entropy

Article

Multi-Task Offloading Based on Optimal Stopping Theory in Edge Computing Empowered Internet of Vehicles

Liting Mu [1], Bin Ge [1,*], Chenxing Xia [1,2] and Cai Wu [1]

[1] College of Computer Science and Engineering, Anhui University of Science and Technology, Huainan 232001, China; ltingmu@163.com (L.M.); cxxia@aust.edu.cn (C.X.); wuc17855370929@163.com (C.W.)
[2] Institute of Energy, Hefei Comprehensive National Science Center, Hefei 230031, China
* Correspondence: bge@aust.edu.cn

Abstract: Vehicular edge computing is a new computing paradigm. By introducing edge computing into the Internet of Vehicles (IoV), service providers are able to serve users with low-latency services, as edge computing deploys resources (e.g., computation, storage, and bandwidth) at the side close to the IoV users. When mobile nodes are moving and generating structured tasks, they can connect with the roadside units (RSUs) and then choose a proper time and several suitable Mobile Edge Computing (MEC) servers to offload the tasks. However, how to offload tasks in sequence efficiently is challenging. In response to this problem, in this paper, we propose a time-optimized, multi-task-offloading model adopting the principles of Optimal Stopping Theory (OST) with the objective of maximizing the probability of offloading to the optimal servers. When the server utilization is close to uniformly distributed, we propose another OST-based model with the objective of minimizing the total offloading delay. The proposed models are experimentally compared and evaluated with related OST models using simulated data sets and real data sets, and sensitivity analysis is performed. The results show that the proposed offloading models can be efficiently implemented in the mobile nodes and significantly reduce the total expected processing time of the tasks.

Keywords: mobile edge computing; computational offloading; optimal stopping theory; structured tasks; time optimization; sequential decision making

check for updates

Citation: Mu, L.; Ge, B.; Xia, C.; Wu, C. Multi-Task Offloading Based on Optimal Stopping Theory in Edge Computing Empowered Internet of Vehicles. *Entropy* **2022**, *24*, 814. https://doi.org/10.3390/e24060814

Academic Editor: Éloi Bossé

Received: 16 May 2022
Accepted: 9 June 2022
Published: 11 June 2022

1. Introduction

The Internet of Vehicles (IoV) is an emerging concept in intelligent transportation systems that aims to improve traffic safety and passenger comfort through integration with the Internet of Things (IoT), and it is an important implementation of the IoT [1]. Based on Vehicle to Everything (V2X) technology, the IoV connects vehicles, roadside units (RSUs), and service providers into a whole organic network, enabling all-round communication between them [2]. Smart vehicles in the IoV can communicate via V2X. Specifically, smart vehicles can share information with other vehicles through Vehicle to Vehicle (V2V) communication; thus, it can obtain a wider view of the road condition information shared by surrounding vehicles and greatly reduce the traffic accidents caused by blind spots [3].

As the number of vehicles on the road continues to increase, the IoV continues to evolve, smart vehicles accounting for an increasing share of Internet-connected devices. In the IoV paradigm, smart vehicles are equipped with computing units and communication technologies that provide services such as intelligent control, traffic management and interactive applications for vehicles. The edge-side computing architecture for autonomous vehicles (AVs) relies on the communication infrastructure and services provided by the edge-cloud collaboration and the (Long-Term Evolution/5th Generation Mobile Communication Technology) LTE/5G. The edge side mainly refers to on-board edge computing units, RSUs, Mobile Edge Computing (MEC) servers, etc. AVs are equipped with a large number of sensors that collect data for different types of traffic systems and navigation applications, etc. [4]. Currently, each AV is equipped with more than 60 to 100 electronic control units to support various functions such as communication, engine, dashboard, seat control, entertainment, etc. AVs generate a large amount of data in real time while driving; for example, AVs will generate and consume about 40 TB of data for every eight hours of driving (e.g., a city's High-Definition (HD) map is about 1.5TB) [5]. Although AVs have small-scale computational and storage resources, the computational capacity of vehicle terminals is relatively limited due to the huge computational tasks and the very high demand for real-time system response; e.g., in hazardous situations, the response time of the vehicle braking system is directly related to the safety of the vehicle, passengers, and the road, so they rely on other computational resources [6]. The MEC servers in the rear seat unit of AVs can play an important role in improving the performance of mobile vehicle terminals [7].

