Alghamdi, I., Anagnostopoulos, C. and Pezaros, D. P. (2021) Data quality-aware task offloading in mobile edge computing: an optimal stopping theory approach. *Future Generation Computer Systems*, 117, pp. 462-479.

(doi: 10.1016/j.future.2020.12.017)

This is the Author Accepted Manuscript.

There may be differences between this version and the published version. You are advised to consult the publisher's version if you wish to cite from it.

https://eprints.gla.ac.uk/227273/

Deposited on: 18 December 2020

Enlighten – Research publications by members of the University of Glasgow
http://eprints.gla.ac.uk

Data Quality-Aware Task Offloading in Mobile Edge Computing: An Optimal Stopping Theory Approach

Ibrahim Alghamdi[1,*], Christos Anagnostopoulos[1], Dimitrios P. Pezaros[1]

[1] *School of Computing Science, University of Glasgow, Glasgow G12 8RZ, United Kingdom*

Abstract

An important use case of the Mobile Edge Computing (MEC) paradigm is task and data offloading. Computational offloading is beneficial for a wide variety of mobile applications in different platforms including autonomous vehicles and smart phones. With the envision deployment of MEC servers along the roads and while mobile nodes are moving and having certain tasks (or data) to be offloaded to edge servers, choosing an appropriate time and an ideally suited MEC server to guarantee the Quality of Service (QoS) is challenging. We tackle the data quality-aware offloading sequential decision making problem by adopting the principles of Optimal Stopping Theory (OST) to minimize the expected processing time. A variety of OST stochastic models and their applications to the offloading decision making problem are investigated and assessed. A performance evaluation is provided using simulation approach and real world data sets together with the assessment of baseline deterministic and stochastic offloading models. The results show that the proposed OST models can significantly minimize the expected processing time for analytics task execution and can be implemented in the mobile nodes efficiently.

Keywords: Mobile edge computing, tasks offloading, data quality, optimal stopping theory, sequential decision making.

1. Introduction

Mobile Edge Computing (MEC) refers to a computing paradigm that moves computing resources closer to the user at the edge of the network. MEC intends to relocate the cloud computing resources to the radio access network to optimize the delivery of content and applications to end users [1]. It involves deploying small data centres (servers) at the edge of the network in locations

*Corresponding author

Email addresses: `i.alghamdi.1@research.gla.ac.uk` (Ibrahim Alghamdi), `christos.anagnostopoulos@glasgow.ac.uk` (Christos Anagnostopoulos), `dimitrios.pezaros@glasgow.ac.uk` (Dimitrios P. Pezaros)

such as base stations or Road Side Units (RSU). MEC servers can be an advantage for various types of mobile nodes and applications including computation offloading for computation-hungry applications such as in Augmented Reality (AR) applications [2, 3]. Further, MEC servers can be an intermediate data-processing layer for data offloaded by mobile nodes [4]. **MEC servers within the RSUs** can play significant roles in improving the performance of mobile vehicular terminals, e.g. Autonomous Vehicles (AV) [5]. The AV can run intelligent vehicle control, traffic management and interactive applications using the built-in computation units. 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 [6]. The AVs can collect and sense contextual data and apply different algorithms for data analytics tasks. 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) [7]. However, despite AVs typically include on-board units, they have small-scale computing and storage resources because of which they are dependent on other computational resources [8]. Also, such applications may require significant computation resources and constrained time delays [6]. The AV then would be required to offload such tasks to one of the MEC servers to enhance its resources capabilities and to meet the applications' requirements.

Consider MEC servers deployed within RSUs as proposed in [6, 9] with mobile nodes passing by these servers as shown in Figure 1. These servers provide computing resources for task offloaded by mobile nodes. The mobile node can be a smart vehicle or a passenger on board who is running different types of applications. **The key problem** is the offloading decision by which the mobile node selects an edge server to offload the computing task as the MEC servers' load have large variation, e.g. sometimes there is a large number of users concurrently using the same server, whereas some other times only a few users are connecting [10]. In other words, the workload of such servers may be different over time [9, 10]. The selection of *where* (which MEC server) and *when* (time) to offload has a significant impact on satisfying the requirements of the AV applications [8]. Meanwhile, computing tasks or data gathered by AVs tend to have strict timeliness requirements which may result in the data becoming out-of-date [11]. Further, the existing studies in the area of computation offloading have mainly focus on the decision of whether the computing task should be offloaded or executed locally and a few studies have considered the selection of where a task should be offloaded as in [9, 12] . It is, therefore, vital to define rules by which the mobile node can select a suitable MEC server to be utilized for task offloading. *Once an offloading decision has been made and as the mobile node moves, should the offloading happen immediately or would rather delay the offloading for later in order to find a superior MEC server in terms of computing performance?*

For example, a naive centralized **offloading method** can be considered 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

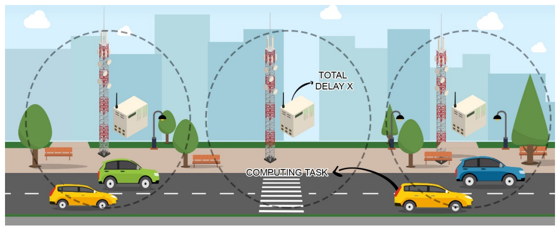


Figure 1: MEC in vehicular network.

[9, 13]. The centralized server can be located in a higher layer and it connects to all MEC servers with wired network connectivity [9, 13]. However, such architecture is not visible as this might introduce load on the network as the number of vehicles in the road increases [9]. Another solution would be using Vehicle to Vehicle communication (V2V) as discussed in [6], 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 solution requires the presence of other vehicles, which is not guaranteed all the time. Thus, the challenge now is how to optimize the decision of selecting the MEC server if the previous methods are not available and the mobile node only knows about the MEC server it is observing. In other words, the mobile node does not have a global information (or the mobile node has incomplete information) about the candidates 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 the 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 better MEC server with low workload increases [14]. 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 [9, 15].

Having such environment, we cast this sort of problem as an **Optimal Stopping Theory (OST)** problem. An OST problem is about deciding when to carry out an action on the basis of a random variable observed in sequence for the purpose of increasing the expected payoff or reducing the expected cost [16]. Secretary Problem (SP), the House Selling (HS) problem or the Fair Coin Problem are some of well-known OST problems [16]. We argue that the OST can play important role to optimize the task offloading decision. We try to provide light-wight and local algorithms that can be implemented in mobile nodes (vehicles or smart phones). As a result, the off-loadable computation applications will operate in an efficient way and optimally select when to offload in an independent manner.

The remainder of **this paper is organized** as follows: we provide a summary of the related work and outline our contributions in Section 2, while details of the system model and the problem formulation are described in Section 3. The OST-based decision making models are described in Sections 4.1 and 4.2.

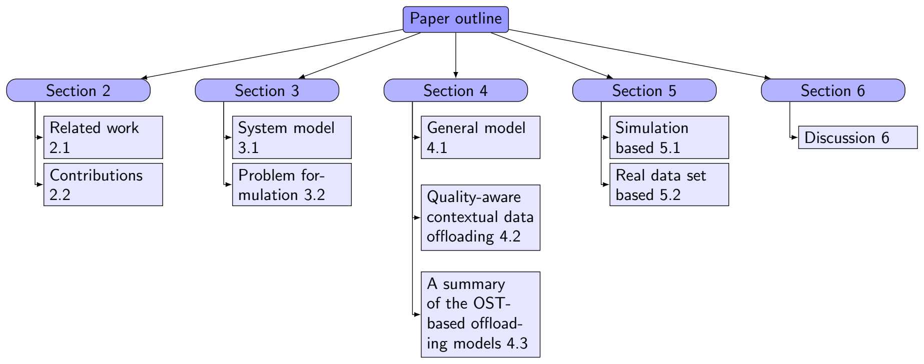


Figure 2: Paper organization.

Performance evaluation is provided in Section 5. Finally, a discussion about the models is provided in Section 6 and Section 7 concludes the paper and outlines future research directions. Additionally, the main sections of the paper are being visualized in Figure 2.

2. Related Work & Contributions

2.1. Related Work

Various pieces of research have been carried out to deal with the issues of offloading data and computing tasks to an edge node. The majority of which have emphasised if there should be a local processing of the data or task, or whether it should be offloaded externally, for example, to an edge server or to a public Cloud. There are two main objectives: minimization of (1) the execution delay and (2) energy consumption. A spatial and temporal computation offloading decision algorithm, ST-CODA [17], is related to our work. This work considers the decision-making of the mobile device in terms of the time and location for offloading tasks by considering the computation nodes and the transmission costs in edge cloud-enabled heterogeneous network. Our work's objective and policy differ from ST-CODA because the time-optimised sequential decision only offloads tasks to the edge servers and not further to the Cloud. In [17], the temporal decision refers to deferring the offloading decision until a low cost network is found, e.g., WiFi network. In our approach, we defer the offloading decision until a lightly loaded server is founded.

The work in [18] puts forward a strategy of computation offloading for a specific data mining application, namely, activity recognition for mobile devices. As the user moves, data is obtained from various places and is held in storage on the mobile device. This data is examined so that a choice can be made as to whether to offload to an edge server or the Cloud, or to carry out the process on the device. Should it be decided that offloading to an edge server takes place, the devices communication interface obtains a list of edge servers and connects to the best server. However, during the movement of the mobile device, e.g. in

AVs use case, a better server could be present and accessible, which is however not picked up by the communication interface whilst the devices communication interface scanning. Therefore, with regards to execution delay, it is possible that a better MEC server is available as will be discussed later. This opportunity is taken into consideration in our proposed scheme in this paper.

The work in [19], for the purpose of reducing the latency of task execution as well as energy consumption, presents the idea of collaborative mobile edge computing. This work focuses on UAV applications that uses photos and videos for tasks like object identification or obtaining traffic information. The captured photos/videos are then offloaded to an edge server. When the task is generated by the UAV, a system orchestrator should determine which server should be selected, what data rate ought to be adopted to transmit data to the selected server and how much workload each server (cooperator) should be allocated. This work [19] is based on the assumption that the system orchestrator makes this choice. However, in our work, the decision is autonomously made by the mobile node itself while in some situations there might be heterogeneous/different operators for the MEC servers.

The authors in [20] proposed a code offloading framework for offloading in a mobile fog environment. The proposed method determines which part of the application should be offloaded and takes an offloading decision considering the current state of the edge node resource by modelling the problem as Markov Decision Processes (MDP) and training it using the Q-learning approach [21]. Also, their algorithm supports the mobility of the user by migrating the offloaded part from one node to another. Their main objective was to minimize the delay of the offloaded applications. In this work, the feasible sites for offloading are: the mobile fog in close proximity, the adjacent mobile fog, or the remote public Cloud. In our OST-based approach, we assume that the mobile node can only offload to an edge server (potentially the most appropriate one with high probability) and there are a set of feasible MEC servers to offload.

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

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

The authors in [24] proposed a framework for joint network and virtual machine selection in cloudlet environment. The authors of this work considered the cloudlet architecture and the QoS of a face recognition application as an input for the proposed system. While the user is trying to offload in a corporate and campus networks, many WiFi access points might be available during the offloading sessions. Thus, the objective is to select the appropriate network along with VM resources that guarantee the quality of access to the cloudlet resources. This work assumes that all APs with their information are available to the mobile node, which is a different setting from ours.

