On the Optimality of Task Offloading in Mobile Edge Computing Environments

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

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

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

***Abstract*—Mobile Edge Computing (MEC) has emerged as new computing paradigm to improve the QoS of users' applications. A challenge in MEC is computation (task/data) offloading, whose goal is to enhance the mobile devices' capabilities to face the requirements of new applications. Computation offloading faces the challenges of *where* and *when* to offload data to perform computing (analytics) tasks. In this paper, we tackle this problem by adopting the principles of Optimal Stopping Theory contributing with two time-optimized sequential decision making models. A performance evaluation is provided using real world data sets compared with baseline deterministic and stochastic models. The results show that our approach optimizes such decision in single user and competitive users scenarios.**

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

I. Introduction

Recently, mobile devices and applications with different functionality, such as drones, Vehicular Networks (VN) have emerged. This development has enabled computing and sensing devices to launch applications such as Augmented Reality (AR) [1], Internet of Things (IoT)-based predictive analytics tasks at the edge [2], intelligent vehicle control, traffic management, and interactive applications [3]. Although such mobile devices have computing capabilities to run such applications, they still cannot efficiently handle them. The reason for this limitation is that modern applications require significant processing in relatively short time and consume significant energy. Such limitations have motivated the emergence of the *computation offloading* concept. Computation offloading is the process of sending a computation task and its data to a remote server for delegating this computation [4]. As the emerging applications require intensive computation processes, computation offloading provides a promising solution to overcome the limitation of the mobile node. A study showed the benefit of task offloading for the Percipio AR application [5]: the computation offloading reduces latency up to 88% and energy consumption of mobile devices up to 93%. The Mobile Edge Computing (MEC) paradigm is a promising environment for computation offloading. The rationale architecture of this concept is to offer cloud services *closer* to mobile nodes by placing data-centres/servers at the edge of the network in e.g., Base Stations (BS), indoor places such Wi-Fi and 3G/4G access points [6] or Roadside Units (RSU) [3].

Motivation & Challenge: Computation offloading in MEC environment faces several challenges, with the most significant one being: *the decision of when and where to offload task/data to perform a computing task while the user is on the move*. The decision making of tasks offloading is of high importance as it is expected to directly affect the Quality of Service (QoS) of the application including the inherent latency due to the current MEC server load and the transmission/communication status between mobile nodes and the MEC server. Moreover, the envisioned deployment of MEC servers in the 5G technology brings new challenges. As mobile nodes move between many MEC servers, they should ideally connect to the *best* server and at the *best time* in terms of servers' load. MEC servers' load have large variation, e.g., in some time, there are a large number of users concurrently using the same server, whereas in others only a few users are connecting [7]. The challenge is then to deal with the computation offloading decision given we experience large variations with respect to the QoS of mobile nodes. Once an offloading decision has been made, should the offloading happen right away or it would be better to delay the offloading for later in order to find a better MEC server in terms of load? Can we efficiently predict this postponement in light of finding a *better* MEC server?

In this work, we argue that in MEC environment, the computation offloading decision can be *optimized and managed* by applying the principles of the Optimal Stopping Theory (OST). In this context, while the mobile node is sequentially roaming (connecting) through MEC servers with different loads and different network conditions, the mobile node has to *locally and autonomously* decide which server should be used for offloading the data to perform the computing task best. The remainder of this paper is organized as follows: we summarize 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