Since MEC servers operate at the edge of the radio access network, connecting to RSUs and performing transmission tasks with their help, their service area may be limited by the radio coverage of the RSUs. Due to the high mobility of the nodes, moving vehicles may pass through multiple RSUs and MEC servers during task offloading and can offload their computational tasks to any MEC server they have access to that provides computational resources for the tasks offloaded by the mobile nodes [8], as shown in Figure 1. The load on the MEC servers varies greatly from moment to moment; sometimes, there are a large number of users using the same server at the same time, causing a high load situation, while at other times, only a few users are connected to the server and the server load is low [9]. Therefore, the key to the problem is how the mobile node defines the offloading decision by which it selects the appropriate edge servers to offload the computational tasks so that the total computational latency is minimized [10].

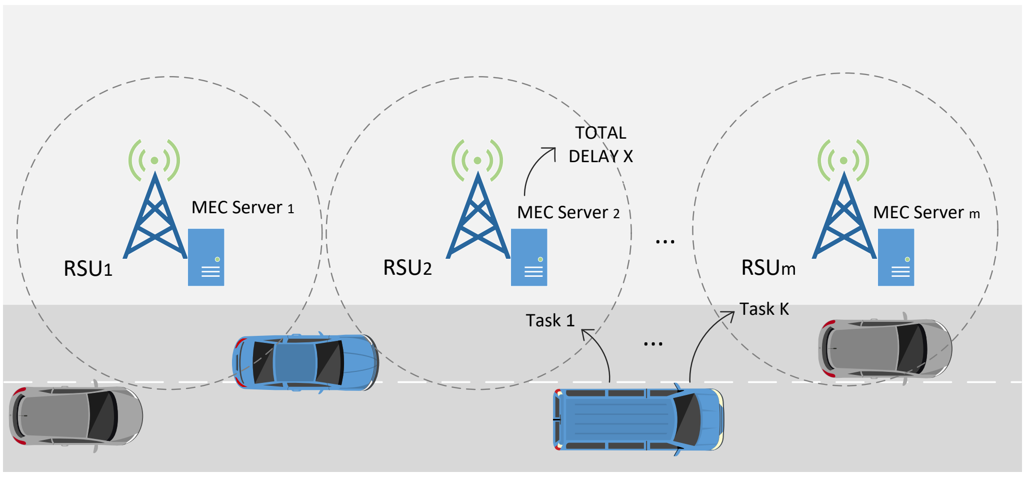


Figure 1. MEC in the Internet of Vehicles.

A centralized offloading strategy has been used in some studies, where vehicles can request information about MEC servers available for offloading along the road from a centralized server located at a higher level [11]. The centralized server is connected to the MEC servers through a wired network. However, this centralized architecture is not visible, and as the number of vehicles on the road continues to increase, the centralization of road information puts a higher load on the network [9]. Offloading strategies based on Vehicle-to-Vehicle communication (V2V) architectures are discussed in other studies, where vehicles can send tasks to the MEC server through other vehicles, each of which can act as an intermediate node for data delivery [4]. This solution requires the presence of other vehicles, but there is no guarantee that other vehicles will be present. Therefore, considering the scenario that if the mobile node only knows the MEC server that is being observed sequentially and has no global information about the candidate MEC server or only local information, how should it choose the optimal MEC server for offloading?

Such problems are known as Optimal Stopping Theory (OST) problems. A basic question about OST is that nodes autonomously decide when to act based on sequentially observed random variables with the goal of increasing expected benefits or decreasing expected costs [12]. The Secretary Problem (SP), the House Sale (HS) problem or the Fair Coin Problem are some well-known OST problems [12]. Single-node OST offloading problems have been studied, and in this paper, we attempt to study OST-based offloading problems for structured tasks with multiple subtasks and provide lightweight local algorithms that can be implemented in mobile nodes (vehicles or smartphones) so that mobile nodes can make offloading decisions independently.