The work in [25] and [26] investigated a multi sites offloading decision based on Analytical Hierarchy Process (AHP) multi-criteria method. The multi sites include the mobile device, near by mobile devices, cloudlet (edge server) and too far Cloud server. This study also contributed with the design and development of an Android offloading enabled framework that can be adopted by developers to build MEC applications. The assumption in this work is that the mobile devices will have to collect the offloading devices' information and based on such information the decision is made. Different from this work, in our work, the mobile node is not required to collect all the edge devices' information. In other words, in our setting, the mobile node is observing the edge nodes (sequentially) and does not need all the edge servers' information in advance for decision making.

The work in [14] considered the computation offloading in Vehicular Fog Network (VeFN). The authors provide a review of the offloading decisions work in VeFN. According to the authors, the offloading decision in the VeFN can be classified into three major modes: vehicle to vehicle, vehicle to RSU and pedestrian to vehicle offloading. This work provided two use cases: the first one is learning-based task offloading applying Multi-armed bandit (MAB) focusing on vehicle to vehicle offloading. The other use case is a delay-constrained task replication exploiting vehicle mobility where RSU collects task by vehicle and pedestrian and offload it to near by fog vehicle. An interesting point considered by the authors in this work is that mobility is not always an obstacle and can be supportive factor to help in finding a better resources for offloading. This is an important aspect we build our model on where we allow the mobile node to explore the MEC servers with the objective of finding a better resource.

The authors in [27] consider offloading decision and resource allocation in MEC environment by applying a reinforcement-learning-based state-action-reward-state-action (RL-SARSA) algorithm. The main goals are to balance the processing delay and the energy consumption when offloading, to define where to offload the task and to provide efficient resource allocation in the MEC servers. This study takes the advantage of adjacent edge servers as well as the remote execution to improve the decision of where to offload the task. In particular, four sites are considered for offloading the task: local execution, nearest edge server, adjacent edge server, and remote execution in the Cloud. The proposed RL-SARSA was compared with the reinforcement learning based Q learning (RL-QL) and the results show the superiority of the RL-SARSA over RL-QL. The limitation of this work is that the proposed model is more centralized and it is executed through edge computing controller in each region as indicated by the system model within the paper. This is different from our proposed approach where we try to make the mobile nodes more dependent and run the decision making algorithm locally at the mobile nodes.

The work in [28] proposed a distributed and context-aware task assignment algorithm in MEC environment. The task assignment in this work is refereed to the decision of where a task should be offloaded. The problem is formulated as an one-to-many matching problem by taking into account the devices and MEC servers computation capabilities, wireless channel conditions, and delay constraints. The main objective of this work is to reduce the overall energy consumption while satisfying task owners heterogeneous delay requirements. The proposed work is compared with Random matching scheme where tasks and edge nodes are randomly paired together, the scheme where the edge nodes with higher computational capability has a higher priority to accept tasks and the the centralized method, i.e., a centralized authority with complete information searches through all possible combinations to find the optimum solution. The proposed solution was the closest one to the centralized method in terms of energy consumption and the average utility. Even though the authors' main idea in this work is to make a distributed tasks assignments, still, the nodes including the edge nodes and the mobile nodes are required to collect information from all the neighbors devices before making the decision of offloading which is different from our work, where the mobile node proceeds sequentially for decision making without having to know about the edge nodes in advance.

The authors in [9] 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. The idea is based on Edgecoin virtual currency where MEC servers and vehicles can store the entire history of the Edgecoin transactions (every transaction by every vehicle), and every workload update transaction by every online mobile edge server. Based on the previous architecture, two algorithms were proposed. The first one is for the generation of all candidate mobile edge server(s) in the vehicle moving direction. The output of the first algorithm is the set of MEC servers to be utilized based on their service ranges. An R-tree was constructed to generate the set of MEC servers that are good enough to be used by the moving vehicle. The second algorithm is for determining the optimal MEC server from the generated list to be used by the moving vehicle. 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 work requires the mobile node (vehicle) to be involved in P2P network which might not be available for the mobile node all the time.

The OST was adopted in [12] 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 in [12] try to enhance the ability of an OST-based model by utilizing layered learning mechanism to define the OST thresholds and the sequence of the edge nodes used for offloading. However, such enhancement results in significant computational 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 in advance.

In our previous work [29, 30, 31], we proposed a set of lightweight sequential decision making models adopting the principle of OST. In particular, in our work in [29, 30], we proposed a Delay Tolerant Offloading (DTO) decision making in mobile edge computing environment. In this work, the problem of task offloading decision was cast as an finite horizon OST (decision should be taken within a predefined time horizon). Our main goal was to have a minimized total delay when offloading a task. Different real world data sets were utilized for evaluation. This work was enhanced in [31] by introducing two types of OST based models. The first one, which we call the Best Choice Problem based optimal task offloading policy (BCP), is to maximize the probability of offloading to the best edge server. The advantage of such model is that the task offloading decision can be made by the mobile node independently and it only requires the number of observations (number of edge nodes) to be provided. The other model we introduced in [31] is the Cost-based Optimal Task Offloading Policy (COT) model where we tried to enhance the performance of the BCP when more information about the performance metrics is available taking into account a cost per observation. The processing time and the delay of MEC servers in this work were simulated following specific probability distributions utilizing real mobility trace. In general, the results show that the OST based models were the closest ones to the Optimal, in which we select the edge server with the minimum total expected delay.

In this work, we first revisit the BCP model proposed in [31] and provide more insight about its optimality. Second, assuming we can incrementally learn the probability distribution of specific variables used for decision making we are observing within *data timeliness constraints*, we adopt the Odds [32] stochastic scheme within the context of the OST to maximize the probability of offloading to the best server based on a threshold provided by the mobile node. Additionally, we provide comparative evaluation of all the OST-based models found in the literature with others offloading methods using simulation-based evaluation and real data set evaluation.

2.2. Contributions

In summary, our contributions are:

- Departing from our previous work in [31], we propose a new quality-aware OST-based model to tackle the objective of maximizing the probability of offloading to the best server (i.e., the server with the minimum processing time) considering the timeliness of the collected data for data-oriented (analytics) tasks.

- We introduce the concept of quality-aware contextual data task offloading represented as a (linear) function over the timeliness of data, which is injected in the dynamic decision making.

- We enhance the Odd OST algorithm with the timeliness function; the optimal decision probabilities (Odds) depend on the freshness of data collected and the current load of the edge serves.
- We provide comparative assessment and extensive sensitivity analysis of our models with other offloading methods found in the literature using real data sets.

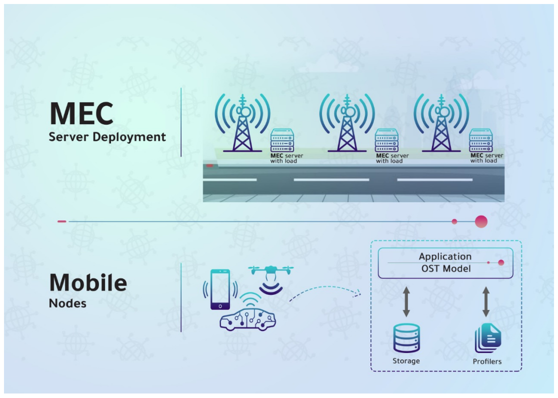


Figure 3: MEC servers and mobile node settings.

3. System Model & Problem Formulation

3.1. System Model

We consider a setting where there exists a finite set of MEC servers deployed along mobile nodes' paths on they move as shown in Fig. 3. These servers are deployed within RSUs or within base stations and are equipped with storage units and computing units [33]. Such setting can be seen in vehicular network applications as studied in [6] and [9] where smart vehicles perform different types of tasks. For instance, a mobile node[1] can offload contextually collected data to perform data analytics task on one of MEC servers. The mobile node can access the RSUs using Vehicle-to-Infrastructure (V2I) communication mode using dedicated short-range communication (DSRC). As the MEC servers operate at the network edge of radio access networks with the help of the RSUs, their coverage areas may be limited by the radio coverage of the RSUs [6]. Thus, the mobile node only knows about the *current* MEC server, i.e., the server in the range of that node. Unlike the centralized architecture, we consider the case where there is no centralized controller or server to assist the mobile node to make the decision. Instead, the mobile node is responsible for making the

[1]A mobile node refers to a smart vehicle or smart phone in vehicle used by passenger.

offloading decision autonomously and locally. We elaborate on the existence of an offloading decision framework implemented in the mobile node from previous work, which provides the entity of network and edge servers profilers as studied in [24]. Such profilers are adopted to provide information about the current load and (or) experienced delay of MEC servers.

Let X_k be the random variable indicating the processing time of k-th observed MEC server.[2] Once a task is locally generated and needs to be offloaded, then, at each time, within the number of observed MEC servers n, the mobile node checks the value of X_k for each MEC server k it passes by using network and server profilers. The mobile node needs to decide whether to offload to the current k-th server or continue observing another server in order to minimize the expected processing time $\mathbb{E}[X_k]$. To keep the continuity of task processing, we assume that there is a mobility management entity in the server [34] which implements a mobility management algorithm, such as path selection, power control algorithms [34, 35] or predictive model as in [6]. For example, if the task involves getting some results from the MEC servers and the mobile node is out of the range of that MEC server, then the selected MEC server should be transmitting the results through the next MEC server using high bandwidth wired connection [9].

3.2. Problem Formulation

Now, our goal is to optimize the decision on *when* to offload the tasks to an available server. Formally, our objectives are: (i) *to maximize the probability of offloading to the optimal server* and (ii) *to minimize the expected value of random variables X_k, i.e., $\mathbb{E}[X_k]$*. These two optimization objectives have been considered in our previous work in [29, 30, 31]. In this work, however, we elaborate on the first objective, i.e., maximizing the probability of offloading to the optimal server. In the following sections, we first revisit the BCP model proposed previously in [31] with more details. After that, we introduce a Quality-aware Contextual Data Offloading based on the Odds algorithm that takes into consideration the timeliness of the data to be processed in analytics tasks. Later, we provide a comprehensive assessment for the OST based models along with other offloading methods. In Table 1, we provide the key notations used in this paper.

4. Maximizing the Probability of Offloading to the Best Server

4.1. General Model

We first provide a review of the BCP model [31] that deals with the objective of maximizing the chance of offloading to the best server where the number $n > 0$

[2] X_k can indicate different random variables, e.g., the transmission time coupled with the computational workload of a server or the time it takes the MEC server to broadcast the results to other system, e.g., real time information for transportation system. For simplicity, we call it processing time throughout the paper unless otherwise specified.

Notation	Explanation
X	the random variable to be optimized
k	the index of the observed MEC server
n	the number of MEC servers or the number of observations
r_n	the optimal cutoff within the BPC model is taken by $r_n - 1$, r_n can also refer to the Odds in the observation n
$P_n^*(r_n)$	the maximum probability of ending up with the best server when applying the BCP policy
r_k	the Odds for observation k
P_k	the probability of having X_k less than or equal to a threshold within the Odds model
f	data quality indicators (timeliness)
s	the stopping threshold in the Odds model from which we start check a MEC server
$P_s^*(r_s)$	the maximum probability of ending up with a server meeting the required threshold when applying the Odds policy
R_s	the sum of the Odds from n until we reach or exceed the value 1
Q_s	the product of the complementary probability $(1 - P_k f_k)$ from n until s
θ	the threshold required within the Odds model
δ	the probability of having less than or equal to θ
T	the traveling time within the communication range of MEC server or RSU

Table 1: Key notations used in the paper.

of the available servers, which are candidates for task offloading, is known to the mobile node in advance. This can be defined by the user or estimated by the application based on the deadline of the task to be offloaded, e.g., offload task(s) to one of the next n available MEC servers. Also, it does not have to be the number of servers; it could be the number of time intervals the mobile node is going to observe (probe) the random variables X_k, $k = 1, \ldots, n$. For example, the mobile node might spend more time within the range of a server. In such case, we care about selecting the best time to offload the task within the time horizon of n probing time instances. Overall, 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 (relatively) ranked in the node from the best to the worst w.r.t. a performance criterion function over X. At each observation, the node should decide whether to choose the current available candidate server (or time) 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. This is due to the fact that, in our setting, the mobile node (AV for example) is moving in 1D mobility model. 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. 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 [16]. 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 k-th server *candidate*, if it is relatively best in terms of X_k, $k = 1, \ldots, n$. We then define a positive integer $r_n \in \{1, \ldots, n\}$, defined as:

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

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. For reasons of completeness, we provide **Theorem 1**[3], where the optimality of the BCP model is based on.