Most of the approaches for task offloading at the edge focus on the decision of whether the task should be processed locally or offloaded to Cloud. There are two main objectives: minimization of (1) the execution delay and (2) energy consumption. The ST-CODA [8] method refers to a spatio-temporal computation offloading decision that supports a mobile node to decide where and when to offload tasks

dealing with the transmission costs in Cloud-enabled heterogeneous networks. Our work is different from [8] because our time-optimized decision refers only to task offloading to edge/MEC servers rather than the Cloud. In [8], the method defers the offloading decision until a low cost network is found. Our approach, however, *optimally* defers the offloading decision until a lightly loaded server with low transmission delay is found. By considering the load of MEC servers and their transmission delay, we are more likely to provide higher expected QoS to applications. The work in [9] presents a decision strategy specifically for data mining applications. When the device has collected data from different sources, then it scans and gets a list of available edge servers and picks the best one in terms of available resources. However, since the mobile node is moving, there might be a better server in its path (which is not considered by the communication interface at the moment of scanning) due to movement to different places. Hence, there might be a better MEC server in terms of execution delay. This opportunity is considered in our method. The work in [10] proposes a cooperative method to minimize the energy consumption and task execution latency. The considered method is for Unmanned Aerial Vehicle (UAV) applications that capture photos/videos which are offloaded to an edge server. When the task is generated by the UAV, a centralized 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 server should be allocated. In [10], the decision is made by the orchestrator, while in our method, the decision is autonomously made by the mobile node itself dealing with heterogeneous operators of MEC servers. In [11], the authors proposed a centralized offloading algorithm implemented in CONCERT architecture (SDN for MEC) proposing a short-term prediction of an offloading decision including the amount of data to be offloaded together with the communication path. Their goal is to minimize the energy consumption while satisfying the delay constraint. Our work differs in the offloading decision deployment run on the mobile node given that it is efficient to execute thus avoiding a centralized management system. The method in [12] determines which part of the application should be offloaded and proceeds with an offloading decision based on the current state of the node's resources modelled as a Markov Decision Process using Q-learning for training. The main goal is to minimize the delay of the offloaded applications considering the mobile fog in close proximity, the adjacent mobile fog, or the Cloud as feasible offloading sites. In our work we study the case where the mobile node only offloads to a MEC server out of a set of feasible MEC servers. A machine-learning method for task offloading is proposed in [13], which considers whether a task should be executed locally or not. Our proposed work could support the method in [13] to optimally decide which server to offload and what time the offloading should occur under the objectives of total delay minimization and the probability of selecting the best server, as will be shown later. The work in [14] proposes an offloading decision model for vehicles, which decides which part of the application should be done locally or in Cloud given task requirements. A heuristic mechanism for partitioning and scheduling between the vehicular and Cloud is proposed. This work is specifically designed for cloud-based architectures and focuses on the decision regarding which part of application should be offloaded. In [15], we proposed a time-optimized task offloading decision making in MEC environments that minimizes the total delay when offloading task/data and compared such method to the optimal solution. One limitation of our previous work is that we considered that the number of MEC servers is known to mobile nodes. In this paper, we depart from our previous method considering the realistic case where the mobile nodes do not know the number of servers and provide two models dealing with two different *objectives for optimization in task offloading*.

Contribution: Our method can be adopted in different applications integrated with offloading decision algorithms and is efficient to run on the mobile devices. In contrast to the previous work, we explicitly focus on the *decision* regarding which MEC server (or simply server) and which time should be determined for offloading after a decision has been made by one of the previous methods. Our contribution is threefold:

- Derived by [15] and [3], we propose a model that maximizes the chance of offloading to the optimal server.
- Extend our work [15] by proposing a model for the realistic case where the number of servers is unknown.
- Provide comprehensive comparative evaluation of our models with others in a single & competitive setting.

Our method is suitable for MEC applications such VN [3], UAV [10] or for data mining applications (e.g., activity recognition [9]) as the mobile nodes roam among servers deployed at the edge of the network. To the best of our knowledge, this is the first work to consider the decision offloading strategy as an OST problem with the objective of maximizing the chance of offloading at the best server and minimizing the expected delay given an unknown number of servers.

