Optimized Contextual Data Offloading in Mobile Edge Computing

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) is a new computing paradigm that moves computing resources closer to the user at the edge of the network. The aim is to have low-latency, high bandwidth, and to improve energy consumption when running computational tasks. The idea of deploying MEC servers near to the users along the 5G technology has led to open an interest in the field of Vehicular Network (VN). MEC servers can play significant roles in improving the performance of VN applications. In this environment, offloading computational tasks over collected contextual data by the mobile nodes (Autonomous Vehicles (AV)) meets the challenge of when & where to offload the collected data while on the move. In this work, we modeled the problem of offloading contextual data to the MEC servers as an optimal stopping problem. Our objectives are to offload to a MEC server with lower execution time and before the collected data get stale. We evaluated our model using real mobility trace with real servers' utilization; the results showed that the proposed model outperforms other offloading methods.**

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

I. Introduction

Mobile Edge Computing (MEC) refers to a computing paradigm that moves computing resources closer to the user at the edge of the network. It intends to relocate the Cloud computing resources in the radio access network to optimize the delivery of content and applications to end-users [1]. It involves deploying small data centers (servers) at the edge of the network in locations such as base stations, access points, or within a Roadside Unit (RSU). Further, MEC servers can be an intermediate data-processing layer for data offloaded by mobile nodes [2]. MEC servers can play significant roles in improving the performance of Autonomous Vehicles (AV) and Unmanned Aerial Vehicles (UAV) applications. For example, despite AVs typically including on-board units, they have small-scale computing and storage resources because of which they are dependent on other computational resources [3]. Further, AVs are equipped with a massive number of sensors that collect contextual data for different types of applications such as transportation systems and navigation applications [4]. An autonomous driving vehicle, for example, produces and consumes approximately 40 terabytes of data per eight driving hours (e.g., a city's High Definition (HD) map is approximately 1.5TB) [5]. It should be noted that using too far data centers (Cloud) for data offloading is ineffective and has

978-3-903176-32-4 © 2021 IFIP

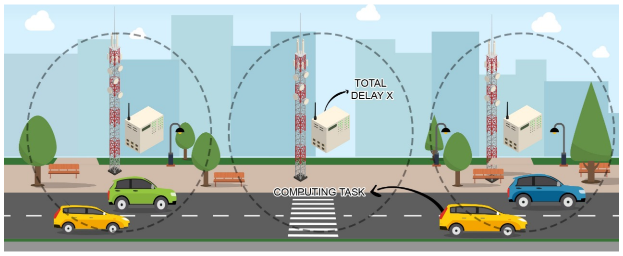

Figure 1: MEC environment.

a cost concerning the network resources such as bandwidth and the delay it may introduce.

Consider **the architecture** where MEC servers deployed within the RSU as proposed in [4] and [6] and visualized in Fig. 1. These servers provide computing resources for mobile nodes. The mobile node in this work mainly refers to autonomous vehicles but it also can refer to smartphones used by the passenger in the AV passing by in the road. The AVs collect data from sensing devices, and offload the data to one of the edge servers for further processing. **The key problem** now is the data offloading decision by which the vehicles select an edge server to offload the data. The selection of where (MEC server) and when (time) to offload has a large impact on satisfying the requirements of the mobile node and the running applications [3]. In addition, data gathered by the AVs tend to have strict timeliness requirements which may result in the data becoming out-of-date.

A naive centralized **offloading method** can be considered within such an environment in which the vehicle can request, from a centralized server, the information about the possible MEC servers that can be used for offloading along the road [6], [7]. The centralized server can be located in a higher layer and it is connected to all MEC servers with wired network connectivity [6], [7]. However, such architecture is not visible as this might introduce load on the network as the number of vehicles on the road increases [6]. Another solution would be using Vehicle to Vehicle communication (V2V) as discussed in [4], where a vehicle can send tasks to the MEC server (using V2V technology) that it will pass by at the time the computational task finishes. Such a solution requires the presence of other vehicles, which is not guaranteed all the time. Without having a centralized server nor global information about the potential MEC servers [6] [8], the mobile node is expected to encounter the situation where it only knows about the server in the range of that mobile node and it does not know about the next MEC servers and their resources. As a result, the mobile node has to make **an independent offloading decision** while in the move. Thus, the challenge now is how to optimize the decision of selecting the MEC server if the mobile node only knows about the MEC server it is observing. In other words, the mobile node does not have global information (or the mobile node has incomplete information) about the candidate MEC servers to be used for offloading. In this case, we consider **two important factors** that can be used in order to delay the offloading in the light of finding a better MEC server. First, we can consider mobility as an advantage to optimize the decision of which MEC servers to offload. As the speed of the vehicle increases, the probability of having a better MEC server with low workload increases [9]. Second, there is usually a certain deadline for the computational task which gives an opportunity for the decision maker to delay and explore more options for offloading [6], [10].

