Time-Optimized Task Offloading Decision Making in Mobile Edge Computing

Ibrahim Alghamdi
School of Computing Science
University of Glasgow, UK
i.alghamdi.1@research.gla.ac.uk

Christos Anagnostopoulos
School of Computing Science
University of Glasgow, UK
christos.anagnostopoulos@glasgow.ac.uk

Dimitrios P. Pezaros
School of Computing Science
University of Glasgow, UK
dimitrios.pezaros@glasgow.ac.uk

***Abstract*—Mobile Edge Computing application domains such as vehicular networks, unmanned aerial vehicles, data analytics tasks at the edge and augmented reality have recently emerged. Under such domains, while mobile nodes are moving and have certain tasks to be offloaded to Edge Servers, choosing an appropriate time and an ideally suited server to guarantee the quality of service can be challenging. We tackle the offloading decision making problem by adopting the principles of Optimal Stopping Theory to minimize the execution delay in a sequential decision manner. A performance evaluation is provided by using real data sets compared with the optimal solution. The results show that our approach significantly minimizes the execution delay for task execution and the results are very close to the optimal solution.**

***Index Terms*—Mobile edge computing, tasks offloading, optimal stopping theory, sequential decision making.**

I. Introduction

Over the past years, new mobile devices and applications, with different functionalities and uses, such as drones, Vehicular Networks (VN) and very advanced smart phones have emerged. This development has enabled such devices to launch applications such as Augmented Reality (AR), intensive contextual data processing, intelligent vehicle control, traffic management, data mining applications and interactive applications. Although these mobile devices have computing and communication capabilities to run such applications, they still cannot efficiently handle them. The main reason for this limitation is that these applications require significant processing in relatively short time and consume significant battery. Such limitations have motivated the emergence of the Mobile Cloud Computing (MCC) [14] paradigm.

MCC allows mobile devices to benefit from Cloud Computing (CC) resources to offload and execute their tasks, applications and/or collected data. A mobile node may use the computing and storage resources of a remote data-centre via accessing a cellular network or Wi-Fi connection. Task offloading from mobile nodes to the cloud leads to extend the battery lifetime of the mobile nodes and save computational resources, thus, enabling advanced applications to users and providing higher data storage capabilities [14]. However, using CC for task/data offloading of mobile applications introduces significantly latency and adds more load to the radio and backhaul of the mobile networks [14].

To deal with the disadvantages of MCC, the Mobile Edge Computing (MEC) paradigm has emerged. The rationale architecture of this concept is to offer cloud services *closer* to mobile devices by placing many data-centres at the edge of the network. The network edge refers to different places e.g., mobile network at the Base Station (BS), or indoor places such Wi-Fi and 3G/4G access points [17].

Motivation & Challenge: An essential use case of MEC is the computing task/data offloading. Computation offloading is the task of sending a computation task and data to a remote server for delegating this computation [1]. As the new emerging applications require intensive computation processes, computation offloading provides a solution to overcome the limitation of the mobile device. Examples of applications that can benefit from computing offloading are mobile AR [6], gaming, Internet of Things (IoT) applications, data analytics tasks at the edge [15], VN [25] and Unmanned Aerial Vehicles (UAV) [28] . A study showed the benefit of using offloading for the Percipio AR application on a real MEC testbed [7]. The study showed that the computation offloading reduces latency up to 88% and energy consumption of mobile devices up to 93%.

Computation offloading faces several challenges, the most significant ones being: (i) *the decision of when to offload tasks/data to a MEC server* and (ii) *mobility patterns/behaviour* of the users in such MEC environments. The decision making of tasks/data offloading is of high importance as it is expected to directly affect the Quality of Service (QoS) of the user application including the inherent latency due to the offloading process. Therefore, different parameters have to be considered *when* to decide to offload tasks and/or data including: the current MEC server load and the transmission / communication status between the mobile node and the MEC server. The decision can be spatial or temporal as stated in [10]. The spatial decision refers to realising the computing tasks *locally* at the mobile device, in the cloud, or at the edge server [11]. The temporal decision refers to a situation *where* it is advisable to *optimally delay* the tasks/data offloading due to the current expected cost in terms of the transmission delay and the processing delay at the MEC server, e.g., from the user prospective, the Wi-Fi connectivity is low, or from the network operator perspective, the server is fully loaded, thus, expecting high latency for delivering the outcome of the delegated computation tasks to the user.