III. Time-Optimised Offloading Decision Making

A. System Model and Problem Formulation

Although we envision our proposed methods in many MEC applications, in this paper we consider the case of VN applications as studied in [3] and presented in Fig. 1. In such a setting, RSUs are deployed along the road. RSUs are equipped with servers that provide computing resources for vehicles in the road to perform a computing task. The vehicles can generate different types of data to be processed by the server. For example, these data can be for real time services such as image recognition, image processing or analytical task for data generated by embedded sensors within the car [16] or any other tasks as inferential and predictive analytics [17], statistical learning models building and/or models selection, [18]. Furthermore, the task can be generated by the passengers in vehicles using smart phones. The connection to the server can be made through LTE wireless link at a time. For each server, at each time instance, there is a temporal *load*

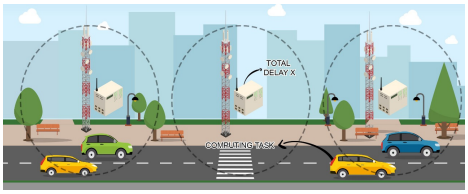


Figure 1: Tasks/data offloading in a VN scenario.

associated with it. Such load refers to the number of user requests the server is processing. There is also a transmission delay from the user to the server, which can be estimated by the expected time for uploading the data to the server and receiving the processed data/analytics results back. The execution delay for a task, hereinafter is referred to as **total delay**, on the server, notated by X, incorporates, as stated in [19]: (1) the transmission duration X_T of the offloaded data to the server; (2) the processing time X_P at the server; (3) the time spent X_R to receive the processed data from the server. We assume the existence of an offloading decision framework implemented in the mobile node from previous work, which provides the entities of a network/edge servers profilers as implemented in [13]. This is adopted to provide information about the current X_T, X_P, X_R delays. Specifically, once a task is generated and needs to be offloaded then, at each time, the mobile node checks the current state of the server load and the network condition. The node may connect to a new server or it might be in the range of the same server. This is based on the speed of the mobile node or/and the density of the servers in the vicinity. In both cases, we focus on optimizing the decision on *when* to offload the tasks/data to an available server. Formally, our objectives are (i) *to maximize the chance of offloading to the optimal server* and (ii) *to minimize the expected total delay* $\mathbb{E}[X]$. To keep the continuity of the connection, we assume that there is a mobility management entity in the server [19] which implements a mobility management algorithm, such as path selection, power control algorithms [19], [20] or predictive model as in [3]. We deal with these two objectives by contributing with two time-optimized decision making models adopting the principles of the Optimal Stopping Theory (OST). The OST is concerned with the problem of choosing the best time to take an action based on sequentially observed random variables in order to minimize an expected cost [21], [22].

B. Maximizing the Probability of Offloading to the Best Server

Our first objective deals with the case that the number $n > 0$ of the available servers, which are candidates for task offloading, is known to a mobile user. The objective is to *maximize the probability of selecting the best server for task offloading*. The mobile node is on-line observing a sequence of candidate servers, which are locally ranked in the node from the best to the worst w.r.t. a performance criterion, i.e., the current total delay X. At each observation, the node should decide whether to choose the current available candidate server or not. In the latter case, in this work, the node cannot recall its decision, i.e., if a candidate server is rejected for selection, it cannot be recalled. The challenge is that the node desires to define an offloading policy/rule which *maximizes the chance of choosing the best server w.r.t. the ranking seen so far*. Every server is relatively ranked based on the previous observed servers and can only be checked sequentially and in a random order. Once a server is relatively ranked and rejected, this choice cannot be re-called. The node should maximize the probability to select the candidate among the n candidates, which is *globally* ranked best. This is cast as a Best-Choice Problem (BCP) [21]. In our BCP, we seek the offloading rule that maximizes the probability P_n^* of selecting the best of all n servers and the corresponding probability of that success. Let us call the t-th server *candidate*, if it is relatively best in terms of X_t, $t = 1, \ldots, n$. We then define a positive integer $r_n \in \{1, \ldots, n\}$, defined as

$$r_n \quad = \quad \min\{r \geq 1 : \frac{1}{r} + \frac{1}{r+1} + \cdots + \frac{1}{n-1} \leq 1\}, \quad (1)$$

for $n \geq 2$. Based on the BCP, the optimal policy is to reject the first $r_n - 1$ servers and then select the first candidate, if any, to offload the tasks.