As an example, consider **the use case** where MEC servers deployed in RSUs can act as an Edge-Assisted HD map update as discussed in [11]. The MEC servers, in this case, can be used to build or update an HD map based on collected data by vehicles exploiting the Vehicle-to-Infrastructure (V2I) communication method [5]. In this situation, it is necessary that the data are offloaded in the best time in terms of the processing time and that the data are offloaded prior to becoming out-of-date [11]. The previous requirements can be also applied to other types of contextual data with aim of maximizing the *quality of data analytics*. Therefore, we need an intelligent algorithm that selects an optimal or near-optimal server based on the requirements of the mobile nodes' applications.

In this work, we argue that in MEC environments, with a set of edge servers and a set of mobile nodes (AVs), data-orientated task offloading decisions can be optimized by applying the principles of **Optimal Stopping Theory (OST)**. The principle of the OST involves selecting a time to pick a given action based on sequentially observed random variable in order to maximize an expected reward or to minimize an expected cost [12]. Our objective is to provide an efficient, a lightweight, independent and contextual data offloading mechanism to meet the requirements of the data offloading in MEC environments.

The paper is organized as follows: we summarize related work and present our contribution in Section II, while details of our OST-based system are described in Section III. Evaluation results are provided in Section IV, and a discussion about the proposed model is provided in Section V. Section VI concludes the paper and outlines future research directions.

II. Related Work & Contribution

Offloading decisions in general in MEC environments have been widely studied in the literature with the objectives of minimizing execution time and power consumption. **In this work, we focus on the decision of when and where to offload in edge computing environments**. In this regard, studies have focused on specific applications as in [13], [14] and [15] or created general models under the assumption that the models can be applied to different applications (tasks) as in [16]. Different from these works, we investigate a scenario where a mobile node lacks information about potential available MEC servers and needs to calculate the best time for data offloading considering the timeliness of the data in a sequential manner. The authors in [6] proposed a decentralized management scheme for mobile edge servers and an offloading approach in the edge computing environment. The proposed idea is based on the Peer to Peer (P2P) networking architecture where peers (MEC servers and moving vehicles) have equal privileges. Different from this study, in our work, we are trying to make the mobile node more independent with respect to the offloading decision-making. The study in [6] requires the mobile node (vehicle) to be involved in a P2P network with the MEC servers which might not be available for the mobile node in all environments.

This work considers the use case where MEC servers can form a platform for gathering, storing and processing a huge amount of data collected and offloaded by mobile nodes, e.g. AVs [2]. In this regard, BEGIN was proposed in [17] to utilize the big data collected from the edge environment to provide energy-efficient edge computing. RedEdge [18] is a big data processing architecture that incorporates a mechanism that facilitates the processing of big data streams at the edge near to the user. The idea of having the edge servers as a platform for data analytics was also proposed in [19]. The proposed MEC architecture is for a mobile crowd-sensing (MCS) service. The MCS refers to a human-driven IoT service to which people send their observations of different phenomena in their surroundings by sharing their sensor data while on the move. The work in [20] proposed a decentralized traffic management system to minimize the average response time caused by traditional centralized traffic management. The scenario in this work is that vehicles can upload sensed events (e.g., traffic jams, car accidents, and road surface damages) to a nearby Road Side Unit (RSU). Our work can be a supportive method for such systems by optimizing the way by which data are collected and offloaded to an edge server.