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

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

Proof : See [16].

For a small value of n, the optimal r_n can be computed using (1). When $n \to \infty$, we obtain the well-known Secretary Problem where the optimal probability tend to be $\frac{1}{e}$ [36].

Let us consider an example with a finite small n, e.g., $n = 3$. That is, there are 3 MEC servers in the AV's path. These servers have different processing times. We refer to the MEC server with the minimum processing time ranked with the number 1, and the server with the highest processing time ranked with the number 3. The MEC servers might come in different order. Thus, we have 6 permutations (3!). One policy is to reject the first server and take it as baseline and then accept the first relatively best server after that. If we follow such policy, 50% of time, we select the best one. In other words, there is 50% chance of offloading to best. If we increase the number of MEC servers by only 1 and follow the same policy, the chance of offloading to the best becomes close to 45%. As the number of MEC servers gets larger, the chance of offloading to the best gets smaller. Therefore, such policy does not work well with larger number of MEC servers and does not give the maximum probability which decreases with the number of MEC servers involved. Moreover, if the mobile node is offloading randomly to one of the encountered servers for e.g., $n = 10$ MEC servers, we have only 10% chance of offloading to the best server.

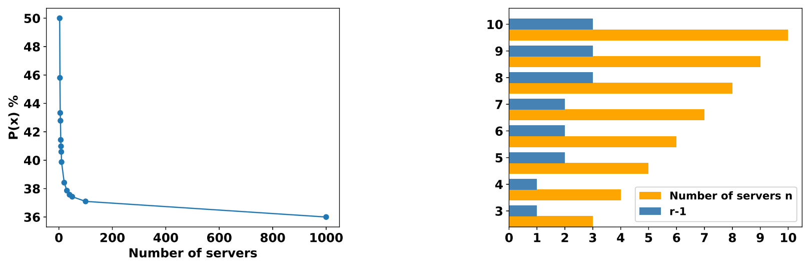


Figure 4: The probability of offloading to the best (left) and the value of $r - 1$ (right) for different numbers of MEC servers n [37].

In Figure 4, we show the (desired maximum) probability of offloading to the best (left) and the value of $r - 1$ (right) for different numbers of MEC servers

[3]For more information about Theorem 1 see [16] and [36].

n. We observe that as the number of MEC servers grows, we end up with the 36% chance of offloading to the best [37]. In reality, we expect the mobile node to have a number of MEC servers less than 10, thus, following the BCP's rule, we have a probability of offloading to the best $\geq$ 39%. Moreover, the BCP model only requires the number of observations n the mobile node is willing to observes. The number of observations can be defined and fed to the BCP model by the mobile node itself. Therefore, the decision making algorithm in this model is very independent and does not require relatively a lot of information.

Nevertheless, in the MEC environment, we expect that there will be other information in addition to the number of MEC severs n. Such information can be utilized to adopt and apply an advanced model within the context of OST. Moreover, it is possible to considers other requirements with a better performance in the decision of task offloading. Therefore, in the following subsection, motivated by the BCP model and aiming to achieve and optimize the same objective, i.e. maximizing the probability of offloading to the MEC server with the minimum processing time, we provide an advanced decision making algorithm with less dependency where the mobile node can provide more information in order to have better results. Focusing on offloading contextual data, the mobile node needs to know (or learn) the probability distribution function of the random variable it is trying to optimize along with the time constrains of collected data in an incremental manner, as will be discussed later. Our assumption is that we could improve the performance of the BCP model adopted previously in [31] and expanded above.

4.2. Quality-aware Contextual Data Offloading

In this section, we study the case that arises when a mobile node desires to offload contextual data to a MEC server and performs data analytics task while on the move. The data analytics task can be data correlation analysis, inferential and predictive analytic [38], statistical learning models building, model selection [39, 40] or data for HD maps as in [33]. The data can be gathered via different applications such as a mobile crowd-sensing (MCS) or vehicular crowd-sensing [41, 42]. For example, in vehicular crowd-sensing applications, vehicles sense data from surrounding environment, process them and send the processed results within a specific deadline to a centralized application manager for further processing [42]. In this use case, beside the main objective (maximizing the probability of offloading to the best server), the mobile node wants to offload contextual data to perform an analytic task *before* the data turns obsolete (stale). To deal with this quality of analytics problem, we elaborate on the the Odds algorithm within the context of the OST enhanced with data quality indicators in the offloading decision task.

Let $f : \mathbb{T} = \{1, 2, \ldots\} \rightarrow [0, 1]$ represent how *stale* the data is, which is a non-increasing function adapted from [43]:

- f is non-increasing in $\mathbb{T}$,
- $f_0 = 1$, where $k = 0$ is the start time before collecting the first data,

- $f_n = 0$, for $k \geq n$.

A linear timeliness function f is as follows:

$$f_k = \begin{cases} 1 - \frac{k}{n+1}, & 1 \leq k < n. \\ 0, & k \geq n. \end{cases} \tag{3}$$

The Odds algorithm is an OST algorithm for computing optimal stopping rules in order to maximize the probability of stopping at the last observation which satisfies a specific criterion [32]. We call an observation that satisfies the defined criterion a *success*. To get more insight on the Odds algorithm, let us consider a mobile node that is sensing data while on the move. The mobile node is trying to offload the data to a MEC server that has a processing time less than or equal to a desired threshold θ defined by the application launched on the mobile node. The Odds of the observed server k denoted by r_k is defined as the ratio of the probability P_k of having the MEC server with a processing time X less than or equal to θ divided by its complementary probability $1 - P_k$ [44]. Specifically, the Odds at time k is defined as follows:

$$r_k = \frac{P_k}{1 - P_k}, P_k < 1 \tag{4}$$

In each observation k, we take into account the Odds $r_1, \ldots, r_k$ as well as the timeliness $f_1, \ldots, f_k$ of the collected data by evaluating the function in (3). Hence, we obtain that:

$$r_k = \frac{P_k f_k}{1 - P_k f_k}, \tag{5}$$

where the Odds r_k depends now on how stale the data are at time instance k. Let $P_k = P(X_k \leq \theta)$ denoted by δ, then we have:

$$r_k = \frac{\delta f_k}{1 - \delta f_k} \tag{6}$$

(3) can be reformed as:

$$f_k = 1 - \frac{k}{n+1} = \frac{n+1-k}{n+1} \tag{7}$$

Then, we can substitute (7) for f_k in (6):

$$r_k = \frac{\delta \frac{n+1-k}{n+1}}{1 - \delta \frac{n+1-k}{n+1}} \tag{8}$$

(8) can be simplified as:

$$r_k = \frac{\delta(n+1) - \delta k}{(1-\delta)(n+1) + \delta k}. \tag{9}$$

Note that the Odds r_k changing with the time (k) as a non-linear function of the observation k reflecting the constraints of the data timeliness while engaging the application specific threshold $\delta = P(X_k \leq \theta)$ for assessing the appropriateness of the k-th MEC server. Figure 5 shows the evolution of the Odds r_k against observation k for different δ values in $\{0.3, 0.5, 0.8\}$ with $n = 10$. The Odds values decrease as we approach the end of (candidate) MEC observations, while a high application threshold $\delta = P(X_k \leq \theta)$ increases the Odds at the beginning of the selection process (being optimistic due to a relatively high δ). However, as $k \to n$, the Odds shrink to a very low value to *enforce* the decision of offloading to be taken, thus, avoiding offloading stale data (at $k = n$ with $f_n = 0$).

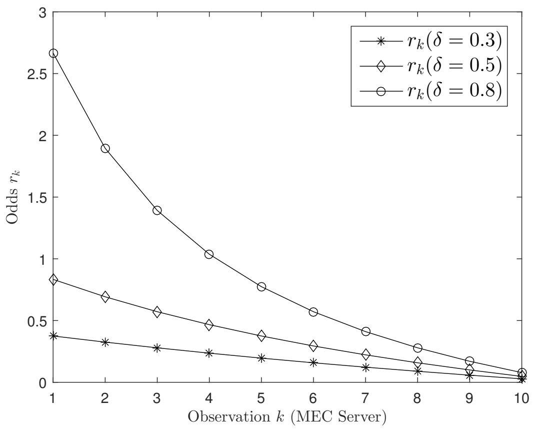


Figure 5: The odds r_k against observation k for different δ values; $n = 10$.

Let us now elaborate on the modified Odds algorithm that takes into consideration the data timeliness indicator f_k in the optimal task offloading decision. Specifically, the traditional Odds-algorithm applies to a class of problems called last-success-problems. The objective is to maximize the probability of identifying in a sequence of sequentially observed independent events the last event satisfying a specific criterion.

In our context, we aim at maximizing the probability of offloading to the *last* MEC server with $P_k = P(X_k \leq \theta)$ for a given θ threshold such that P_k is higher than all preceding probabilities $P_l, l = 1, \ldots, k-1$ seen so far. And, this decision must be taken at the time of observation. Hence, the very last MEC server with the above-mentioned criterion is the highest *bid*. Maximizing the probability of offloading on the last k-th MEC server with $P(X_k \leq \theta)$ therefore means maximizing the probability of offloading to the best MEC server w.r.t. θ. In the proposed quality-aware data offloading, we principally add the timeliness indicator f_k to the P_k, thus, now r_k represents the Odds of the k-th event turning out to be candidate for data offloading.

The Odds-algorithm sums up the Odds in reverse order:

$$r_n + r_{n-1} + r_{n-2} + \cdots ,$$

until this sum reaches or exceeds the value 1 for the *first* time. Let us denote that this happens at observation s, i.e., the corresponding sum R_s exceeds 1 with

$$R_s \quad = \quad r_n + r_{n-1} + r_{n-2} + \cdots + r_s. \tag{10}$$

If R_s does not reach 1, then we set $s = 1$. Also, at the same time we compute the product:

$$Q_s \quad = \quad \prod_{k=s}^{n} (1 - P_k f_k). \tag{11}$$

Based on the R_s and Q_s, we then apply the Odds algorithm to determine the optimal strategy for offloading to the best MEC server. The optimal strategy is as follows:

Quality-aware Odds Strategy: The mobile node observes the MEC servers one after the other and decides to stop on the first MEC server from time s onwards (if any), where s is the stopping threshold such that $R_s \geq 1$.

This strategy is optimal, that is, it maximizes the probability of stopping on the last best MEC server. And, this is happening with the maximum probability which equals to:

$$P_s^*(r_s) \quad = \quad Q_s R_s : R_s \geq 1. \tag{12}$$

Note that $P_s^*(r_s)$ is always at least 0.368 and this lower bound is best possible, which is achieved by the BCP policy with a very large number of MEC servers n. It is also worth mentioning that the Odds-algorithm computes the optimal strategy and the optimal probability $P_s^*(r_s)$ at the same time. Also, the number of operations of the Odds-algorithm is sublinear in n.