Theorem 1. *The optimal probability in selecting the best candidate in the BCP in (1) is given by:*

$$P_n^*(r_n) \quad = \quad \frac{r_n - 1}{n} \sum_{k=r_n}^{n} \frac{1}{k-1} \quad (2)$$

Proof. Proof is omitted due to space limitations. □

In the case where there is a relatively high number of servers, the optimal probability $\lim_{n\to\infty} P_n^* = \lim_{n\to\infty} \frac{r_n}{n} = e^{-1} \approx 0.368$[1]. Based on this approximation and Theorem 1, we provide the optimal policy for maximizing the probability of finding the best server out of n available servers.

Proposition 1 (BCP-based Optimal Task Offloading Policy)**.** *The node observes the first n/e servers and ranks them immediately w.r.t. their total delay provided by each of them upon request. Then, the node offloads their task/data to the first t-th server with $t > \lceil n/e \rceil$ which is ranked as the relatively best server compared to the previously ones.*

Based on this optimal offloading policy, the node is guaranteed to maximize the probability of offloading the task/data to the best server. If no offloading decision is made after observing the n servers, the node offloads the task/data to the n-th server, since no recall is allowed.

C. Minimizing the Expected Total Delay of Task Offloading

Task offloading to the optimal server, i.e. to the server with the minimum expected total delay is not trivial. By applying a satisfactory strategy to minimize the total expected delay is actually beneficial in the context of applications. Our challenge is then to find an optimal policy with the objective being the minimization of the total expected delay. This is different with the previous policy (maximizing the probability of finding the

[1]When $n \to \infty$, we obtain the well-known Secretary Problem [22].

best), since now we care for minimization of the sequentially observed delay values. In our previous work [15] we dealt with this delay minimization with the assumption that the number of the servers n is known to the nodes. We now drop this assumption and contribute with an optimal policy for delay minimization where the number of server is unavailable to the nodes including a cost for delaying, thus, being applied in realistic scenarios. Let X_t be the random variable denoting the total delay the node is observing for the t-th server at time t. We desire to find when to offload to which server that minimizes the total expected delay $\mathbb{E}[X]$. However, the node *pays* c cost units per observation when it has not yet offloaded the task/data. The cost can be application specific e.g., the rate the gathered data turn obsolete before being processed, a degree of urgency for task computation, the cost for requiring access to a server to ask for its current load. We then define the cost function Y_t at time t *including* the cost for observing servers up to t and the load X_t of the t-th server as:

$$Y_t \quad = \quad X_t + ct. \tag{3}$$

The target is to find the optimal offloading time $t^* = \arg\min_{t\geq 1} \mathbb{E}[Y_t]$ to decide to offload task to the t^*-th server such that up to t^* the expected cost $\mathbb{E}[Y_{t^*}]$ in (3) is minimized.

Theorem 2 (Cost-based Optimal Task Offloading Policy)**.** *The node minimizes the expected cost in (3) by offloading at the first t-th server such that:*

$$t^* \quad = \quad \min\{t > 0 : X_t \leq V^*\} \tag{4}$$

where the V^* is the solution of:

$$\int_{V^*}^{\infty} (x - V^*) dF(x) = c. \tag{5}$$

where $F(x) = \int xp(x)dx$ is the Cumulative Density Function (CDF) of the load X.