In our previous works [21], [22] and [23], we proposed a set of OST-based models with the objectives of maximizing the probability of offloading to the best server or at the best time and minimizing the total delay when offloading a task. Our previous studies are general frameworks for task offloading in MEC environments. The OST was also utilized in [8] for the objective of deriving a good balance between the gain of choosing the best edge device and the accumulated cost of deep resource probing. The authors try to enhance the ability of the proposed OST-based model by utilizing a layered learning mechanism to define the OST thresholds and the sequence of the edge nodes used for offloading. However, such enhancement can be an overhead and battery consumption for the mobile nodes as it implements deep neural network and

Deep Q-Networks. Also, in their applications, the assumption is that the mobile node will have a list of edge devices once a task is generated and then the mobile node will define which edge node makes a good balance between the cost of probing and the execution delay of the task. We depart from the previous work and try to focus on offloading collected data from the mobile nodes. **The contributions of this work are:**

- Derived by the use case of AVs in MEC environments as suggested by [4] and [5], we enhance our work in [21]–[23] and study the case that arises when a mobile node (AVs) wants to offload contextual data to a MEC server while it is moving. We treat the decision-making as an optimal stopping time problem.
- Performance evaluation of our quality-aware optimal data offloading model and a comparison with baseline solutions and the methods [21]–[23] in the literature.

III. System Model & Problem Formulation

A. System Model:

We consider a setting where there exists a set of MEC servers deployed along the mobile node's path on the move as shown in Fig. 1. Such a setting can be seen in the VN applications as studied in [4] and [6], where smart vehicles, e.g. AVs, perform different types of tasks. This can be also applied to other mobile nodes such as passengers in cars [11] or unmanned aerial vehicles [2]. A mobile node can offload contextually collected data to perform data analytics tasks on one of the deployed MEC servers. As the MEC servers are operated within the radio access networks with the help of RSUs, their services coverage are limited [4]. As a result, as the AVs move, they will pass by a set of MEC servers once a time during the data offloading session. In this work, we assume that the AV can have (check or connect to) only one MEC server at a time and does not know about the MEC servers in the road ahead [4]. The AVs or mobile nodes in the car can access the RSUs using Vehicle-to-Infrastructure (V2I) communication mode. We elaborate on the existence of an offloading decision framework implemented in the mobile node from previous work, which provides the entity of a network/edge servers profilers as studied in [14]. Network profiler is utilized to provide information about the current load (or delay) of MEC servers. The data analytics task can be data correlation analysis, inferential and predictive analytics [24], statistical learning models building, model selection [25], [26], a mobile crowd-sensing (MCS) [19] or data for HD maps as in [11]. In this setting, we consider the following optimization problem.

B. Problem Formulation:

A mobile node (AVs for example) collects contextual data from its surrounding environment and desires to offload them to the *best* MEC server in terms of computational load, expected execution delay or the time it takes the MEC server to broadcast the results to other systems to perform analytics task *before* the data turns obsolete. Trying to offload the most up to date data is usually described in the context of data analytics as the *timeliness* or the *freshness* of contextual data [27]. The data can be environmental data such as critical, i.e. safety or traffic data, weather condition, traffic information or information for HD maps to be used by the MEC server to update map information in a specific area [11]. Let X_k be the random variable indicating the time the MEC server k needs to perform an analytical task. As mentioned earlier, X_k can also indicate different random variables, e.g., the transmission time, the computational workload of a server or the broadcasting time for the results to other systems. Let n be the maximum number of MEC servers (or simply the number of observations) that can be observed before the collected data turns obsolete.

We then define a data offloading cost function Y_k including the delay X_k for $k-$th MEC accessible server and the staleness of the data c when observing server k as:

$$Y_k \quad = \quad X_k + c \cdot k, \tag{1}$$

with cost rate $c = 1/n, n > 0$, i.e. the collected data, at each observation, get old by $\frac{1}{n} \cdot k$. For example, if $n = 5$, at observation 3 ($k = 3$), the timeliness of the data is $\frac{1}{5} \cdot 3 = \frac{3}{5} = 0.6$. We then formulate the problem of data offloading as follows:

Problem 1. *Given a staleness data within n observations, the mobile node after collecting (fresh) data tries to find the best MEC server $k \in [1, n]$ to offload the data such the expected data offloading function in (1) is minimized. At that time (optimal data offloading time) the following infinum is attained:*

$$\inf_{\tau \in [1,n]} \mathbb{E}[Y_\tau]. \tag{2}$$

In order to proceed with an optimal solution for Problem 1, we introduce two states of our system represented by z_k:

- The system state $z_k = z_\top$ at $k \leq n$ indicates that the mobile node has offloaded the data to a MEC server with delay X_k and staleness degree $c \cdot k$, and where $z_\top$ is the terminating state.
- The system state $z_k = y_{k-1}$ at $k \leq n$ indicates that the mobile node has not yet offloaded the data. In such case, the state z_k refers to the MEC server delay and staleness value when observing MEC server $k-1$, i.e., $y_{k-1} = x_{k-1} + c(k-1)$.