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 task offloading models are described in Sections 4.1 and 4.2. Performance evaluation is provided in Section 5. Finally, Section 6 concludes the paper and outlines future research directions.

2. Related Work and Contributions

2.1. Related Work

In dealing with the issue of how to offload computational tasks and data to edge nodes, various pieces of research have emphasized whether data or tasks should be offloaded to servers or computed locally. The offloading decision has two main goals: minimization of computational delay and energy consumption.

Ref. [13] proposed the ST-CODA algorithm for computing offloading decisions based on space and time in this paper to design mobile devices' decisions in terms of time and location for offloading tasks by considering computing nodes and transmission costs in heterogeneous networks supported by edge clouds, where the time decision refers to postponing the offloading decision until a low-cost network, such as a wireless network, is found. Unlike ST-CODA, the approach studied in this paper will only offload tasks to edge servers and not further to the cloud, and the decision is deferred until a less loaded server is found.

Ref. [14] used Device-to-Device (D2D) communication technology to establish a direct link between mobile devices (MDs) and proposed a Secure and Energy-aware Collaborative Task Offloading for D2D communication (Sec2D) considering the data security. A new security model is constructed in terms of CPU cores, CPU frequency, and data size to measure the secure workload on heterogeneous MDs with guaranteed data security, a collaborative task offloading problem is proposed to minimize the time-averaged delay and energy consumption of MDs, and the Lyapunov optimization framework is applied to achieve online decision making. Greedy approach and optimal approach with different time complexity are proposed to deal with the generated mixed-integer linear programming (MILP) problem. The D2D technique mentioned in the paper is similar to the V2V communication architecture in the vehicular network, which can reduce the communication delay and improve the network capacity of the underlying wireless network.

In [15], OST was employed in order to obtain a good balance between the benefit of selecting the best edge device and the cumulative cost of deep resource detection. The authors tried to enhance the OST-based model by using a hierarchical learning mechanism to define the OST threshold and the sequence of edge nodes used for offloading. However, as it implements deep neural networks and deep Q-networks, it leads to a significant increase in the computational overhead and battery consumption of mobile nodes. Moreover, in their application, it is assumed that once the task is generated, the mobile node will have a list of edge devices, and then, the mobile node will define which edge node strikes a good balance between detection cost and execution delay of the task in advance.

In [16,17], the authors proposed a set of lightweight sequential decision models using the OST principle. In particular, Delay-Tolerant Offloading (DTO) decisions in the MEC environment are proposed, in which the unloading decision should be made within a predefined time horizon. This can be considered as a finite time horizon OST problem. This has been further investigated in [18], and two OST-based models have been introduced. The first one is the Best Choice Problem-based optimal task offloading policy (BCP), where the objective is to maximize the probability of offloading to the best edge server. The advantage of this model is that task offloading decisions can be made independently by mobile nodes, and it only needs to provide the number of observations (the number of edge nodes). Another one is the Cost-based Optimal Task Offloading Policy (COT), which enhances the BCP model by further considering the cost per observation of the server and obtaining more metric information about the performance. In [19], the authors summarized this series of OST-based offloading models, introduced the concept of contextual data quality awareness, proposed a timeliness function, injected offloading decisions, proposed an OST-based optimal selection method, and provided detailed evaluation and analysis by modeling and experimenting the application of OST models in MEC environments.

In [20], the authors considered the possible existence of the situation of multi-tasks and structured tasks offloading. A system model for a multi-user, multi-offload point and structured task environment is first developed to formalize the offload decision problem in this environment as a cost minimization problem, and then, a backtracking-based approach is designed to obtain its exact solution. To reduce the complexity, a method based on an improved genetic algorithm and a method based on a greedy strategy are designed. Finally, the three methods are verified and compared in terms of total cost and resource utilization at the unloading point and execution time for all users.