Let us apply this optimal methodology in practice. The mobile node should reject the observations (MEC servers) from $k = 1$ until s and from the observation s, the mobile node starts checking each observation (candidate MEC server). If it is a success, i.e., $X_k \leq \theta$ for $k > s$, then, the mobile node should offload the data to the k-th MEC server, otherwise it continues observing until f_n, i.e., where the data must be offloaded since $f_n = 0$. As an example, assume that MEC processing time $X \ \mathcal{N}(50, 10)$ follows normal distribution with mean 50 ms, a standard deviation of 10 ms for a specific data size and analytics tasks and the data on the mobile node must be offloaded within the next $n = 10$ observations. The timeliness of the data can be specified by the task application (it can be the number of time intervals or it can be the number of MEC servers the mobile node should observe before $f_n = 0$). If we assume that the mobile node will have $n = 10$ observations and the mobile node is looking for a MEC server with processing time less than $\theta = 50$, then, based on the Odds algorithm enhanced with f_k timeliness indicators, the strategy suggests to start looking for a MEC server to offload from $k = 5$ and onward. By doing this, there is $\approx$ 42% chance of offloading to the (last) best MEC server , which is the maximum that can be achieved. The probability here refers to the situation where we end up with a MEC server with processing time less than 50, thus, satisfying our criteria.

In real world scenarios and in the long run of a mobile node application, the MEC servers' provider can provide the probability distributions of the random variable X_k of the MEC servers based on the locations of the mobile node. Alternatively, the mobile node itself can use the historical data of the task offloading to learn the probability distribution. Once we can estimate the probability distribution of the processing time, the mobile node can estimate, based on the model above, where it should start checking the performance criterion in order to maximize the probability of offloading to a MEC server that meets the defined condition.

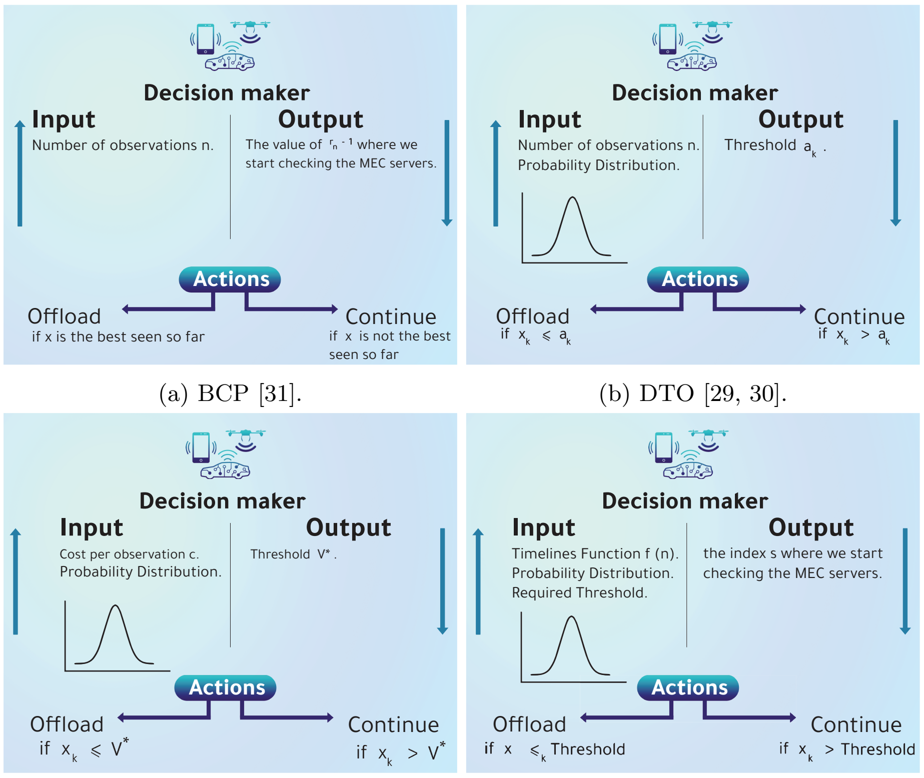
4.3. A Summary of the OST-Based Offloading Models

To review the proposed models, including our previous work in [29, 30, 31] and the model presented in subsection 4.2, they are visually described in Figure 6. As shown in Figure 6a, the BCP model takes the number of observations n as an input and outputs the numbers of servers that should be rejected before considering a MEC server for offloading. The mobile node should offload if the processing time/load X_k is the best seen so far, otherwise, the mobile node should continue observing until the server n. By that time, the mobile node must offload to server n.

The DTO model [29, 30], shown in Figure 6b, takes the number of observations n and the probability distribution function $p(X_k)$ as inputs and outputs a scalar decision threshold a_k for each server $k = 1, \ldots, k$. The mobile node should offload if the observed processing time/load $X_k \leq a_k$, otherwise, the mobile node should continue observing until server n. By that time, the mobile node must offload to server n.

The COT model [31], shown in Figure 6c, 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.

The proposed data quality-aware Odds model (subsection 4.2), shown in Figure 6d, takes the probability distribution function, a timeliness function/indicator f_k and a defined threshold as inputs and outputs the numbers of servers $s < n$ that should be rejected before considering a MEC server for offloading. The mobile node then starts evaluating the condition based on the required threshold θ. If the condition is true, then, the mobile node should offload, otherwise, the



(a) BCP [31]. (b) DTO [29, 30].

(c) COT [31]. (d) Quality-aware Odds Model.

Figure 6: A summary of the OST based offloading models.

mobile node should continue observing until $f_n = 0$. By that time, the mobile node must offload before the data turns obsolete.

It should be noted that the previous models can be also applied to a situation situation where the mobile node is moving within the range of one MEC server and tries to choose a time instance (within a specified time horizon n) with minimized processing time. In such case, the horizon n can be divided into time slices and then we obtain the offloading rules based on the procedure of each model. The decision (which server *or* which time instance) is based on the mobility of the mobile node as well as on the density of the MEC servers deployment. For example, in AV scenario as considered in [6] and [9], we could go for MEC server selection especially if there is high mobility and high density deployment of the MEC servers.

5. Performance Evaluation

We use two settings to evaluate the proposed OST based models: simulation based evaluation and real world data sets based evaluation. In both settings,

Parameter	Value / Range
X	$\mathcal{N}(50, 10)$ & $\mathcal{U}(0, 1)$
Number of mobile nodes	1000
n	$\{3, 5, 10\}$
θ	$\{30, 40, 50, 60\}$ & $\{0.3, 0.4, 0.5, 0.6\}$
c	$\{1, 2, 3, 4, 5, 20, 30\}$ & $\{0.1, 0.3, 0.4\}$
p for the p-model	0.8

Table 2: Simulation experiment parameters' values.

we compare our OST-based offloading models namely BCP (subsection 4.1), the proposed quality-aware Odds model (subsection 4.2), DTO [29, 30] and COT [31] with the Random selection model (Random), and the p-stochastic model (p-model; which will be discussed later). 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 processing time for each offloading session. The closer a model is to the Optimal, the better the model performs in terms of the task offloading decision. In the following subsections, we provide the details and the results of each setting. Table 2 shows the values of the parameters of the simulation experiment.

5.1. Simulation Based Evaluation

In the simulation evaluation, the probability distribution of the random variable X, e.g., the processing time, will be known in advance. To simulate the environment of MEC, we used `Simpy` in Python [45]. `Simpy` is a process-based discrete-event simulation framework. Each MEC server k is modelled as a resource that advertises its processing time X_k each time during the simulation. The mobile node is modelled as a process that passes by the MEC servers in 1D mobility model and checks the processing delay advertised by each MEC server. We first consider five MEC servers, i.e. $n = 5$. The processing time X is drawn from normal distribution with $\mu = 50$ ms and $\sigma = 10$ ms and it was generated using `Python` function that generates random numbers following a specified distribution, i.e. normal or uniform distribution. It should be noted that the processing time has been named in the literature with different terms including total delay [34], latency [2] or waiting time [9]. Also, the range of the processing time varies according to the application types. As an example, it is being ranged from 0.1 seconds to ≈ 800 seconds in [46] and in the range of 10 ms to ≈ 30 ms as in [9]. Therefore, different values or scales of X can be used with the proposed models generating similar results as we will see in the real data set experiment. Each minute, a mobile node (in e.g., a vehicle/car) starts checking the MEC servers. It starts with server number 1 and applies the proposed models above to select a MEC server for offloading. In the BCP, the rule, based on Figure 4 (right), 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. In the

DTO, each server k is compared with the decision threshold a_k as proposed in [29, 30]. If the processing time X_k is less than the threshold a_k, the mobile node should offload, otherwise, it continues till it reaches the last server, and then it must offload to the last server. In the COT model [31], there is a unique solution for V^* for each value $c > 0$. We obtain the the threshold values V^* for each cost value $c \in [1, 50]$. As plotted in Figure 7, we can see that the generated threshold V^* values are around the mean when $c \leq 10$. In our modelling, the value of the cost c is interpreted as the need for a lower processing time, but different interpretations can be obtained based on the application requirements. As a result, we can see from Figure 7 that low costs have higher thresholds and thus it will accept higher processing time. In contrast, higher costs have less thresholds and thus it will look for less processing time. Once the value of c is defined, the value of V^* can be obtained as shown in our previous work [31]. We start by defining the cost to be $c = 4$, and later, we show the performance for different values c. Starting from server 1, if the processing time X_k is less than that threshold V^*, we offload to that server. If we reach server 5 without offloading, we then must offload to server 5 [4]. In the quality-aware Odds model, we first define $\theta = 50$ as a threshold. Thus, based on the proposed model, the model suggests to start from server 2 ($s = 2$) and pick the server that has a processing time less than or equal to $\theta = 50$. Note that as n is set to 5, we calculate the indicators $f_1, \ldots, f_5$.

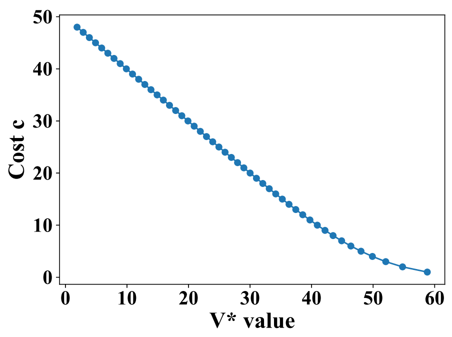


Figure 7: The V^* value for the processing time used in the experiment vs. cost c.

In the Random selection model, for each user, we uniformly at random select a server to offload the task. In the p-model, for each server, we assign a probability of offloading; in this setting we experimented with $p = 0.8$. In each user's movement, each server has probability $p = 0.8$ of being selected for task offloading. 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. The p-model, here, is a simulation for scenario where the mobile node offloads at the first server as we have higher probability, i.e. $p = 0.8$, as we will see later in the

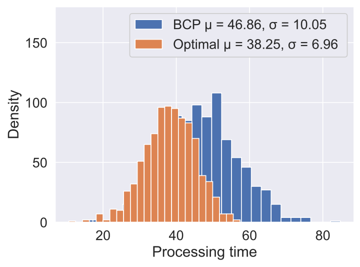
[4]For more details about the thresholds for the DTO and the COT models, see [29, 30, 31].

real data set evaluation. The Optimal model was captured at each offloading session by selecting the server with the minimum processing time.