Hence, based on the decision of the mobile node to either offload the data or not at the observation k in light of minimizing the expected cost in Problem 1, the system state is then:

$$z_{k+1} = \begin{cases} z_\top, & \text{if } z_k = z_\top \text{ (stop and offload data)}, \\ z_k, & \text{(do not offload data and continue)}. \end{cases} \tag{3}$$

At each observation k, the staleness of the data increases by $\frac{1}{n}$. Let $J_k(z_k)$ be the optimal time to offload the data. The Bellman's equation for our system, based on principle of optimality, is:

$$J_n(z_n) = z_n \tag{4}$$

for $k = n$, , i.e., at the end of the staleness period, and

$$J_k(z_k) = \min\left(z_k, \mathbb{E}[J_{k+1}(z_{k+1})]\right), \tag{5}$$

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

Equation 4 refers to the situation where the mobile node arrives at the last observation n. At that stage, we must offload to the last observation. Equation 5, on the other hand, refers to the situation where the mobile node is at observation k. In that case, the mobile node chooses to offload to the minimum between the current observation $z_k = y_{k-1}$ and the expected $z_{k+1} = y_k$. In particular, the expectation $\mathbb{E}[J_{k+1}(z_{k+1})]$ denotes the expected data offloading cost of the mobile node offloads the data at time $k+1$. Hence, for the mobile node:

- It is optimal to stop and offload the data to the MEC server k such that $z_k \leq \mathbb{E}[J_{k+1}(z_{k+1})]$
- Or, it is then optimal to continue to the next state.

If we notate $a_k = \mathbb{E}[J_{k+1}(z_{k+1})] - ck$ then we obtain:

$$J_k(z_k) = \min(z_k, a_k + ck) = \min(x_k, a_k) + ck. \tag{6}$$

This indicates that the optimal offloading time, when the mobile node evaluates the value x_k, is achieved according to the following optimal data offloading rule:

- Offloading data to the server with x_k, if $x_k < a_k$
- Continue and do not offload, if $x_k > a_k$
- Both actions (offload or continue) are optimal if $x_k = a_k$

Based on this formulation, we then seek the optimal decision scalar values $\{a_k\}_{k=1}^n$ to be known to the mobile node before starting off any data offloading decision at any time within the number of observations n. We obtain these scalar values as stated in the following theorem.

Theorem 1. *The scalar decision values $a_1, a_2, \ldots, a_n$ are calculated through backward induction based on the recursion:*

$$a_k = a_{k+1}(1 - F_X(a_{k+1})) + \int_0^{a_{k+1}} x dF_X(x) + c, \tag{7}$$

for $k = 1, \ldots, n-1$, with initial condition

$$a_n = \mathbb{E}[X] + c, \tag{8}$$

where $F_X(x) = P(X \leq x)$ is the cumulative distribution function of the delay X.

Proof. We have that $J_k(z_k) = \min(x_k, a_k) + ck$ based on the definition of $a_k = \mathbb{E}[J_{k+1}(z_{k+1})] - ck$. We then obtain that $J_{k+1}(z_{k+1}) = \min(x_{k+1}, a_{k+1}) + ck + c$. Hence, by taking the expectation and taking away the factor ck from both sides, we obtain that: $a_k = \mathbb{E}[J_{k+1}(z_{k+1})] - ck = \mathbb{E}(\min(x_{k+1}, a_{k+1})) + c$, which is the recursion: $a_k = \mathbb{E}_x(\min(x, a_{k+1})) + c$ and obtain

$$\begin{aligned} a_k &= \int_0^{a_{k+1}} x dF_x(x) + \int_{a_{k+1}}^{\max(X)} a_{k+1} dF_X(x) + c \\ &= \int_0^{a_{k+1}} x dF_x(x) + a_{k+1}(1 - F_X(a_{k+1})) + c, \end{aligned}$$