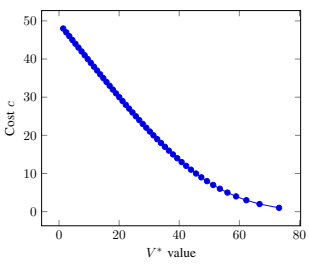
Proof. Proof is omitted due to space limitations. □

Theorem 2 states that, the node offloads the task to the first server whose $X_t \leq V^*$, where the V^* value is the solution of (5). For instance, given a uniformly distributed load $X \in [0, 1]$, i.e., $F(x) = x$ and thus $dF(x) = dx$, we obtain for $V^* \in [0, 1]$:$\int_{V^*}^{1}(x - V^*)dF(x) = (1 - V^{*2})/2$ while for $V^* < 0$: $\int_{0}^{1}(x - V^*)dF(x) = 1/2 - V^*$. Hence, based on Theorem 2, $V^* = 1 - (2c)^{(1/2)}$ for $c \leq 1/2$, and $V^* = -c + (1/2)$ for $c > 1/2$. Fig. 2 (left) shows the V^* optimal threshold vs the cost associated for a total delay following normal distribution $F(x) = 0.5(1 + \text{erf}(\frac{x-\mu}{\sqrt{2}\sigma}))$ with a mean $\mu = 50$ and standard deviation $\sigma = 20$; erf() is the error function. A lower cost c indicates accepting higher total delay, whereas higher cost means a high demand for a small total delay. Also, when we choose a low cost, then the offloading will happen very early at the beginning of the offloading process.

In real-life scenarios, the node should incrementally approximate the CDF of the observed load X based on past load values. This can be achieved by on-line Kernel Density Estimation (KDE) using Gaussian kernel function $K(u) = \frac{1}{\sqrt{2\pi}} e^{-0.5u^2}$ of width $h > 0$, which is widely adopted in an on-line mode, i.e., $\hat{F}_t(x) = \frac{1}{th}\int_0^{x_{\max}} \sum_{k=1}^{t} K(\frac{x-x_k}{h})xdx$, where $x_{\max}$ is the maximum observed load value. A rule-of-thumb[2] width estimator is $h \approx 1.06\sigma t^{-1/5}$. The CDF $\hat{F}_t(x)$ is approximated at the t-th time where the node has observed up to t load values $\{x_1, \ldots, x_t\}$ since the last task offload. Hence, the CDF can be efficiently incrementally updated as:

$$\hat{F}_t(x) \quad = \quad \frac{t-1}{t}\hat{F}_{t-1}(x) + \frac{1}{th}\int_0^{x_{\max}} K(x, x_t)xdx. \tag{6}$$

After calculating V^* for a given cost c, the node starts off the load observation per server, one at a time: If it is less than V^*, the node offloads the task; otherwise, it continues checking for a better load before a pre-defined deadline. When the node has not yet offloaded its task after the deadline, the node has to offload to the last observed server.



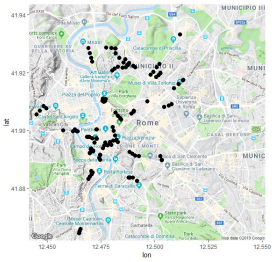


Figure 2: (Left) The V^* value for $\mu = 50$ and $\sigma = 20$ for load X vs. cost c; (right) taxis trajectories in Rome.

IV. Performance Evaluation

Data Set: We used the real dataset of taxi cabs' movements in Rome [23]. The dataset contains GPS coordinates of 320 taxis collected over 30 days. For each row we have the cab-id, date/time and GPS coordinates of the cab's location. For the servers, we used the *k-means* clustering algorithm [24] to divide the GPS coordinates to a set of cells (servers). For the total delay, for each movement, we assumed that the mobile node is observing (connecting to) a server to get the total delay. Thus, for each row, we added a total delay that follows normal distribution with mean $\mu = 50$ and standard deviation $\sigma = 20$. We considered one day trace for 50 cars and focused on one minute movements (date: 2014-02-01 from 00:00:00.73 until 00:00:58.56). Each taxi makes on average 4-5 movements in this time interval. Table I shows a sample of the data we used to evaluate our proposed models, and Fig 2 (right) shows the real movements of the 50 cars in Rome's map in the specified time interval. For example, in Table I, in the first row, car 156 was in position (41.88,12.48); at time (00:00:00.73), the mobile node observes the total delay of the server 4, which has a total delay 80.61 ms. Note that, in Table I, car 156 stays at the same location (within the range of cell 4), but there is variation in terms of total delay. Thus, with respect to the first objective, when we have the same server during the user movements, we try to optimize the decision of selecting the

[2]Silverman, B.W. (1986). Density Estimation for Statistics and Data Analysis. London: Chapman & Hall/CRC. p. 45.