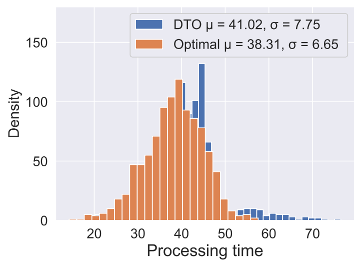
The results of the experiment are shown in Figure 8. We can observe that there is a noticeable overlapping between the Optimal and the OST based models (the DTO, the COT and the quality-aware Odds) as shown in Figures 8b, 8c and 8d, respectively. This overlapping decreases in the BCP, the Random and the p-model as it can be seen from the Figures 8a, 8e and 8f. However, the BCP model is achieving lower expected processing time ($\mu = 46$) than the Random ($\mu = 49$) and the p-model ($\mu = 50$). This is clear in Figure 9 as the difference between the Optimal and the OST based models including (BCP, DTO, COT, Odds) is significantly less than the Random and the p-model models.

It should be noted that, in general for the DTO and the COT models, the Optimal thresholds generated by each model for each observation k is close to the mean of the processing time, i.e. 50. For example, in the DTO, the generated threshold values $\{a_k\}_{k=1}^{n}$ for $n = 5$ are $\{46, 47, 48, 49, 50\}$. As we can see, the values are close to the mean of the processing time. This also applies to the COT model. Based on these Optimal thresholds, we first set the threshold value $\theta = 50$ for the Odds, and later we show the performance for different θ values. As it can be seen from the Figures, we had a good performance in the Odds. This good performance is due to the fact that we have around 42% probability of picking a server with processing time less than 50 [5]. Therefore, a lesson learnt here is that setting a threshold value close to the mean value achieves less processing time. We had better performance in the BCP model than the Random and the p-model as the BCP offloading policy has higher probability of offloading to the minimum processing time than the Random and the p-model as we observer earlier in subsection 4.1. This higher probability is translated into less expected processing time than the Random and the p-model. We should note that, we have similar probability of offloading the best in the BCP and the Odds, but having a defined threshold θ when checking the MEC server has increased the performance in the quality-aware Odds model.

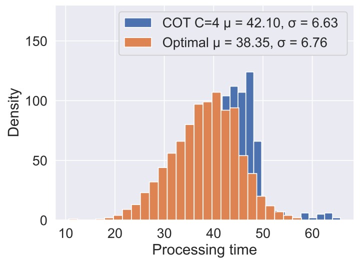
[5]This probability is calculated using equation (12).



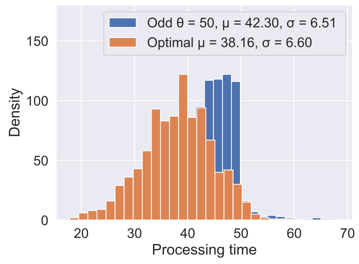
(a) BCP and the Optimal selections.



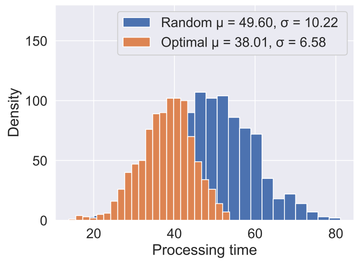
(b) DTO and the Optimal selections.



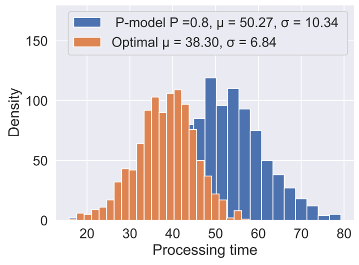
(c) COT and the Optimal selections.



(d) Odds and the Optimal selections.



(e) Random and Optimal selections.



(f) P-model and Optimal selections.

Figure 8: Simulation results for all the models when X normally distributed.

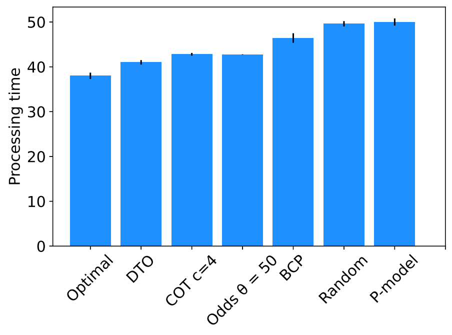
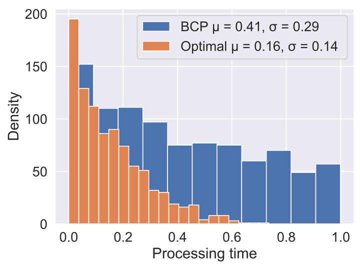


Figure 9: Confidence interval in the simulation experiment when X is normally distributed.

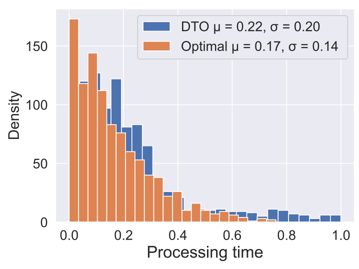
In the previous experiment, the random variable X we are trying to observe and optimize is following normal distribution. We now let X be uniformly distributed scaled in $[0, 1]$. This can refer to the server utilization, i.e., the CPU utilization. For example, 0.5 indicates that 50% of servers' CPU is utilized. We follow the same steps as we did in the previous experiment for all the models. In the COT model, we obtain the Optimal threshold V^* values for each cost value $c \in \{0.1, 0.2, 0.3, 0.4\}$. We first show the results when $c = 0.2$, where the Optimal threshold $V^* \approx 0.36$. Later, we show the performance for the rest of the values c. The interpretation for the cost is similar to the situation when we have X normally distributed, i.e. higher cost (small threshold V^*) means high demand for less processing time. In the quality-aware Odds model, we set the threshold $\theta = 0.5$, thus, there is around 42% chance of offloading to a server with $X \leq \theta = 0.5$. Although we have the same probability in the BCP model, but again, it turns out that setting a threshold can improve the performance of the Odds model. In general, we can see from the results shown in Figure 10 and Figure 11 that the models performance is similar to the results we obtain when X following normal distribution. We still have the DTO model perform the best, and in general, the OST based models are closer to the Optimal than the Random and $p-$model.

5.1.1. Sensitivity Analysis (Simulated Environment)

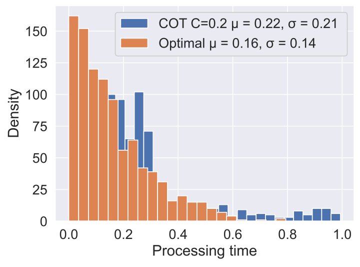
In this sections, we provide the models' results for different parameters values. We start by showing the results of the BCP model for different values of n. We observe that when n is small, the difference between the BCP and the Optimal decreases. For normally distributed X, the difference was 5.24, and 9.07 for $n = 3$ and $n = 10$ respectively as shown in the Figures 12a and 12c. This is also true when X uniformly distributed: the difference was 0.16, and 0.22 for $n = 3$, $n = 10$ respectively as shown in the Figures 12b and 12d. These results support Figure 4 (left), i.e. the probability of offloading to the best decreases and approaches 36% as n increases.



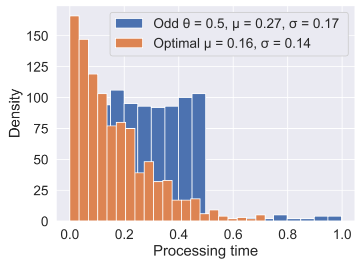
(a) BCP and the Optimal selections.



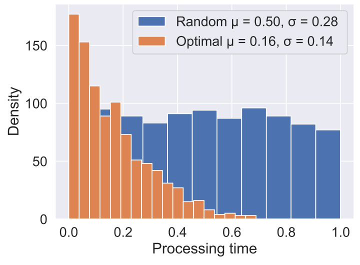
(b) DTO and the Optimal selections.



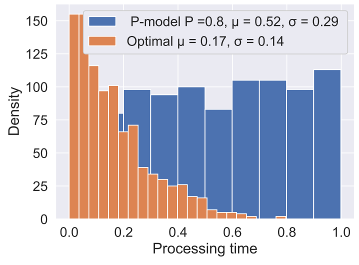
(c) CBT and the Optimal selections.



(d) Odds and the Optimal selections.



(e) Random and Optimal selections.



(f) P-model and Optimal selections.

Figure 10: Simulation results for all the models when X uniformly distributed.

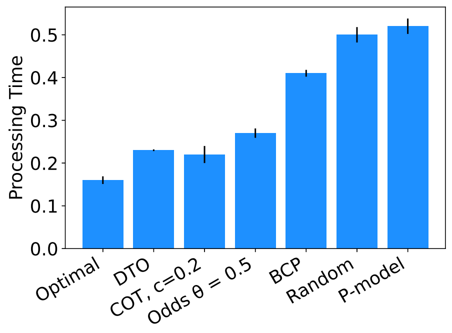
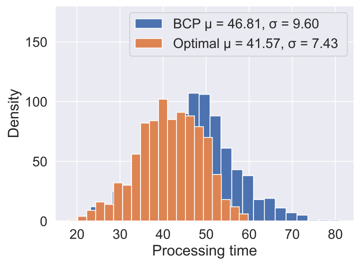
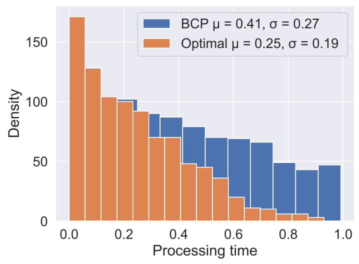


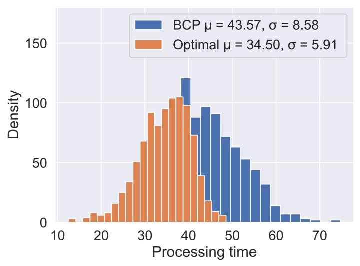
Figure 11: Confidence interval in the simulation experiment when X uniformly distributed.



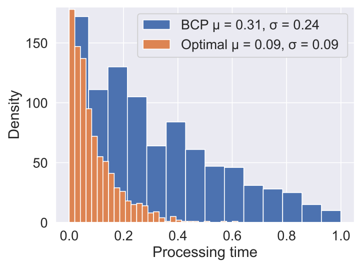
(a) BCP when $n = 3$ and X normally distributed.



(b) BCP when $n = 3$ and X uniformly distributed.



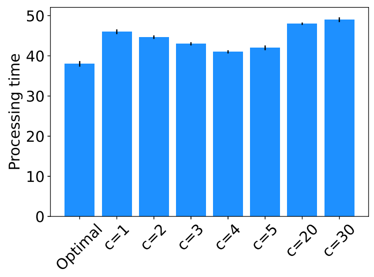
(c) BCP when $n = 10$ and X normally distributed.



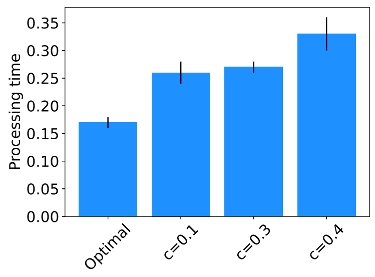
(d) BCP when $n = 10$ and X uniformly distributed.

Figure 12: BCP models for different values n.