$\max(X)$ is the maximum X value and $a_n = \mathbb{E}[X] + c$. □

Remark. *It should be noted that Theorem 1 can also be applied to a situation where the mobile node is moving within the range of one MEC server and tries to choose a time with minimized delay. In such case, the horizon n can be divided into time slices, and then we calculate decision values $\{a\}$ using (7) and (8) based on the probability distribution of the X of the MEC server over time.*

As an example, consider Fig. 2 where the collected contextual data have to be offloaded to a MEC server before the data get obsolete for stale periods $n \in \{10, 20, 30\}$. We can observe that when we are at an earlier time $k = 1$, we look for a lower delay. As the data turn more obsolete with time, we become more tolerant to accept higher delay, thus the decision values are getting higher reflecting this tolerance. Moreover, to have a clear idea, in Fig. 3, we show the values of $\{a_k\}_{k=1}^{50}$ with the load (or delay) X_k following normal distribution for $\mu = 50$ and $\sigma = 10$ when $n = 50$; it is optimal to offload when $k = 27$ as it is the first time the condition $x_{27} < a_{27}$ holds true. In particular, in our model, we firstly specify the maximum number of MEC servers n that can be observed during the timeline of the collected data which directly defines how many decision time instances the mobile node can wait until the data become obsolete. We also need to get the scalar values a_k by solving (7) and (8) based on the probability distribution of the delay X, e.g., server resource utilization if we are looking for minimized server load. After we obtain the values of a_k, we check the value of x_k. If $x_k \leq a_k$, the mobile node selects the MEC server k for data offloading, otherwise, it continues to the next available MEC server. At $k = n$, if the mobile node has not yet offloaded the data, it offloads the data to the first available MEC server (n-th MEC server).

IV. Performance Evaluation & Comparative Assessment

A. Data Set

To simulate the movements of the mobile nodes, we used the real data set of taxi cabs' movements in Rome [28]. The data set contains the 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. It is worthwhile to mention that the use of mobility trace here is not for studying the mobility of users. It is used in our experiment to use each time movement as location or time to check for a server/time to offload. In other words, each movement is modeled as an observation or connection to a MEC server.

The random variable X is represented in this experiment by real servers' utilization (the CPU utilization) data set obtained from [29]. In the servers' data set, we have around 150 servers' data (more than 1 billion rows). Thus, for each movement in the mobility trace, the car picks a server from the servers' data set, checks the server utilization and takes a decision of whether the car should offload at that time or continue observing based on the decision suggested by the proposed model being compared with different selection

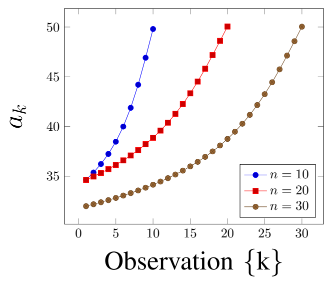

Figure 2: The optimal scalar values $\{a_k\}_{k=1}^n$ for data staleness horizon $n \in \{10, 20, 30\}$, delay X is following normal distribution for $\mu = 50$ and $\sigma = 10$.

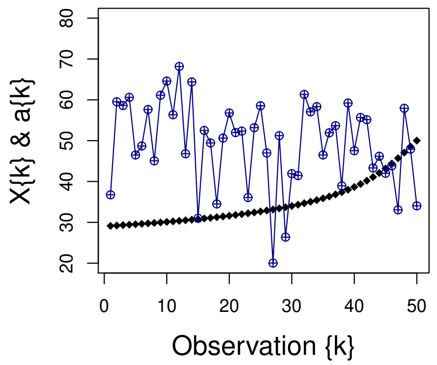

Figure 3: The decision values $\{a_k\}_{k=1}^{50}$ (black points) and simulated server delay or load X_k (blue points) vs. observations k for horizon $n = 50$; the optimal data offloading time when $k = 27, 29, 46, 47, 48$ and 50 where $X < a$.