Table I: Data set used in the experiment

car id	Date	lat	long	Delay	Cell
156	”2014-02-0100:00:00.73”	41.88	12.48	80.61	4
156	”2014-02-0100:00:16.47”	41.88	12.48	62.97	4
156	”2014-02-0100:00:30.70”	41.88	12.48	4.53	4
156	”2014-02-0100:00:45.30”	41.88	12.49	4.37	4
187	”2014-02-0100:00:01.14”	41.92	12.46	70.17	1
187	”2014-02-0100:00:16.15”	41.92	12.46	66.59	1
187	”2014-02-0100:00:30.81”	41.92	12.47	31.65	4

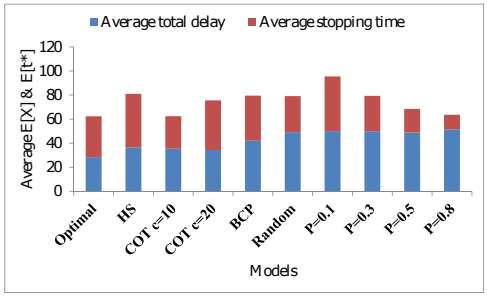


Figure 3: Average total delay $\mathbb{E}[X]$ and average stopping time $\mathbb{E}[t^*]$ in a single user setting.

best time to offload, i.e., in terms of maximizing the chance of selecting the best time to start offloading. When we have high density deployment of servers, our algorithm can be also applied to maximize the probability of selecting the best server. This is applied in the second objective as well.

Performance Assessment in Single user scenario: We compare our OST-based offloading models namely BCP (Section III.B) and COT (Section III.C), respectively, with the HS model in [15], 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 p-model, for each 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. 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. 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.

In Fig. 3, COT ($c = 20$) is the closest to the Optimal with average total delay $\mathbb{E}[X]$ being 34 ms, while BCP and HS perform better than the Random and p-models. Fig. 3 also shows the average stopping time $\mathbb{E}[t^*]$ for all models. The stopping time refers to the time at which the mobile node decides to offload. The Optimal model, on average, stopped at 33s; the p-model ($p = 0.8$) stopped at the beginning of each decision period while the p-model ($p = 0.1$) has the highest stopping time as it delays the offloading decision until the end

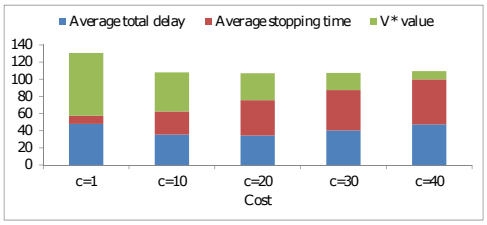


Figure 4: Average total delay $\mathbb{E}[X]$, average stopping time $\mathbb{E}[t^*]$, and optimal decision thresholds V^* in the COT model with different costs c.

of the decision interval.

To examine the COT behaviour with different values of c and V^*, in Fig. 4, we show the average total delay, average stopping time and V^* values. As mentioned that low cost means accepting higher total delay and higher cost means looking for less total delay. We can see that that V^* decreases as the cost increases. When $c = 1$, the model accepts larger total delay as the value of V^* is very high, i.e, 73. As result, the model offloads very early and thus less stopping time. When $c = 40$, the V^* is very small, and thus, looking for very small total delay. In such case, the model delays its decision to find a total delay less than 9, but since, we obtain a few total delays less than 9 (based on the distribution used), the model goes with the last server. We achieve an improvement in terms of total delay when $c \in \{10, 20\}$ with an ideal average stopping time, close to the Optimal as shown in Fig. 5 (in ascending order). That is, our models fall in the area around the *actual* optimal stopping time dictated by the Optimal.