In this work, we propose a time-optimised task offloading decision algorithm in MEC environments by finding the optimal task offloading time taking into consideration the expected transmission delay and the expected processing delay for the task execution at the MEC server. In our context, while the mobile node is sequentially roaming (connecting) through a set of MEC servers, the mobile node has to locally and autonomously decide which server should be used for offloading the data to perform the computing task. Such decision takes into account the MEC server load and the transmission delay between the mobile node and the server, which are expected to affect the application QoS. To deal with this sequential decision making problem, we cast the considered offloading decision making problem as an *optimal stopping time problem* adopting the principles of optimality of the Optimal Stopping Theory (OST) [19]. Based on the OST, our contribution is to determine the best strategy of when to choose the server with the minimum execution delay for minimizing the expected holistic cost when offloading. We believe that the adapted algorithm can be suitable for MEC applications such VN [25], UAV [28] or for data mining applications such activity recognition as in [23] as the mobile nodes roam between a set of MEC servers deployed at the edge of the network.

The remainder of this paper is organised as follows: we summarise related work and present our contribution in Section II, while details of the proposed OST-based decision making system are described in Section III. Performance evaluation results are provided in Section IV, and Section V concludes the paper and outlines future research directions.

II. Related Work & Contribution

A number of studies on the decision of offloading data and computing task to an edge node have been conducted aiming to address different challenges. Two main objectives prior work focused on included the minimization of the execution delay and energy consumption. The goal of the former objective is considered in [13] by applying a one-dimensional search algorithm. This approach decides, during each time slot, whether the application *waiting* in a local buffer should be processed locally or at the MEC while minimizing the execution delay. Moreover, it is assumed that the mobile users are not moving before and during the offloading [14]. The offloading decision in this study is to optimally decide whether the offloading should be realised (executing the task at the MEC server) or not (executing the computing task locally at the mobile device) to minimize the execution delay.

Other studies on the offloading decision to an edge node have been conducted [10], [16], [21], [22], [24], [26], [27]. The most relevant work to our approach is ST-CODA [10]. ST-CODA is a spatial and temporal computation offloading decision algorithm that helps the mobile device to decide where and when to offload tasks in consideration of the advantages and disadvantages of the computation nodes and the different transmission costs in edge cloud-enabled heterogeneous networks. Our work is different from ST-CODA because our time-optimised sequential decision refers only to task offloading to the edge servers rather than the cloud. In [10], 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. Other works among these studies considered different scenarios in which the users and the MEC node are mobile and they assume that the MEC server is another mobile device, e.g., when the MEC server is a vehicle that has the computing utilities.

The work in [23] presents an computation offloading strategy for a data mining application, i.e. activity recognition application for mobile devices. Basically, while the user is moving, data is collected from different sources and stored in the mobile device. This data is processed in order to get a decision: whether to be offloaded to an edge server, doing the task locally or in the cloud. When the decision is to offload the data to an edge server, the communication interface in the mobile device starts scanning and gets a list of edge servers. In this work, the authors assume that there is going to be a list of edge servers and then the offloading decision strategy will pick the best one in terms of available resources of these servers. However, if the mobile node is moving, there might be a better server in its path (that is not seen by the communication interface at the moment of doing the scan) as it is moving to different places. Thus, there might be a better MEC server in terms of the execution delay.

The work in [28] proposes a cooperative mobile edge computing to minimize the energy consumption and task execution latency. The considered use case is for UAV application that captures photos or videos regularly for different tasks such as identifying a certain object or acquiring the traffic condition. 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. It is assumed that the decision is made by the system orchestrator. However, in our work, the decision is made by the mobile node itself as in some situations, there might be heterogeneous applications or different operators for the MEC servers. In such case, the decision of selecting the MEC server is only done by the mobile node itself without having a global information or knowing about the next encountered server and its current load. In fact, all we need in our proposed model as we will see in the next sections, is historical data for the servers load in similar time.