We also study the COT model [31] for different values c. In our modelling, high value c will generate small threshold V^*. This can be interpreted as the high demand for small value of X. On the other hand, small value of c will generate large threshold V^*. Therefore, when we set the cost to a small value, then, it implies we are tolerant to expect higher value X. Figure 13 shows the confidence interval and the average processing time achieved by the COT model for different c values when X is normally distributed (Figure 13a) and when X uniformly distributed (Figure 13b). We can see from the results that the processing time achieved by the COT is higher when $c = 1, c = 20, c = 30$. When $c = 1$, V^* is high, thus, the mobile node offloads at the firs server. When $c = 20, c = 30$, V^* is very small, thus, it delays the offloading and in fact the mobile node did not find a server with processing time less than V^*. Therefore, the mobile node has to offload to the last server. The value of V^* was around the mean when $c = 3, 4, 5$. Thus, we had a closer processing time to the Optimal. This is also true for the uniform distribution, when $c = 0.1$, the value of the V^* was 0.5. Therefore, when the threshold V^* is close to the expected value (mean), the COT performs better.



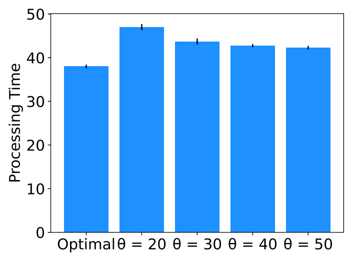
(a) Confidence interval when X normally distributed.



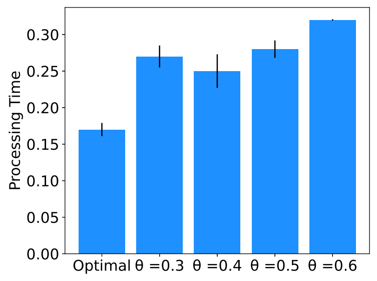
(b) Confidence interval when X uniformly distributed.

Figure 13: Confidence interval in the COT for different cost c values.

We also consider the quality-aware Odds model with different θ values. Figure 14 shows the confidence interval and the average processing time achieved by the quality-aware Odds model for different θ values when X is normally distributed (Figure 14a) and when X is uniformly distributed (Figure 14b). Similar to the COT model, we observe that when θ is close to the mean, the model performs better. As mentioned earlier, when $\theta = 50$, we have around 42% chance of offloading to a server with processing time less than or equal to 50. This is also true when $\theta = 60$, i.e. we have around 42% chance of offloading to a server with processing time less than or equal to 60. However, due to the higher value of the threshold, i.e. 60, we had higher processing time than the processing time when we set θ to 50. Setting the value of θ to 40 had an acceptable performance. We had higher processing time when $\theta = 30$ as we have small chance of offloading to a server with the specified thresholds. When X is uniformly distributed and when $\theta = 0.4, 0.5$ and 0.6, we have higher chance of offloading to a server meeting the specified thresholds, i.e. $> 40\%$. However, the performance was not good when $\theta = 0.6$. This reason for having high processing time is because the higher value of threshold (0.6), and thus, the mobile node will accept a server with higher processing time. Therefore, for the Odds model, in both setting, i.e. when X is $\mathcal{N}(50, 10)$ or $\mathcal{U}(0, 1)$, setting the threshold θ to value of the expected value or less by ≈ 10 achieves less processing time. We should note that each θ value has different stopping index s based on model presented in subsection 4.2. As an example, when $\theta = 30$, $s = 1$, and when $\theta = 60$, $s = 3$. This is also true when X is uniformly distributed, e.g. when $\theta = 0.3$, $s = 1$, and when $\theta = 0.6$, $s = 2$.



(a) Confidence interval when X normally distributed.



(b) Confidence interval when X uniformly distributed.

Figure 14: Confidence interval in the quality-aware Odds for different θ values.

5.1.2. Mobility Scenarios

As the mobility of the mobile node plays a key role in task offloading within the MEC environment, it is important to consider its effects when applying the OST based decision making. Therefore, we simulate a mobile node (i.e. smart vehicle) that moves in one direction with different velocity values uniformly distributed in $[1, 5]$ meters per second and it passes by a set of MEC servers. The communication range of the MEC servers is 100 meters. Five MEC servers over a distance of 1000 meters were deployed, i.e. one server each 200 meters. Figure 15 shows such a setting.

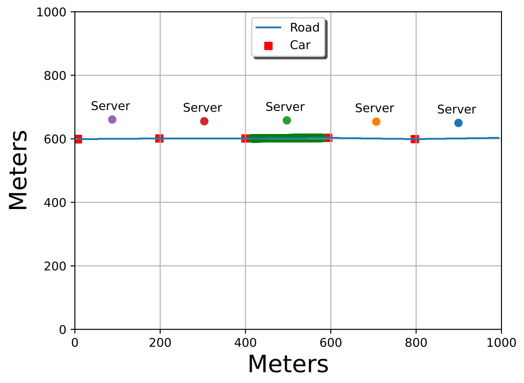


Figure 15: Mobility simulation.

As it can be seen from Figure 15, the car selects server 3 for offloading. The green line shows that the car was within the communication range of server number 3. Now we define the **traveling time** T to be the time it takes the car from the first point of the green line till the last point of the green line. The car offloads data to be processed before going out of the range. Having such settings, the mobile node will face one situation from two. **First**, when we have a processing time less than the traveling time. This case can arise when the load of the MEC server is light, e.g. due to the density of the vehicles being low [6] or the velocity of the mobile node is not high. In this case, the OST models have higher probability to select a MEC server that finishes the task before the mobile node gets out of the range of that MEC server with minimized processing time. To check this, a velocity $[1, 5]$ that generates traveling time higher than the processing time has been used. The processing time are assumed to follow normal distribution, i.e. $X\ \mathcal{N}(50s, 10s)$. The results show that the difference between the processing time and the traveling time when applying the OST models is higher than the other models: the Random, first selection as shown in Figure 16. The higher the difference the more reliability the model has. This indicates that the OST based offloading is more reliable offloading than the other offloading methods as the proposed models ensure the task finishes before going out of the range of the communication.

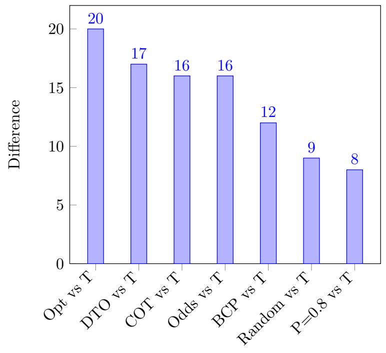


Figure 16: Absolute difference between the processing time and the traveling time T.

Second, when the processing time is higher than the time the mobile node spends within the communication range, i.e. $X > T$. In this case, we need a mobility management algorithm that handles such a mobility. Examples of such algorithms are power control for low mobility, and path selection or task migration for high mobility as stated in [34]. The adoption of the OST based selection in such scenario can be benefit. For example, in each method of the mobility algorithms, the first step is to make a selection for an edge node to offload to. Therefore, it is not difficult to consider the proposed models for this kind of scenario, but this is left for future work and out of the scope of the paper.

5.2. Real Data Sets Based Evaluation

We also consider real data sets to evaluate our models. The purpose of this evaluation is to see how our models perform when dealing with real data sets. To simulate the movements of the mobile nodes, we first used the real data set of taxi cabs' movements in Rome [47]. The data set contains GPS coordinates of 320 taxis collected over 30 days. For each row in this data set, we have the cab-id, date/time and GPS coordinates of the current location. 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 modelled as an observation or connection to a MEC server. Figure 17 shows the movements of the cars in Rome map.

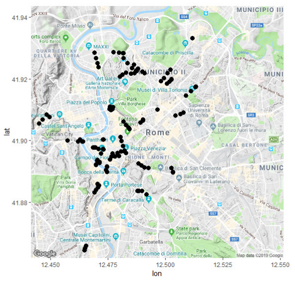


Figure 17: Taxis trajectories in Rome.

The processing time is represented in this experiment by real servers' utilization (the CPU utilization) obtained from [48]. In the servers' data set, we have around 150 servers' data (more than 1 billion rows). Thus, for each movement, the car picks a server from the servers' data set, checks that 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 model as explained earlier in the simulation evaluation section. We focus on the movements of over 5 days (5000 rows of movements). An offloading decision was taken for each minute. Thus, we have more than 1000 offloading decisions. This will ensure to see the behavior of the proposed models for a long time. Figure 18 shows the probability distribution of the servers' utilization for the all servers in the data set. We can see that the servers' utilization in general follows normal distribution with $\mu = 36$ and $\sigma = 16$. Also, for illustrative purposes, an example of one offloading decision session is shown in Table 3. In Table 4, we show the key parameters' values in this experiment.

Note that, as the server utilization is approximately following normal distribution, when we apply the OST models, we have to feed the models (Odds, DTO and COT) with the mean and the standard deviation. In this experiment, the mean and the standard deviation were taken once at the beginning of the experiment for the whole servers' utilization data set. In other words, we did

Cap id	Movement time	Location	Machine name	CPU utilization
156	2014-02-05 00:11:01	(41.8911, 12.49073)	m_1939	(51)
156	2014-02-05 00:11:11	(41.89905,12.4899)	m_1936	(47)
156	2014-02-05 00:11:22	(41.8994,12.48940)	m_1941	(20)
156	2014-02-05 00:11:31	(41.8994,12.489401)	m_1941	(37)

Table 3: A sample of the data set used in the experiment.

Parameter	Value / Range
X	real servers CPU utilization in $\mathcal{N}(36, 16)$
Number of movements	5000 movements
Number of offloading decision	> 1000
n	5
θ	$\{20, 30, 40, 50, 60\}$
c	$\{1, 2, 3, 4, 5, 20, 30\}$
p for the p-model	0.8

Table 4: Real data set experiment parameters' values.

825 not apply the models with the mean and the standard deviation of the observed utilization during the experiment. Instead, we only take this information once when we start the experiment. This is an important aspect of conducting this experiment as in real world scenario, the mobile node does not know the mean and the standard deviation of a specific MEC server, but can obtain this in-
830 formation from historical data for the MEC servers in one area in specific time with the help of MEC servers operators.

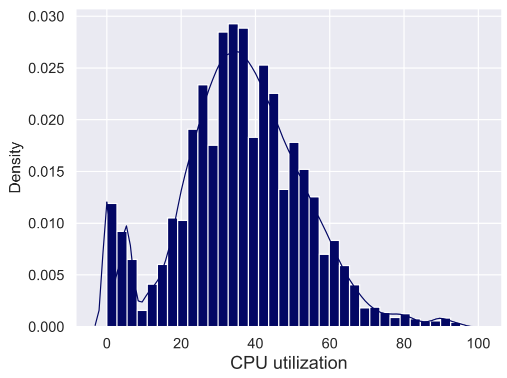


Figure 18: The distribution of the servers' CPU utilization.

Similar to the simulation setting, we compared 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 offloading decision session. For example, the Optimal in Table 3 is to offload at 00:11:22 with CPU utilization of 20%. The closer a model is to the Optimal, the better the model performs in terms of the task offloading decision. We run all models on each minute (offloading decision session) for evaluation. In short, each minute consists of around 5 movements. Each model selects a server for offloading as suggested by that model. We then take the average server utilization achieved by each model in all offloading decision session.

Figure 19 shows the average server utilization suggested by each model. We can see that the OST models are the closest models to the optimal. The DTO performs better than the rest of the models with absolute difference, compared to the optimal of 5 and it is higher than the Optimal by 23% as shown in Figures 20a and 20b.

In reality, the mobile node would normally offload to the first server or in the first time. A simulation for such case is the p-model with $p = 0.8$. This is clear in Figure 19 where the p-model has the lowest offloading times (offload earlier than other models). We can see from the results that the p-model is too far from the optimal. In other words, our results show that going with the first server (time) or (immediate server) is not a good idea. Moreover, which server/time is the Optimal is not known and not provided to the mobile node. In other words, in the considered environment, the Optimal is not available to the mobile node so having the OST-based model implemented in the mobile node can achieve near-optimal server utilization.

