–7.
13. Braud, T.; Bijarbooneh, F.H.; Chatzopoulos, D.; Hui, P. Future networking challenges: The case of mobile augmented reality. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; pp. 1796–1807.
14. Harth, N.; Anagnostopoulos, C. Edge-centric Efficient Regression Analytics. In Proceedings of the International Conference on Edge Computing, San Francisco, CA, USA, 2–7 July 2018; pp. 93–100.
15. Ko, H.; Lee, J.; Pack, S. Spatial and Temporal Computation Offloading Decision Algorithm in Edge Cloud-Enabled Heterogeneous Networks. *IEEE Access* **2018**, *6*, 18920–18932. [CrossRef]
16. Kekki, S.; Featherstone, W.; Fang, Y.; Kuure, P.; Li, A.; Ranjan, A.; Purkayastha, D.; Jiangping, F.; Frydman, D.; Verin, G.; et al. MEC in 5G networks. *ETSI White Pap.* **2018**, *28*, 1–28.
17. Le Tan, C.N.; Klein, C.; Elmroth, E. Location-aware load prediction in edge data centers. In Proceedings of the 2nd FMEC, Valencia, Spain, 8–11 May 2017; pp. 25–31.
18. Zhang, J.; Letaief, K.B. Mobile Edge Intelligence and Computing for the Internet of Vehicles. *arXiv* **2019**, arXiv:1906.00400.
19. Alghamdi, I.; Anagnostopoulos, C.; Pezaros, D.P. Time-Optimized Task Offloading Decision Making in Mobile Edge Computing. In Proceedings of the 2019 Wireless Days (WD), Manchester, UK, 24–26 April 2019; pp. 1–8.
20. Ur Rehman, M.H.; Sun, C.; Wah, T.Y.; Iqbal, A.; Jayaraman, P.P. Opportunistic computation offloading in mobile edge cloud computing environments. In Proceedings of the 2016 17th IEEE International Conference on Mobile Data Management (MDM), Porto, Portugal, 13–16 June 2016; Volume 1, pp. 208–213.
21. Alam, M.G.R.; Hassan, M.M.; Uddin, M.Z.; Almogren, A.; Fortino, G. Autonomic computation offloading in mobile edge for IoT applications. *Future Gener. Comput. Syst.* **2019**, *90*, 149–157. [CrossRef]
22. Junior, W.; Oliveira, E.; Santos, A.; Dias, K. A context-sensitive offloading system using machine-learning classification algorithms for mobile cloud environment. *Future Gener. Comput. Syst.* **2019**, *90*, 503–520. [CrossRef]
23. Ashok, A.; Steenkiste, P.; Bai, F. Vehicular Cloud Computing through Dynamic Computation Offloading. *Comput. Commun.* **2018**, *120*, 125–137. [CrossRef]
24. Ferguson, T. Optimal Stopping and Applications. 2019. Available online: http://www.math.ucla.edu/~tom/Stopping/Contents.html (accessed on 1 November 2018).
25. Harth, N.; Anagnostopoulos, C.; Pezaros, D. Predictive intelligence to the edge: Impact on edge analytics. *Evol. Syst.* **2018**, *9*, 95–118. [CrossRef]
26. Anagnostopoulos, C.; Kolomvatsos, K. Predictive intelligence to the edge through approximate collaborative context reasoning. *Appl. Intell.* **2018**, *48*, 966–991. [CrossRef]
27. Bertsekas, D.P. *Dynamic Programming and Optimal Control*; Athena Scientific: Belmont, MA, USA, 1995; Volume 1.
28. Anagnostopoulos, C.; Hadjiefthymiades, S. Intelligent trajectory classification for improved movement prediction. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 1301–1314. [CrossRef]

29. Bracciale, L.; Bonola, M.; Loreti, P.; Bianchi, G.; Amici, R.; Rabuffi, A. CRAWDAD dataset roma/taxi (v. 2014-07-17). 2014; doi:10.15783/C7QC7M. Available online: https://crawdad.org/roma/taxi/20140717 (accessed on 1 March 2019).
30. Alibaba Cluster Trace Program cluster-trace-v2018. 2018. Available online: https://github.com/alibaba/clusterdata/blob/master/cluster-trace-v2018/trace_2018.md (accessed on 1 April 2019).