In [25], the authors introduce a predictive off-loading framework in vehicular networks. The motivation for this work is the development of vehicles which provide intelligent vehicle control, traffic management, and interactive applications with their equipped computation units and communication technologies. These applications require higher computing

capabilities which are limited in such environment [25]. In short, the scenario the authors consider is that a set of vehicles want to offload tasks to MEC servers that are connected to roadside unites. Two communication methods were proposed: Vehicle to Infrastructure (V2I) and Vehicle to Vehicle (V2V). In the first method, when a task is generated by a vehicle, the vehicle sends the required data through the roadside unit, and get the results back from another predicted roadside unit which is located near to the user at the time when the result is ready. In the other method, when a task is generated, a vehicle sends the required data through other vehicles in the road. The data is submitted to the roadside unit by which the user is more likely to connects when the result is ready. In the first method, the task is always submitted to the first MEC. In our work, we delay the decision in the light of connecting to a better MEC server by applying the concept of OST.

Contribution: To the best of our knowledge, this is the first work to consider the decision offloading strategy as an OST problem under the context where the user is passing by many MEC servers deployed at the network edge with the goal of minimizing the execution delay by providing an offloading strategy of when to select a MEC server. Our work is a general model that can be adapted in different applications and it is easy to apply and implement in the decision maker device.

III. Time-Optimised Offloading Decision Making

A. System Model

We consider a MEC system as shown in Fig. 1, where a mobile device can offload data to perform a computing task on a specific MEC server. For each MEC server, at each time instance, there is a temporal *load* associated with it. Such load refers to the number of user requests the server is processing. The offloaded tasks can be computing tasks over offloaded data e.g., image recognition, image processing, data correlation analysis, inferential and predictive analytics [9], statistical learning models building and/or models selection [8], [3]. The mobile node can be a smart phone as in [23], UAVs as considered in [28] or a VN as proposed in [25]. 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 receive the processed data/analytics results back.

The execution delay for a task (hereinafter is referred to as **total delay**) on the MEC server, D_o, incorporates, as stated in [14]:

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

We consider the case of tasks/data offloading as an incentivisation mechanism so that the expected total delay at the MEC server $\mathbb{E}[D_o]$ is lower than the expected delay when executing the tasks locally on the mobile device $\mathbb{E}[D_l]$, i.e., $\mathbb{E}[D_o] < \mathbb{E}[D_l]$. That said, the mobile node desires 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.

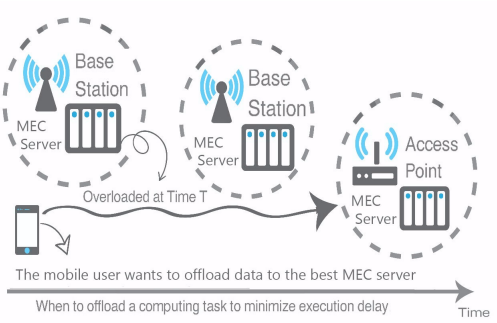

Fig. 1: MEC Scenario for time-optimised tasks/data offloading.

B. Problem Statement

We consider a mobile node who desires to offload data to an edge server, and there are many deployed MEC servers in the user path as shown in Fig. 1 and considered in [25]. The deployed MEC servers are sequentially observed as candidates and the mobile node has to decide which server is the best in terms of total delay to offload the tasks/data. We are challenged to determine the best offloading strategy that minimizes the expected total delay. We abstract this problem into the context of sequential decision making whether to offload the tasks/data to the currently available MEC server or not. The deal with the above-mentioned strategy, we rely on the OST to determine an *optimal offloading rule* for the mobile node. In the following sections, we provide an overview about the OST and, subsequently, we show how we develop the offloading rule in our context.

C. Optimal Stopping Theory

The OST is concerned with the problem of choosing a time to take a given action based on sequentially observed random variables in order to maximize an expected payoff or to minimize an expected cost [19]. There are several models that can be categorised under the OST with different objectives, such as the Secretary Problem (SP), the House Selling (HS) problem, or in problems that aim to maximize the average as in the Fair Coin Problem [19].