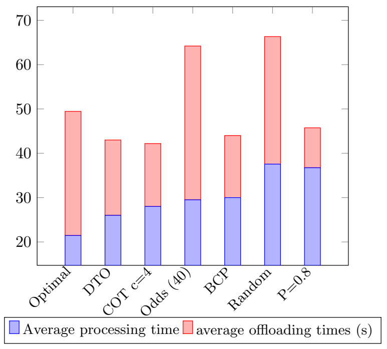
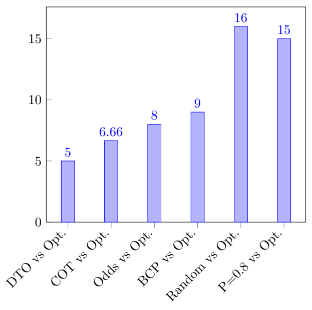


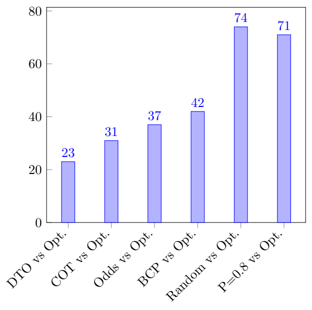
Figure 19: Average processing time and average offloading times suggested by each model.

5.2.1. *Sensitivity Analysis (Real Data Sets)*

To further investigate how significant the difference between the proposed models and the Optimal is, in Figure 21, we plot the confidence interval (95% confidence limits) for the results obtained by each models. We can see that the difference is more significant with the Random and p-model. Also, the significance in the difference increases in the BCP, and it decreases with the rest of the models.



(a) Absolute difference.



(b) Percentage difference.

Figure 20: Difference between the Optimal and all models.

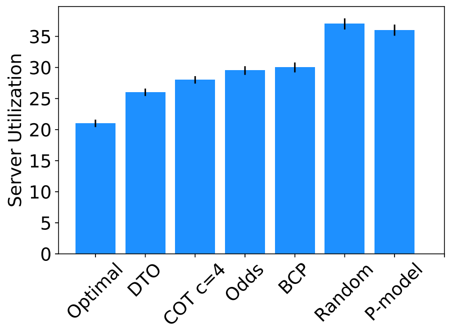


Figure 21: Confidence interval for the the real data sets experiment.

Figure 22 shows the V^* optimal threshold for the COT model vs the associated cost for the server utilization used in the experiment. Similar to simulation experiment, a lower cost c indicates accepting higher server utilization, whereas higher cost means a high demand for small server utilization.

Figure 23 shows the confident interval and the average utilization server achieved by the COT model for different cost c values. We observe that the server utilization is closer to the optimal when $c = 4$, $c = 5$ and $c = 6$. The V^* optimal threshold values when $c = 4$, $c = 5$ and $c = 6$ are 39, 37 and 34 respectively which are around the mean of the server utilization. Similarly, Figure 24 shows the confident interval and the average utilization server achieved by the Odds model for different θ values. It is also clear that when the threshold is close to the mean, the server utilization gets closer to the Optimal. This supports our findings in the simulation experiment.

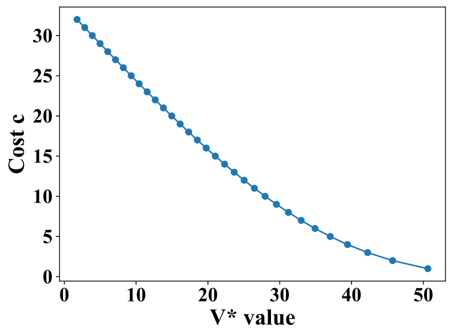


Figure 22: The V^* value for the server utilization used in the experiment vs. cost c.

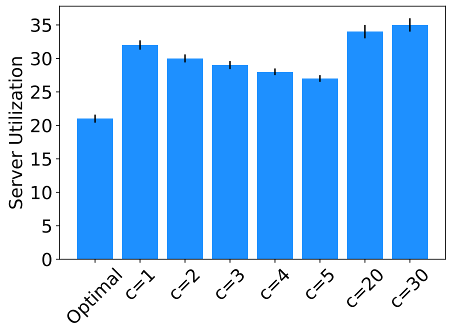


Figure 23: Confidence interval in the COT for different cost c values.

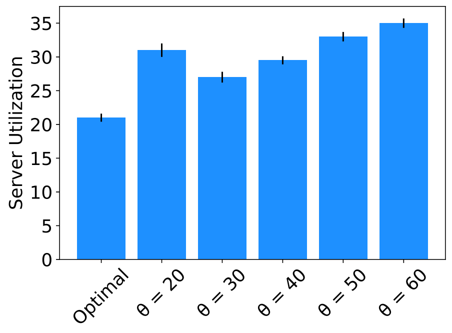


Figure 24: Confidence interval in the Odds model for different θ values.

In addition to the average server utilization as an performance metric, we use the number of successful offloading for each model. The number of successful offloading refers to number of offloading decisions, suggested by each model, that meets specific requirements. To have an idea about the number of successful offloading metric for each model, let's first assume that we have 3 different MEC applications x, y and z. Each of which has a specific requirement. For example, application x requires a CPU utilization less than 10, application y requires a CPU utilization less than 20 and application y is tolerant to offload to a server with CPU utilization less than 30. Now, if an offloading happens to server with utilization less than 10, we then consider that a successful offloading for application x.

Figure 25 shows the number of successful offloading for different resource requirements for all the models. For an application that requires, $\leq 10\%$ CPU utilization, the Optimal achieves 102 successful offloadings, i.e. 102 times the Optimal selects a server with a utilization less than 10%. In the second requirements, $\leq 20\%$, the Optimal had 463 successful offloadings. For the first and the second requirements, the Odds model was the best among the other models. In the third requirement, $\leq 30\%$, the Optimal had 887 successes. The Odds model was the closest one to the Optimal in the number of successful offloading with 797 success.

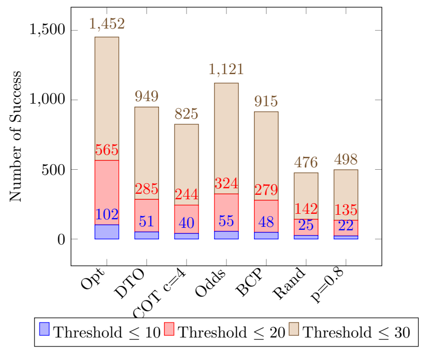


Figure 25: The number of successful offloadings for each model based on different threshold values.

6. Discussion

6.1. Deployment Environment

The proposed models can be integrated with the current offloading architectures and frameworks. For example, considering the existing work in offloading decision framework in the smart phones devices, in general, the main components for the offloading framework are decision engine or code offloader, network, edge servers and application profilers e.g. [20] [22] [26]. 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 [20]. 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 device. Our models can be implemented within the decision engine component, and it can be triggered whenever the output of the decision is to offload to an edge server. Most of the existing work in task offloading do not explicitly define the components of on-board computing unit in the smart vehicles but it is not difficult to have the same components mentioned above, including the profilers for network and the edge servers in the smart vehicles.

6.2. Computational Complexity

Regarding the time and space complexity of our models, in general, time complexity in the worst case is $\mathcal{O}(n)$. The mobile node is going to observe n servers if the condition for each model is met at the server number n. For the models DTO, COT and the Odds, there is one step before the observation which is the generation of the thresholds, i.e. a_k, $V*$ and s. We assume that this step is to be done once by the services provider outside the mobile node, but it is also not difficult to implement such step in the mobile node. For example, in the DTO model, we need $\mathcal{O}(n)$ if we calculate the threshold in the mobile node. This is also true for the Odds model. We require more time for calculating the threshold for the COT model, but this depends in the probability distribution. For example, in the uniform distribution, we only perform one operation as shown in [31]. In the normal probability distribution, we calculate and estimate the integration as shown in [31] with time complexity no more than $\mathcal{O}(n^2)$. For the space complexity, in the BCP model, we do not need extra space to store any data, thus, the space complexity is $\mathcal{O}(1)$. This is also true for the rest of the models if we assume that the training phase is done outside of the mobile node. In the case where we do the training phase locally in the mobile node, we only need to store the parameters of the probability distribution. When X is uniformly distributed, we need to store the maximum and the minimum values. When X is normally distributed, we need the mean and the standard deviation.

6.3. Local & Autonomous Decision Making

It is important to mention that we can achieve full independence for the models DTO, COT and the Odds. For example, in most of the existing offloading framework, e.g. [20] [22] [26], the decision engine is supported by a database where information about the offloading history can be stored. The challenge now is how to estimate the probability functions based on the stored data so we can apply the proposed models. Such estimation can be done using Kernel Density Estimation (KDE) [49]. KDE is a non-parametric method of estimating the probability density function of a continuous random variable. The term kernel refers to a special type of probability density function with the properties: nonnegative and integrates to one. One can estimate the density function $\hat{f}_X(x)$ of a random variable X in an incremental manner at observation $t+1$ using the following equation:

$$\hat{f}_{t+1}(x) = \frac{t}{t+1}\hat{f}_t(x) + \frac{1}{t+1}K_h(\frac{x - X_{t+1}}{h}). \quad (13)$$

Model	Performance	Applications
BCP [31]	Better than the p-model and the Random	Delay-Tolerant task offloading [13]
DTO [29, 30]	Near Optimal	Delay-Tolerant task offloading [13]
COT [31]	Near Optimal	Resource-intensive and delay constrained tasks
Odds Section (4.2)	Near Optimal	Data analytics task offloading [41, 42]

Table 5: Lessons to be drawn from this work.

where $K_h(\cdot)$ is a kernel function and $h > 0$ is a smoothing parameter called the bandwidth. Gaussian kernel function $K(x) = \frac{1}{\sqrt{2\pi}}e^{-0.5x^2}$ of width $h > 0$, is widely adopted in the KDE. A simple method for choosing the value of h is the Sliverman's rule-of-thumb width estimator which is $h \approx 1.06\sigma t^{-1/5}$ [50]. Once the probability distribution function is estimated, we then calculate the thresholds for each model based on the steps and the equations for each models.

6.4. Overall Performance and Application Domain/Use Cases

Table 5 gives a summary of the models in terms of their performance and examples for tasks to be offloaded for each model taken from the literature. From the results, we noticed that the more information we provide to the model the better results we get. This is clear in the DTO, the COT and the Odds models. In the BCP, however, the random variable X was higher, but we have to consider that we only feed the model with the number of observations. In other words, this model is fully independent, it is very light to be run in the mobile node and performs better than the Random and the p-mode.

In general, our proposed models can be utilized in applications where there is a deadline by which the task has to be finished. In the BCP and the DTO, we generally aim to minimize the random variable without defining further parameters. In the COT and Odds, on the other hand, there are parameters within the models that can be exploited to characterize the model based on the task to be offloaded. In the COT model, the parameter c can be used to define the demand of the task. As an example, we manage to select a server with less processing time (Section 5.1) and less CPU utilization (Section 5.2) by adjusting the value of c. Therefore, the COT can be used to manage offloading resource-intensive or delay constrained tasks. Moreover, the Odds model can be utilized in the data analytics task offloading where the mobile node collects and senses data with the aim of offloading them to an edge server for further analysis.