Performance Assessment in Competitive Setting: We consider the case where many users have similar start times thus similar expected stopping times when applying our models. The users are expected to connect to the same server with high probability. We used the `simmer` [25] discrete simulator in R environment used recently in [26] to simulate 5G scenarios. For each user, there is a movement trajectory (extracted from the dataset) which includes the start time, the stopping time and the total delay X. We assume that the server handles up to 5 requests at a time. We evaluated all models in terms of the average Waiting Time Ratio (WTR) defined as: $WTR = \frac{(t_E - t_S) - t_A}{t_E - t_S} * 100$, where t_E refers to the time at which the server finished the computing task and sends the results back to the mobile node, t_S is the time at which the task to be offloaded occurs. In our experiment, this time, t_S, is the time where the user starts moving (starting time), and t_A is the activity time which includes searching time and the total delay X. The lower the WTR is, the less time the users need to wait for their tasks to be processed. For example, if the WTR is 50%, that implies that 50% of the total time is waiting time in the server queue. Fig. 6 shows that the HS and COT ($c = 20$) are almost the same with WRT 50% and are both the closest to Optimal; when $c = 10$ in COT, HS outperforms. Fig. 5 shows that the average task offloading time (average stopping time) varies, but, in general when we obtain smaller stopping times, like in p-model ($p = 0.8$) in Fig. 6 and COT ($c = 1$) in Fig. 6, we achieve higher WTR. The p-model ($p = 0.8$) is more likely to offload at the beginning of each interval;

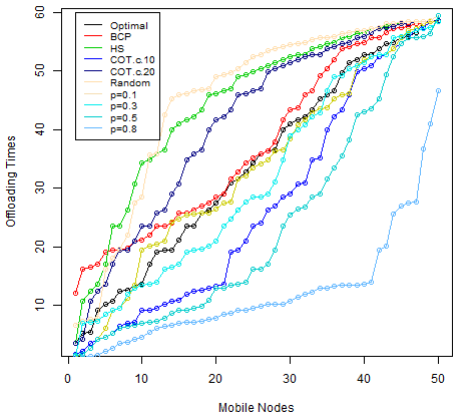


Figure 5: The task offloading times for all methods.

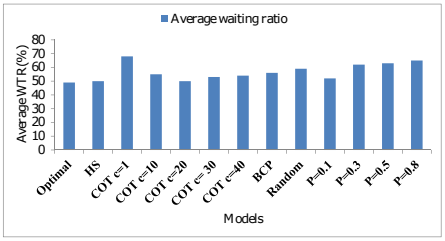


Figure 6: Average WTR for all models in a competitive setting.

while p-model $(p = 0.1)$ achieves a good WTR. The same holds true in our models. Among all OST-based models, we obtain higher stopping times, thus, less WTR. This supports **our initial hypothesis:** *it is not beneficial to offload at the very first server; the mobile node should, at least, pass a couple of servers to obtain a lower total delay and lower WTR when competing with other nodes*. **Note:** our models have linear computational time thus efficiently run in mobile nodes.

V. Conclusions

We propose two Optimal Stopping Theory-based offloading sequential decision strategies for mobile users in MEC. Mobile nodes sequentially determine *when* and *which* server to offload their task/data to considering the total delay incurred at each server. Our models outperform other baseline solutions in terms of the total delay and average waiting ratio. We aim to study the impact of the timeliness of data and applications' characteristics on our OST models and develop new ones dealing with these constraints.

VI. Acknowledgement

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 CA15127:RECODIS Resilient communication and services, and by EU-H2020 GNFUV (#Grant 645220). 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.

References

[1] T. Braud, F. H. Bijarbooneh, D. Chatzopoulos, and P. Hui, "Future networking challenges: The case of mobile augmented reality," in *37th ICDCS*. IEEE, 2017, pp. 1796–1807.