In the HS problem, as stated in [19] and applied in [2], the optimal stopping rule is to stop at some stage k^* (optimal stopping time) to minimize the expected cost. A stopping rule problem has a finite horizon if there is a known upper bound n on the number of stages at which one may stop. If stopping is required after observing $S_1, S_2, \ldots, S_n$, the problem has horizon n. In principle, such problems can be solved by the method of backward induction. Since we must stop at stage n, we first find the optimal rule at stage $n-1$. Then, knowing the optimal cost at stage $n-1$, we find the optimal rule at stage $n-2$, and so on back to the initial stage (stage 0). Let $J_k(x)$ $(1 \leq k \leq n)$ represent the minimum expected cost one can obtain starting from stage k. We define $J_n = y_n$ and,

then, inductively for $k = n-1$ backward to $k = 0$, $J_k(x) = \min(y_k, \mathbb{E}[J_{k+1}](x_{k+1}))$. The meaning of this equation is that, at stage k, we compare the cost for stopping, namely y_k, with the cost $\mathbb{E}[J_{k+1}(x_{k+1})]$ that we expect to incur by continuing and using the optimal rule for the stages $k+1$ through n. The optimal cost is, therefore the minimum of these two quantities, and it is optimal to stop at the earliest k when the criterion $y_k \leq \mathbb{E}[J_{k+1}(x_{k+1})]$ holds true, that is

$$k^* = \min_{1 \leq k \leq n} \{k > 0 | y_k \leq \mathbb{E}[J_{k+1}(x_{k+1})]\} \tag{1}$$

We treat the sequential offloading decision problem as a finite horizon OST problem, since we should take an offloading decision within n observations as assumed in [25]. Specifically, when a task is generated to be offloaded to an MEC server, the mobile device can either stop the observation process and offload to the current MEC server k or continue at observation $k+1$ server. All past $s_l^* = 1, \ldots, k-1$ servers that are not accepted by the decision maker to offload to are not recalled in this mode. The last server, i.e., at stage n, must be accepted if every prior server has been rejected for offloading the task. Note that in our future work we are dealing with a recall-mechanism, where the mobile node can recall any past *rejected* MEC server.

The objective is to find an optimal stopping time k^* for stopping with the minimum total delay D_{ok}^* to guarantee the QoS of the decision maker. We define a discrete-time dynamic system, which expresses the evolution of a scalar variable, hereinafter referred to as the systems state x_k, under the influence of decisions made at discrete instances of time associated with the kth observation. A state x_k summarizes past information that is needed for future optimization. By writing that the system is at state $x_k = s_{k-1}^*$ 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 x_k = x_T, we mean that the decision maker has already offloaded the data to a MEC server, $k \leq n$, where x_T is defined as the terminating state. We take $x_1 = 0$ (a fictitious state). With these conventions (adopted from [18]) the system equation (the mechanism by which the system is updated) has the form:

$$x_{k+1} = \begin{cases} x_T, & \text{if } x_k = x_T \text{ (stop).} \\ s_k^*, & \text{otherwise (continue).} \end{cases} \tag{2}$$

Let $J_k(x_k)$ be the optimal server to offload data/task to. The Bellmans equation for this system is then:

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

for $k = n$, and

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

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

Note that $\mathbb{E}[J_{k+1}(s_k^*)] = \mathbb{E}[J_{k+1}(x_{k+1})]$. 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 high number of observations before proceeding with a offloading decision. The term $(1+r)^{n-k}$ denotes the risk if the offloading happens at k and $\mathbb{E}[J_{k+1}(s_k^*)]$ denotes the expected risk if the decision maker continues the observation process. Hence, it is optimal to stop at stage k iff

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

else, it is optimal to continue. The optimal stopping rule is determined by the scalar values $a1, a2, \ldots, a_n$ through which the mobile node decides either to offload or not. Specifically, the optimal stopping rule of the mobile node is:

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

In particular, the optimal stopping rule states that the offloading decision (stopping) should happens right after receiving the k-th observation for which the total delay $D_o \leq a_k$. The scalar variable a_k values are calculated once through the method of backward induction using the equations (6) and (7).

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

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

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

As it was mentioned earlier, the r value is a delay factor which prompts the decision maker to delay or speed its optimal decision.

Before calculating the stopping rules, we need to know the probability distribution of the random variable. For example, if the total delay D_o, including D_{ot}, D_{op} and D_{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:

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

were $a \leq X \leq b$.