6.5. Model Limitations

First, the proposed models work in a single setting, i.e. each mobile node will run the models without taking into consideration other mobile nodes' context when offloading. In other words, each mobile device will offload based in the suggestion from the model implemented in that node. As a result, with large number of mobile nodes, there will be a high chance of offloading at the same time for different mobile nodes. Therefore, one might think about the competitive scenario, i.e., what happens if there is a high probability to have the same suggestions for multiple users. Another limitation is the situation where each server has different probability distribution function. In our experiment, and specifically in the simulation experiment, we assume that the MEC servers have the same probability distribution functions. However, it might be an interesting research direction if we could apply the OST based models when we have different probability distribution functions for the MEC servers.

7. Conclusions

In this paper, we concentrate on the application of OST approaches in the task offloading decision making in MEC environments. Derived by the literature and our previous work, we put forward a selection method to be used by the mobile nodes for the MEC servers when offloading data-oriented task considering the quality of the offloaded data. We provide a detailed assessment when applying and adopting the OST in quality-aware task offloading decision in MEC environments. Our experimental evaluations show that the OST based models perform better than the other offloading methods, efficient to be used in the mobile node and do not require a lot of resources. This work suggests that the mobile nodes should exploit (1) the mobility (2) the potential deployment of the MEC server at the edge of the network and (3) the deadline of the computational task to delay the offloading in the light of finding better resources. Additionally, the OST based model is suitable for situation where the mobile nodes need to make local and an independent decision within the environment of MEC. We think that it is not difficult to obtain the required information when making the decision with the aid of the MEC services providers. As future work, we plan to consider the case where mobile nodes are moving in different and advanced mobility patterns where recall of the observed MEC servers is allowed. We also plan to implement the OST models in real mobile nodes dealing with implemented MEC applications. We further aim to study the competitive scenario where many mobile nodes apply the same models at similar times.

Funding: 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.
Acknowledgment 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] G. Brown, Mobile edge computing use cases and deployment options, Juniper White Paper (2016) 1–10.

[2] J. Dolezal, Z. Becvar, T. Zeman, Performance evaluation of computation offloading from mobile device to the edge of mobile network, in: 2016 IEEE Conference on Standards for Communications and Networking (CSCN), IEEE, 2016, pp. 1–7. doi:10.1109/CSCN.2016.7785153.

[3] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, et al., Mec in 5g networks, ETSI white paper 28 (2018) 1–28. doi:www.etsi.org.

[4] Q.-V. Pham, F. Fang, V. N. Ha, M. Le, Z. Ding, L. B. Le, 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.

[5] J. Feng, Z. Liu, C. Wu, Y. Ji, Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling, IEEE vehicular technology magazine 14 (1) (2018) 28–36. doi:10.1109/MVT.2018.2879647.

[6] K. Zhang, Y. Mao, S. Leng, Y. He, Y. Zhang, Mobile-edge computing for vehicular networks: A promising network paradigm with predictive offloading, IEEE Vehicular Technology Magazine 12 (2) (2017) 36–44. doi:10.1109/MVT.2017.2668838.

[7] 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.

[8] R. A. Dziyauddin, D. Niyato, N. C. Luong, M. A. M. Izhar, M. Hadhari, S. Daud, Computation offloading and content caching delivery in vehicular edge computing: A survey, arXiv preprint arXiv:1912.07803.

[9] W. Tang, X. Zhao, W. Rafique, L. Qi, W. Dou, Q. Ni, An offloading method using decentralized p2p-enabled mobile edge servers in edge computing, Journal of Systems Architecture 94 (2019) 1–13. doi:https://doi.org/10.1016/j.sysarc.2019.02.001.

[10] C. N. Le Tan, C. Klein, E. Elmroth, Location-aware load prediction in edge data centers, in: 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), IEEE, 2017, pp. 25–31. doi:10.1109/FMEC.2017.7946403.

[11] Z. Zhou, H. Yu, C. Xu, Z. Chang, S. Mumtaz, J. Rodriguez, Begin: Big data enabled energy-efficient vehicular edge computing, IEEE Communications Magazine 56 (12) (2018) 82–89. doi:10.1109/MCOM.2018.1700910.

[12] T. Ouyang, X. Chen, L. Zeng, 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. doi:10.1109/RTSS46320.2019.00041.

[13] M. Li, P. Si, Y. Zhang, Delay-tolerant data traffic to software-defined vehicular networks with mobile edge computing in smart city, IEEE Transactions on Vehicular Technology 67 (10) (2018) 9073–9086. `doi:10.1109/TVT.2018.2865211`.

[14] S. Zhou, Y. Sun, Z. Jiang, Z. Niu, Exploiting moving intelligence: Delay-optimized computation offloading in vehicular fog networks, IEEE Communications Magazine 57 (5) (2019) 49–55. `doi:10.1109/MCOM.2019.1800230`.

[15] D. Huang, P. Wang, D. Niyato, A dynamic offloading algorithm for mobile computing, IEEE Transactions on Wireless Communications 11 (6) (2012) 1991–1995. `doi:10.1109/TWC.2012.041912.110912`.

[16] T. Ferguson, Optimal Stopping and Applications (2020).
URL `http://www.math.ucla.edu/~tom/Stopping/Contents.html`

[17] H. Ko, J. Lee, S. Pack, Spatial and temporal computation offloading decision algorithm in edge cloud-enabled heterogeneous networks, IEEE Access 6 (2018) 18920–18932. `doi:10.1109/ACCESS.2018.2818111`.

[18] M. H. ur Rehman, C. Sun, T. Y. Wah, A. Iqbal, 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. `doi:10.1109/MDM.2016.40`.

[19] S. Zhu, L. Gui, J. Chen, Q. Zhang, N. Zhang, Cooperative computation offloading for uavs: A joint radio and computing resource allocation approach, in: 2018 IEEE International Conference on Edge Computing (EDGE), 2018, pp. 74–79. `doi:10.1109/EDGE.2018.00017`.

[20] M. G. R. Alam, M. M. Hassan, M. Z. Uddin, A. Almogren, G. Fortino, Autonomic computation offloading in mobile edge for iot applications, Future Generation Computer Systems 90 (2019) 149–157. `doi:https://doi.org/10.1016/j.future.2018.07.050`.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, nature 518 (7540) (2015) 529–533. `doi:https://doi.org/10.1038/nature14236`.

[22] W. Junior, E. Oliveira, A. Santos, K. Dias, A context-sensitive offloading system using machine-learning classification algorithms for mobile cloud environment, Future Generation Computer Systems 90 (2019) 503–520. `doi:https://doi.org/10.1016/j.future.2018.08.026`.

[23] A. Ashok, P. Steenkiste, F. Bai, Vehicular cloud computing through dynamic computation offloading, Computer Communications 120 (2018) 125–137. doi:https://doi.org/10.1016/j.comcom.2017.12.011.

[24] B. Silva, W. Junior, 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. doi:https://doi.org/10.1007/978-3-030-19223-5_11.

[25] D. Sulaiman, A. Barker, Mamoc: Multisite adaptive offloading framework for mobile cloud applications, in: 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2017, pp. 17–24. doi:10.1109/CloudCom.2017.34.

[26] D. Sulaiman, 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. doi:10.1109/MobileCloud.2019.00017.

[27] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, G. Fortino, Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on sarsa, IEEE Access 8 (2020) 54074–54084. doi:10.1109/ACCESS.2020.2981434.

[28] B. Gu, Y. Chen, H. Liao, Z. Zhou, D. Zhang, A distributed and context-aware task assignment mechanism for collaborative mobile edge computing, Sensors 18 (8) (2018) 2423. doi:https://doi.org/10.3390/s18082423.

[29] I. Alghamdi, C. Anagnostopoulos, D. P. Pezaros, Time-optimized task offloading decision making in mobile edge computing, in: 2019 Wireless Days (WD), IEEE, 2019, pp. 1–8. doi:10.1109/WD.2019.8734210.

[30] I. A. I. Alghamdi, C. Anagnostopoulos, D. Pezaros, Delay-tolerant sequential decision making for task offloading in mobile edge computing environments, Informationdoi:https://doi.org/10.3390/info10100312.

[31] I. Alghamdi, C. Anagnostopoulos, D. P. Pezaros, On the optimality of task offloading in mobile edge computing environments, in: 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6. doi:10.1109/GLOBECOM38437.2019.9014081.

[32] F. T. Bruss, Sum the odds to one and stop, Annals of Probability (2000) 1384–1391.

[33] J. Zhang, K. B. Letaief, Mobile edge intelligence and computing for the internet of vehicles, Proceedings of the IEEE 108 (2) (2020) 246–261. doi:10.1109/JPROC.2019.2947490.

[34] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Communications Surveys & Tutorials 19 (3) (2017) 1628–1656. doi:10.1109/COMST.2017.2682318.

[35] J. Plachy, Z. Becvar, P. Mach, Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network, Computer Networks 108 (2016) 357–370. doi:https://doi.org/10.1016/j.comnet.2016.09.005.

[36] T. S. Ferguson, et al., Who solved the secretary problem?, Statistical science 4 (3) (1989) 282–289.

[37] J. Nyhoff, Algorithms to live by: The computer science of human decisions, Perspectives on Science and Christian Faith 69 (2) (2017) 127–129.

[38] N. Harth, C. Anagnostopoulos, D. Pezaros, Predictive intelligence to the edge: impact on edge analytics, Evolving Systems 9 (2) (2018) 95–118. doi:https://doi.org/10.1007/s12530-017-9210-z.

[39] N. Harth, C. Anagnostopoulos, Edge-centric efficient regression analytics, in: 2018 IEEE International Conference on Edge Computing (EDGE), IEEE, 2018, pp. 93–100. doi:10.1109/EDGE.2018.00020.

[40] C. Anagnostopoulos, K. Kolomvatsos, Predictive intelligence to the edge through approximate collaborative context reasoning, Applied Intelligence 48 (4) (2018) 966–991. doi:https://doi.org/10.1007/s10489-017-1032-y.

[41] M. Marjanović, A. Antonić, I. P. Žarko, Edge computing architecture for mobile crowdsensing, IEEE Access 6 (2018) 10662–10674. doi:10.1109/ACCESS.2018.2799707.

[42] L. Pu, X. Chen, G. Mao, Q. Xie, J. Xu, Chimera: An energy-efficient and deadline-aware hybrid edge computing framework for vehicular crowdsensing applications, IEEE Internet of Things Journal 6 (1) (2018) 84–99. doi:10.1109/JIOT.2018.2872436.

[43] C. Anagnostopoulos, Time-optimized contextual information forwarding in mobile sensor networks, Journal of Parallel and Distributed Computing 74 (5) (2014) 2317–2332. doi:https://doi.org/10.1016/j.jpdc.2014.01.008.

[44] F. T. Bruss, The art of a right decision: why decision makers may want to know the odds-algorithm, Newsletter-European Mathematical Society 62 (2006) 14–15.

[45] T. SimPy, Simpy: Discrete event simulation for python, Python package version 3 (9) (2017) 7. doi:https://simpy.readthedocs.io/en/latest/.

[46] J. V. Joseph, J. Kwak, G. Iosifidis, Dynamic computation offloading in mobile-edge-cloud computing systems, in: 2019 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2019, pp. 1–6. doi:10.1109/WCNC.2019.8885461.

[47] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, A. Rabuffi, CRAWDAD dataset roma/taxi (v. 2014-07-17), Downloaded from https://crawdad.org/roma/taxi/20140717 (Jul. 2014). doi:10.15783/C7QC7M.

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

[49] A. Zhou, Z. Cai, L. Wei, Density estimation over data stream.

[50] B. W. Silverman, Density estimation for statistics and data analysis, Vol. 26, CRC press, 1986.