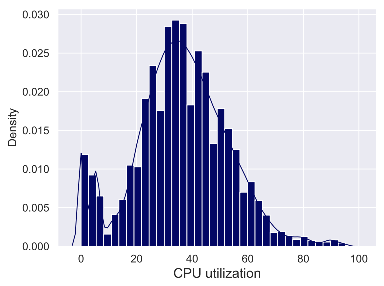

Figure 4: The distribution of the servers' CPU utilization.

schemes explained in the coming sections. In the mobility trace, we focus on the movements of 5 days (5000 rows of movements). An offloading decision was taken for each minute, i.e. we have to offload the data within one minute. Therefore, the value of n is set to be the maximum number of locations (observed CPU servers' utilization) within one minute. For example, if a car observes 5 servers within one minute, then the value of n is set to be 5. Now, we have more than 1000 offloading decisions. This will ensure to see the behavior of the proposed model for a long time. Fig. 4 shows the probability distribution of the servers' utilization for all servers in the data set. We can see that the servers' utilization in general follows normal distribution with $\mu = 36$ and $\sigma = 16$. We calculated the values of the threshold a using (7) and (8) and based on the mean and standard deviation of the servers' CPU utilization in the servers' data set. Specifically, consider one cab that has 5 movements during one minute in different locations. We set the value of $n = 5$ representing that the data get stale at $k = 5$. Each time, the AV checks (connects to) a server and checks the server utilization and compares it with the value of a_k. If the server utilization is less than the value of a_k, we select that server for offloading. If the AV does not find a server that satisfies the condition, we enforce the AV to select the last server.

B. *Performance Assessment*

We compare the Quality-aware OST-based data offloading model (QDO) discussed in Section III with the offloading models proposed previously namely DTO [22], BCP and COT with $c = 4$ [23], the Random selection model (Random), the p-stochastic model (p-model) for $p = 0.8$. In the BCP [23], as $n = 5$, the rule is to reject the first two servers, take the best among them as a baseline and start looking for a server that is better than the baseline. If we reach server 5 without offloading, we then must offload to server 5. The COT model [23] takes the probability distribution function $p(X_k)$ and a cost per observation (probing cost) c as inputs and outputs a threshold V^* for each cost c. The mobile node should offload if the observed processing time $X_k \leq V^*$, otherwise, the mobile node should continue observing until a defined deadline. By that time, the mobile node must offload to the first observed server. We set the value of $c = 4$, but different cost values can be used as shown in [23]. In Random, for each offloading session (one minute), we randomly select a server to offload the data. In p-model, for each offloading session, each server has probability p of being selected for data offloading (not selected with probability $1 - p$). We assign a probability of offloading p to be $p = 0.8$. 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 data offloading. The goal of this model is to enforce the mobile node to offload at the first observed server which is a simulation of a situation where the mobile node tries to offload at the first observed 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 CPU utilization for each car. The closer a model is to Optimal, the better the model performs in terms of the data offloading decision. Note that the optimal is not available in the considered scenario as the mobile node is independent in the decision making and only knows about the current network and server status. We obtain **the average server utilization** suggested by each model and the results are in Fig. 5 where we show the expected CPU utilization along with the 95% Confidence Intervals. We also show the difference between the Optimal and all the models in Fig. 6. We can see from the two figures that the QDO and the DTO are the closest models to the optimal with servers' utilization of (QDO,24.70) (DTO,24.60) and difference is less than 7. The COT is also achieving a good performance comparing to the rest of the models. The BCP model is performing better than the Random and the p-stochastic model.

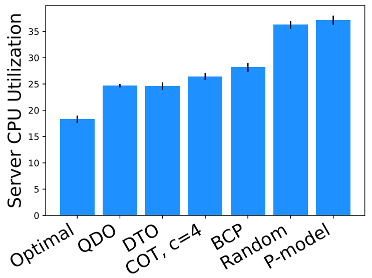

Figure 5: Expected CPU utilization selected by each model.

In terms of the offloading time (when the mobile node offloads), the p-model, most of the time, as it can be seen from Fig. 7 (average stopping times) will go with first server. This model has a high difference comparing to the Optimal model. Thus, going with the first observed server (time) is not a good decision. In other words, the mobile node should exploit the mobility and the deadline of the task to look for a better MEC server. From Fig. 7, we also observe that the BCP delays the offloading with higher expected offloading time.