For example, if we have an idea that the D_{op} in a specific time interval is uniformly distributed between $a = 1$ and $b = 20$ seconds by studying the previous D_{op} 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 (7) and (6).

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,

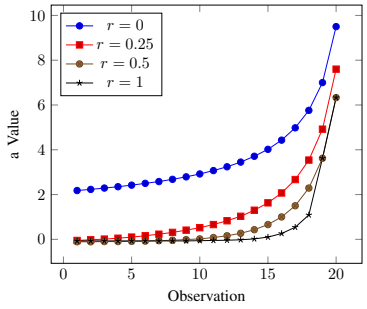

Fig. 2: The values of the decision scalars $\{a_k\}_{k=1}^{n}$ for $n = 20$ observations based on a uniform distribution of the load for different delay factors r.

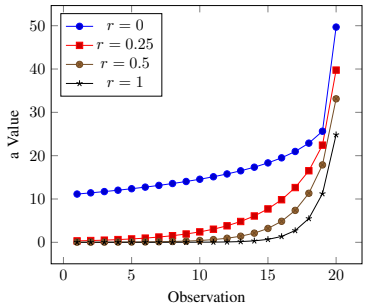

Fig. 3: The values of the decision scalars $\{a_k\}_{k=1}^{n}$ for $n = 20$ observations based on a normal distribution of the load for different delay factors r.

if the total delay $D_o \leq a_k$; 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 3 shows the a values when the random variable, i.e. D_{op}, 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:

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

To summarize, the first procedure of the HS model is to obtain the scalar values $\{a_k\}_{k=1}^{n}$ considering the above-mentioned model. In 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 D_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 D_o less than or equals to the variable a_k. Based on this optimal offloading rule, the mobile node is more likely to minimize the total delay D_o. If no offloading decision is made after observing the 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. In our future research agenda, we consider the recall option.