[2] N. Harth and C. Anagnostopoulos, "Edge-centric efficient regression analytics," in *EDGE*. IEEE, 2018.

[3] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[4] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Applied computing and informatics*, 2016.

[5] J. Dolezal, Z. Becvar, and T. Zeman, "Performance evaluation of computation offloading from mobile device to the edge of mobile network," in *CSCN*. IEEE, 2016, pp. 1–7.

[6] M. 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.

[7] C. N. Le Tan, C. Klein, and E. Elmroth, "Location-aware load prediction in edge data centers," in *2nd FMEC*. IEEE, 2017, pp. 25–31.

[8] H. Ko, J. Lee, and S. Pack, "Spatial and temporal computation offloading decision algorithm in edge cloud-enabled heterogeneous networks," *IEEE Access*, vol. 6, pp. 18 920–18 932, 2018.

[9] M. H. ur Rehman, C. Sun, T. Y. Wah, A. Iqbal, and P. P. Jayaraman, "Opportunistic computation offloading in mobile edge cloud computing environments," in *17th MDM*, vol. 1. IEEE, 2016, pp. 208–213.

[10] S. Zhu, L. Gui, J. Chen, Q. Zhang, and N. Zhang, "Cooperative computation offloading for uavs: A joint radio and computing resource allocation approach," in *EDGE*. IEEE, 2018, pp. 74–79.

[11] F. Yu, H. Chen, and J. Xu, "Dmpo: Dynamic mobility-aware partial offloading in mobile edge computing," *J FGCS*, vol. 89, pp. 722–735, 2018.

[12] M. G. R. Alam, M. M. Hassan, M. Z. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for iot applications," *J FGCS*, vol. 90, pp. 149–157, 2019.

[13] W. Junior, E. Oliveira, A. Santos, and K. Dias, "A context-sensitive offloading system using machine-learning classification algorithms for mobile cloud environment," *J FGCS*, vol. 90, pp. 503–520, 2019.

[14] A. Ashok, P. Steenkiste, and F. Bai, "Vehicular cloud computing through dynamic computation offloading," *Com. Com.*, vol. 120, pp. 125–137, 2018.

[15] I. A. I. Alghamdi, C. Anagnostopoulos, and D. Pezaros, "Time-optimized task offloading decision making in mobile edge computing," in *11th IEEE Wireless Days*, 2019.

[16] J. Wang, J. Cho, S. Lee, and T. Ma, "Real time services for future cloud computing enabled vehicle networks," in *WCSP*. IEEE, 2011, pp. 1–5.

[17] N. Harth, C. Anagnostopoulos, and D. Pezaros, "Predictive intelligence to the edge: impact on edge analytics," *Evol. Sys.*, pp. 9(2):95–118, 2018.

[18] C. Anagnostopoulos and K. Kolomvatsos, "Predictive intelligence to the edge through approximate collaborative context reasoning," *Appl. Intell.*, pp. 48(4):996–991, 2018.

[19] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *arXiv:1702.05309*, 2017.

[20] J. Plachy, Z. Becvar, and P. Mach, "Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network," *Com. Net.*, vol. 108, pp. 357–370, 2016.

[21] S. A. Peskir, G., "Optimal stopping and free-boundary problems, brickhauser," 2006.

[22] T. S. Ferguson, "Optimal Stopping and Applications," http://www.math.ucla.edu/ tom/Stopping/Contents.html, March 2019.

[23] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, "CRAWDAD dataset roma/taxi (v. 2014-07-17)," Downloaded from https://crawdad.org/roma/taxi/20140717, Jul. 2014.

[24] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer Inc., 2001.

[25] I. Ucar, B. Smeets, and A. Azcorra, "simmer: Discrete-event simulation for r," *arXiv:1705.09746*, 2017.

[26] I. Ucar, J. A. Hernández, P. Serrano, and A. Azcorra, "Design and analysis of 5g scenarios with simmer: An r package for fast des prototyping," *IEEE Comm. Mag.*, no. 99, pp. 2–8, 2018.