We also evaluate our proposed model in terms of **the Successful Offloading Probability (SOP)** [30] which refers to the proportion of times a server with pre-defined CPU utilization was selected. The aim is to see how reliable the proposed model in selecting an offloading time when we have specific requirements [30]. For example, if the mobile node is looking for a MEC server with CPU utilization less than 20%, the SOP is the number of times the model selects a server with a CPU utilization less than 20% divided by the number of offloading decisions that were made in our experiment, i.e. $\approx$ 1728 offloading decisions. We set three CPU utilization thresholds: $\{18\%, 20\%, 25\%\}$. The results are shown in Fig. 8. We can see that the BCP [23] is performing better than the other models, but as mentioned earlier, it delays the offloading and it has higher expected offloading time which

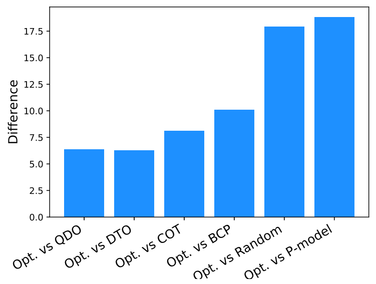

Figure 6: The difference between the Optimal and all the models.

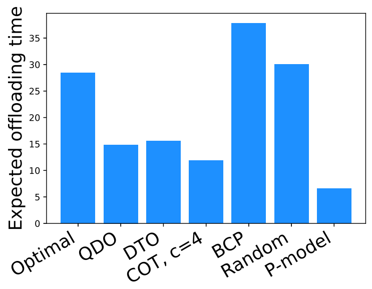

Figure 7: Expected offloading time for each model.

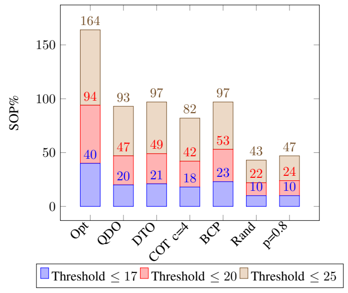

Figure 8: The Successful Offloading Probability (SOP) for each model based on different threshold values.

can affect the quality of the offloaded data being out of date. The proposed model QDO, DTO and COT have similar results and performing better than the Random and the p-model.

V. Discussion

The proposed model can be integrated with the current offloading architectures and frameworks. For example, considering the existing work in offloading decision framework in the smartphones devices, in general, the main components for the offloading framework are decision engine or code offloader, network, edge servers and applications profilers e.g. [31] [32] [33]. The previous components are envisaged to be implemented on a middle-ware on top of the smart-devices' operating system to perform code offloading framework [31]. The offloading engine, in general, is fed with the information collected from profilers. Based on the collected information, the decision engine is expected to give a decision of whether the task should be offloaded or run locally on the mobile node. The proposed model can be implemented within the decision engine component in mobile nodes, including future smart vehicles, and it can be triggered whenever the output of the decision is to offload to an edge server. Another important aspect of the proposed model is that the probability distribution function has to be known in order to calculate the threshold values. There are different methods to estimate the probability distribution function with the simplest method used above in our experiment the histogram utilizing the mean and the standard deviation. However, when such information is not available, we can utilize a nonparametric density estimation method [34] to estimate the probability distribution function utilizing historical data stored in the database of the offloading framework [31] [32] [33].

VI. Conclusions

With the envision MEC paradigm and the corresponding applications such as AVs' applications, we proposed optimized contextual data offloading decision-making model to be utilized by mobile nodes when dealing with tasks that involve collecting contextual data from the surrounding environment. Our evaluation showed that the proposed model is promising and can be easily implemented in MEC environments. In future work, we aim to implement the proposed model in real mobile nodes to evaluate the effectiveness of utilizing the OST in the data offloading decision in MEC environments.

VII. Acknowledgement

This research has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) project EP/N033957/1, and by the European Cooperation in Science and Technology (COST) Action CA15127: RECODIS Resilient communication and services. The first author is funded by Al-Baha University, Saudi Arabia.

References

[1] G. Brown, "Mobile edge computing use cases and deployment options," *Juniper White Paper*, pp. 1–10, 2016.

[2] Q.-V. Pham, F. Fang, V. N. Ha, M. Le, Z. Ding, L. B. Le, and W.-J. Hwang, "A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art," *arXiv preprint arXiv:1906.08452*, 2019.

[3] R. Akmam Dziyauddin, D. Niyato, N. Cong Luong, M. A. M. Izhar, M. Hadhari, and S. Daud, "Computation offloading and content caching delivery in vehicular edge computing: A survey," *arXiv*, pp. arXiv–1912, 2019.

[4] 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.