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 D_{o,k} ≤ 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;
```

IV. Performance Evaluation

A. Real Datasets

We used two real data sets in order to evaluate the proposed optimal strategy and show the benefit of the OST approach in tasks/data offloading. The first data set is a real mobility trace [12] which contained the location (Access Point association) and the timestamp of each wireless card seen on Dartmouth university campus. The second data set is a real-world mobile network traffic data set, published in 2014 by Telecom Italia [4]. It contained mobile users' activities, such as call, SMS, and Internet data connectivity, in terms of the observed traffic for each cell over one hour. We considered the period from 1-11-2013 until 7-11-2013 and adopted the Internet activities data in our simulation.

The considered data sets have been used in the literature to study mobility, offloading decision and resource management in MEC as in [20] and [5]. The movements between the MEC servers are represented by the users' traces obtained from the first data set. The MEC servers with the temporal load for each are represented by the second data set, i.e the Internet traffic. Thus, the two data sets were combined together to form a new data set that was used to simulate a MEC environment. Specifically, we mapped the cells, from the mobile network data set, to the APs in the mobility trace in order to have a load for each AP to simulate the considered scenario. Hence, the APs with the mapped cell and the Internet activity form the MEC servers with a load for each one. Tables I, II and III show a sample of the two data sets and how they were combined for the purpose of our simulation.

For example, if we consider the first row in Table III, we see that the user at the time of 1039045116 (a UNIX timestamp equivalent to 04/12/2002 23.38) connects to AP AcadBldg18AP2 (simulated as a MEC server) with the load of 10.466 (The load of the mapped cell). We assumed that the Internet load is the processing delay D_{op} at the MEC. Also, we add a transmission delay, i.e., $0 < D_{ot}, D_{or} \leq 1$ for each server. Hence, the total delay D_o is the sum of D_{op}, D_{ot}, and

TABLE I: Mobility Trace

Time	Access Point
1043522712	ResBldg55AP2
1043523266	AcadBldg18AP5
1043523287	ResBldg55AP2
1043523792	ResBldg55AP4

TABLE II: Internet Traffic For a Set of Cells

Time	Cell ID	Internet Traffic
06/11/2013 01.00	1	42.68
06/11/2013 01.00	2	42.76
06/11/2013 01.00	3	42.84
06/11/2013 01.00	4	42.45

TABLE III: Combined Data Set

Time1	Time2	Access Point	Cell ID	Cell Internet Load
1039045116	04/12/2002 23.38	AcadBldg18AP2	2158	10.466
1039045116	05/12/2002 22.52	AcadBldg18AP2	2158	19.1095
1040424957	20/12/2002 22:52	AcadBldg10AP15	5395	20.1029
1040424957	20/12/2002 22:55	AcadBldg10AP12	1643	13.2935

D_{ot} and the random variable we care about is D_o. Our goal is to minimize the total delay D_o.

B. Performance Metrics & Assessment

We implemented the Algorithm 1 and applied it to 40 mobile user traces considering an interval time of one day to evaluate our OST-based offloading model. As the total delay for these traces follows Gaussian distribution, we use the mean and the standard deviation of the same interval and applying the model in Section III-C to calculate the values of $\{a_k\}_{k=1}^{n}$. The results are presented in Fig 4 in which we show the difference between the HS model with different *delay factor* values and the optimal solution in seconds. The optimal solution is obtained by selecting the server with the minimum total delay in each interval. In general, the HS model is close to the optimal and this is due to the fact that the mean and the standard deviation is obtained from the same interval. However, this situation might not be realistic in a real world scenario, as sometimes, the mean and the standard deviation of the current time interval will not be available.

A realistic case would be based on the mean and the standard deviation obtained from previous records for the servers' load in the user route. For example, If we have an idea about the route the user is taking as proposed in [25], we look at the mean and the standard deviation of the load of the MEC servers in that path in a previous day (similar time). Thus, for each interval, before running the HS model, we first get the mean and the standard deviation for the servers' load in that interval from the whole trace. For example, let's say that the user in the current interval connects to server A,B,C and D. First, we get the load for these servers from all intervals in the trace, then we calculate the average and the standard deviation. The results are shown in Fig 5. Again, the HS model is very close to the optimal and the difference is less than the previous first experiment. The obvious reason for this good results is that the average and the standard deviation are known (from the trace) and very close to the actual average of each interval. Thus, in real world scenario, if we have good statistics about the load of the MEC servers' load, then the HS model is good to apply and it guarantees to offload to an edge server with minimized total delay.

To see the behaviour of the HS model in one interval for different values of $delay = \{0, 0.25, 0.5, 1\}$, we considered one interval, i.e one day. In this interval, the servers the user connects to are very similar to the ones in the interval before. Thus, it is more realistic to consider the mean and the standard deviation from the previous record. Thus, we considered the mean and the standard deviation from the previous interval. The results are shown in Fig 6. As you can see, the HS is very close to the optimal and the difference is high for $r = 0$ and for $r = 1$. When r value is low, the decision is delayed. This gives the model higher chance of picking the optimal one as observed in Fig 4 and 5. But, for some cases, the optimal can be at the beginning of the interval and sometimes it can be the last server, but after all, the goal of the HS is to minimize the total delay and not to pick the optimal one.

In general, when considering the value of r, we observe two important things. First, in overall, as it can be seen in Fig 2, 3, 4 and 5, when the value of r is high, the model picks the server earlier, and most of the time, the first server is picked. In such case, we have big difference comparing to the optimal solution as shown in Fig 4 and 5. When the value of r is low, we observe that the model is very close to the optimal. Such results give us an indication that speeding the offloading decision to the first servers is not always a good idea to have a minimized delay. Second, sometimes, in addition to finding a minimized delay, we have some constrains or a deadline for the task. In that case, we can adjust the value of r and force the model to make the selection in an early stage.

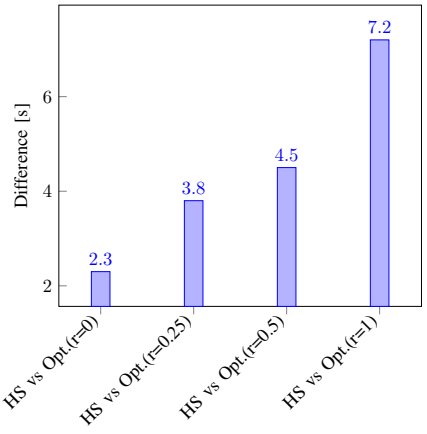

Fig. 4: Overall difference between the Optimal and House Selling model; The σ and μ are taken from the same interval.

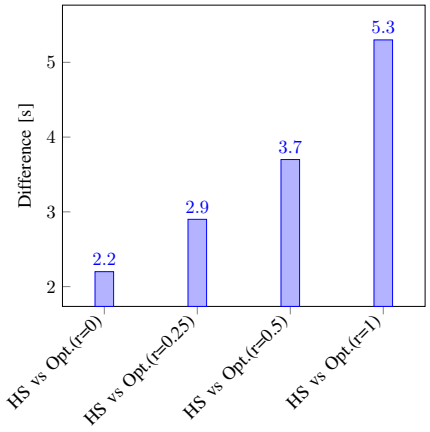

Fig. 5: Overall difference between the Optimal and House Selling model; the σ and μ are taken from the all traces.

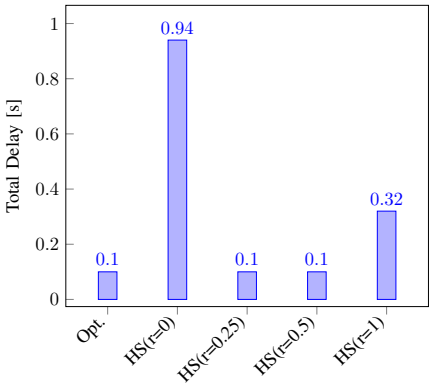

Fig. 6: The Optimal and the House Selling model for one interval for different delay factors r.

V. Conclusions

In this paper, we proposed an Optimal Stopping Theory-based offloading sequential decision strategy for mobile users in Mobile Edge Computing environments. In this context, mobile users sequentially determine *when* and *which* server to offload their tasks/data considering the total delay incurred at each server. Our model is very close to the optimal solution and in sometimes, it obtains the optimal server to offload a task. We believe that this model can be suitable for several MEC applications involving drones or novel/emerging smart phones mobile applications. As future work, we aim to consider the case where there is a deadline by investigating the value of the delay factor r and how it can be adapted for different uses cases.

Acknowledgments

This research has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) projects EP/N033957/1, and EP/P004024/1; by the European Cooperation in Science and Technology (COST) Action CA 15127: RECODIS – Resilient communication and services; by the Huawei Innovation Research Program (Grant No. 300952); and by the EU H2020 GNFUV Project RAWFIE-OC2-EXP-SCI (Grant No. 645220), under the EC FIRE+ initiative.

References

[1] Khadija Akherfi, Micheal Gerndt, and Hamid Harroud. Mobile cloud computing for computation offloading: Issues and challenges. *Applied computing and informatics*, 2016.

[2] Hadjiefthymiades Stathes Anagnostopoulos Christos. Intelligent trajectory classification for improved movement prediction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(10):1301–1314, 2014.

[3] Kolomvatsos Kostas Anagnostopoulos Christos. Predictive intelligence to the edge through approximate collaborative context reasoning. *Applied Intelligence*, pages 48(4):996–991, 2018.

[4] Gianni Barlacchi, Marco De Nadai, Roberto Larcher, Antonio Casella, Cristiana Chitic, Giovanni Torrisi, Fabrizio Antonelli, Alessandro Vespignani, Alex Pentland, and Bruno Lepri. A multi-source dataset of urban life in the city of milan and the province of trentino. *Scientific data*, 2:150055, 2015.