[5] D. Sabella, H. Moustafa, P. Kuure, S. Kekki, Z. Zhou, A. Li, C. Thein, E. Fischer, I. Vukovic, J. Cardillo *et al.*, "Toward fully connected vehicles: Edge computing for advanced automotive communications," *5GAA Automotive Association White Paper*, 2017.

[6] W. Tang, X. Zhao, W. Rafique, L. Qi, W. Dou, and Q. Ni, "An offloading method using decentralized p2p-enabled mobile edge servers in edge computing," *Journal of Systems Architecture*, vol. 94, pp. 1–13, 2019.

[7] M. Li, P. Si, and Y. Zhang, "Delay-tolerant data traffic to software-defined vehicular networks with mobile edge computing in smart city," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9073–9086, 2018.

[8] T. Ouyang, X. Chen, L. Zeng, and Z. Zhou, "Cost-aware edge resource probing for infrastructure-free edge computing: From optimal stopping to layered learning," in *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2019, pp. 380–391.

[9] S. Zhou, Y. Sun, Z. Jiang, and Z. Niu, "Exploiting moving intelligence: Delay-optimized computation offloading in vehicular fog networks," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 49–55, 2019.

[10] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.

[11] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the internet of vehicles," *Proceedings of the IEEE*, 2019.

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

[13] 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 *Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, vol. 1. IEEE, 2016, pp. 208–213.

[14] B. Silva, W. Junior, and K. L. Dias, "Network and cloudlet selection for computation offloading on a software-defined edge architecture," in *International Conference on Green, Pervasive, and Cloud Computing*. Springer, 2019, pp. 147–161.

[15] 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 *2018 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2018, pp. 74–79.

[16] 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.

[17] Z. Zhou, H. Yu, C. Xu, Z. Chang, S. Mumtaz, and J. Rodriguez, "Begin: Big data enabled energy-efficient vehicular edge computing," *IEEE Communications Magazine*, vol. 56, no. 12, pp. 82–89, 2018.

[18] M. Habib ur Rehman, P. Jayaraman, S. Malik, A. Khan, M. Medhat Gaber *et al.*, "Rededge: A novel architecture for big data processing in mobile edge computing environments," *Journal of Sensor and Actuator Networks*, vol. 6, no. 3, p. 17, 2017.

[19] M. Marjanović, A. Antonić, and I. P. Žarko, "Edge computing architecture for mobile crowdsensing," *IEEE Access*, vol. 6, pp. 10 662–10 674, 2018.

[20] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4568–4578, 2018.

[21] I. Alghamdi, C. Anagnostopoulos, and D. P. Pezaros, "Time-optimized task offloading decision making in mobile edge computing," in *2019 Wireless Days (WD)*. IEEE, 2019, pp. 1–8.

[22] I. A. I. Alghamdi, C. Anagnostopoulos, and D. Pezaros, "Delay-tolerant sequential decision making for task offloading in mobile edge computing environments," *Information*, 2019.

[23] I. Alghamdi, C. Anagnostopoulos, and D. P. Pezaros, "On the optimality of task offloading in mobile edge computing environments," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[24] N. Harth, C. Anagnostopoulos, and D. Pezaros, "Predictive intelligence to the edge: impact on edge analytics," *Evolving Systems*, vol. 9, no. 2, pp. 95–118, 2018.

[25] N. Harth and C. Anagnostopoulos, "Edge-centric efficient regression analytics," in *2018 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2018, pp. 93–100.

[26] C. Anagnostopoulos and K. Kolomvatsos, "Predictive intelligence to the edge through approximate collaborative context reasoning," *Applied Intelligence*, vol. 48, no. 4, pp. 966–991, 2018.

[27] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of management information systems*, vol. 12, no. 4, pp. 5–33, 1996.

[28] 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. 2018.

[29] "Alibaba cluster trace program cluster-trace-v2018," Downloaded from https://github.com/alibaba/clusterdata/blob/master/cluster-trace-v2018/trace_2018.md, November 2018.

[30] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-mec server systems," in *2018 IEEE international conference on communications (ICC)*. IEEE, 2018, pp. 1–6.

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

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

[33] D. Sulaiman and A. Barker, "Mamoc-android: Multisite adaptive computation offloading for android applications," in *2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. IEEE, 2019, pp. 68–75.

[34] A. Zhou, Z. Cai, and L. Wei, "Density estimation over data stream," 2015.