[5] Mathieu Bouet and Vania Conan. Mobile edge computing resources optimization: a geo-clustering approach. *IEEE Transactions on Network and Service Management*, 15(2):787–796, 2018.

[6] Tristan Braud, Farshid Hassani Bijarbooneh, Dimitris Chatzopoulos, and Pan Hui. Future networking challenges: The case of mobile augmented reality. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 1796–1807. IEEE, 2017.

[7] Jakub Dolezal, Zdenek Becvar, and Tomas Zeman. Performance evaluation of computation offloading from mobile device to the edge of mobile network. In *Standards for Communications and Networking (CSCN), 2016 IEEE Conference on*, pages 1–7. IEEE, 2016.

[8] Anagnostopoulos Christos Harth Natascha. Edge-centric efficient regression analytics. In *Edge Computing (EDGE), IEEE International Conference on*. IEEE, 2018.

[9] Pezaros Dimitrios Harth Natascha, Anagnostopoulos Christos. Predictive intelligence to the edge: impact on edge analytics. *Evolving Systems*, pages 9(2):95–118, 2018.

[10] Haneul Ko, Jaewook Lee, and Sangheon Pack. Spatial and temporal computation offloading decision algorithm in edge cloud-enabled heterogeneous networks. *IEEE Access*, 6:18920–18932, 2018.

[11] Christos Anagnostopoulos Kostas Kolomvatsos. In-network decision making intelligence for task allocation in edge computing. In *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2018.

[12] David Kotz, Tristan Henderson, Ilya Abyzov, and Jihwang Yeo. CRAWDAD dataset dartmouth/campus (v. 2009-09-09). Downloaded from https://crawdad.org/dartmouth/campus/20090909/movement, September 2009. traceset: movement.

[13] Juan Liu, Yuyi Mao, Jun Zhang, and Khaled B Letaief. Delay-optimal computation task scheduling for mobile-edge computing systems. In *Information Theory (ISIT), 2016 IEEE International Symposium on*, pages 1451–1455. IEEE, 2016.

[14] Pavel Mach and Zdenek Becvar. Mobile edge computing: A survey on architecture and computation offloading. *arXiv preprint arXiv:1702.05309*, 2017.

[15] Christos Anagnostopoulos Natascha Harth. Edge-centric efficient regression analytics. In *Proceedings of the International Conference on Edge Computing*. IEEE, 2018.

[16] Chhabi Rani Panigrahi, Joy Lal Sarkar, and Bibudhendu Pati. Transmission in mobile cloudlet systems with intermittent connectivity in emergency areas. *Digital Communications and Networks*, 4(1):69–75, 2018.

[17] Milan Patel, B Naughton, C Chan, N Sprecher, S Abeta, A Neal, et al. Mobile-edge computing introductory technical white paper. *White Paper, Mobile-edge Computing (MEC) industry initiative*, 2014.

[18] Goran Peskir and Albert Shiryaev. *Optimal stopping and free-boundary problems*. Springer, 2006.

[19] Shiryaev A. Peskir, G. Optimal stopping and free-boundary problems, brickhauser. 2006.

[20] Yan Shi, Shanzhi Chen, and Xiang Xu. Maga: A mobility-aware computation offloading decision for distributed mobile cloud computing. *IEEE Internet of Things Journal*, 5(1):164–174, 2018.

[21] Wei-Tsung Su and Chao-Yi Kao. Estito: An efficient task offloading approach based on node capability estimation in a cloudlet. In *Wireless Communications and Networking Conference (WCNC), 2017 IEEE*, pages 1–6. IEEE, 2017.

[22] Tram Truong-Huu, Chen-Khong Tham, and Dusit Niyato. To offload or to wait: An opportunistic offloading algorithm for parallel tasks in a mobile cloud. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 182–189. IEEE, 2014.

[23] Muhammad Habib ur Rehman, Chee Sun, Teh Ying Wah, Ahsan Iqbal, and Prem Prakash Jayaraman. Opportunistic computation offloading in mobile edge cloud computing environments. In *Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, volume 1, pages 208–213. IEEE, 2016.

[24] Duc Van Le and Chen-Khong Tham. A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018.

[25] Ke Zhang, Yuming Mao, Supeng Leng, Yejun He, and Yan Zhang. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading. *IEEE Vehicular Technology Magazine*, 12(2):36–44, 2017.

[26] Yang Zhang, Dusit Niyato, and Ping Wang. Offloading in mobile cloudlet systems with intermittent connectivity. *IEEE Transactions on Mobile Computing*, 14(12):2516–2529, 2015.

[27] Yang Zhang, Dusit Niyato, Ping Wang, and Chen-Khong Tham. Dynamic offloading algorithm in intermittently connected mobile cloudlet systems. In *Communications (ICC), 2014 IEEE International Conference on*, pages 4190–4195. IEEE, 2014.

[28] Shichao Zhu, Lin Gui, Jiacheng Chen, Qi Zhang, and Ning Zhang. Cooperative computation offloading for uavs: A joint radio and computing resource allocation approach. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 74–79. IEEE, 2018.