In this work, we consider the case of dividing the tructured tasks into multiple subtasks [20]. To solve the multi-task sequential offloading problem, inspired by the work of [19], we design distributed multi-task offloading model based on OST. The offloading models should be as lightweight as possible to facilitate local execution by mobile nodes. We also introduce the time-optimized function to ensure the timeliness of the offloading process. Additionally, we provide a comparative evaluation of all the OST-based models found in the literature with other offloading methods using simulation-based evaluation and real data set evaluation.

2.2. Contribution

In summary, our contributions are:

- We consider the multi-task offloading problem based on OST, where structured tasks are divided into multiple subtasks and offloaded to the MEC servers with the goal of minimizing the total task processing time.
- We propose a time-optimized function to simulate the situation where a mobile node needs to perform offloading as far as possible before the task data become obsolete in a realistic scenario and inject it into the decision model.
- We propose a time-optimized multi-task offloading model based on OST aiming at maximizing the probability of offloading to the optimal servers for general distribution scenarios.
- We propose a better time-optimized multi-task offloading model based on OST with the objective of minimizing the total task processing time for some realistic road scenarios where the CPU utilization may have an approximately uniform distribution.
- We provide an experimental comparison, performance evaluation and sensitivity analysis of the models in this paper and a series of models in other research using simulated and real data sets.

3. System Model and Problem Description

Consider a vehicle networking application scenario studied in [4,10]: mobile nodes travel on the road with RSUs and base stations deployed on the roadside, where a limited set of MEC servers with storage units and computation units are deployed [21]. Vehicles traveling on the road generate computational tasks and try to offload them to the MEC servers along the road. As user requirements become more complex, individual tasks gradually fail to meet user needs, and mobile nodes may generate a large number of structured tasks. For example, a structured task that is generated to meet business travel needs can be divided into three subtasks: weather forecasting (t_1), flight reservation (t_2), and hotel reservation (t_3). t_2 and t_3 are independent of each other and thus can be executed in parallel, but they all depend on the results provided by t_1 [22]. So, the three subtasks can be executed in the order of t_1, t_2, and t_3 for offloading, and thus, they can be offloaded in chunks by dividing the structured task into multiple subtasks [20].

The MEC server can operate at the edge of the network with the help of the RSU, so its communication range will be limited by the RSU communication range [5]. Since the input data for the vast majority of tasks are much larger than the output data of the computation results, to ensure the continuity of task completion, it can be assumed that there are mobility management entities in the server that implement mobility management algorithms, such as path selection, power control algorithms [23,24] or as the prediction model in [4]. If the node needs to obtain some results from a server other than the one selected for offloading this task, and the mobile node is outside the range of this MEC server, the selected MEC server should use a high-bandwidth wired connection to transmit the task results to the node through the next MEC server [25], thus solving the problem of continuity in the transmission of computational results after the vehicle is out of communication range of the MEC server.

The mobile node observes the MEC servers along the route in order at this point, and X_i is the random variable of the ith server processing time observed. When a node generates tasks locally and needs to offload, the node can observe n MEC servers; the mobile node uses the network and server analyzer to check each MEC server it passes and obtains the X values. Assuming that K blocks of tasks need to be unloaded, in n observation durations, then K unloading decisions need to be made.

The current goal is to optimize the decision to offload multiple targets to available servers. Overall, the offloading objectives are twofold: (1) maximize the probability of offloading to K optimal servers; (2) minimize the expected value of the total X for K tasks. We first propose an offloading model that is generalizable to the distribution of X for the first objective, and then, we propose a better offloading model with the second objective (minimizing the expected value of the total X for K tasks) for certain road scenarios where there may be a uniform distribution of X. Table 1 provides the key notations used in this paper.

Table 1. Key notations used in the paper.

Notation	Explanation
X	Random variable for server processing time
i	Random variable for server processing time
n	The serial number of the observed server
θ	Threshold in the offloading model
K	Number of tasks that need to offload
r_n	Stop node in the BCP model
P_n	Maximum probability in the BCP model
T_i	Time-optimized function
c_i	The i-th decision number that has been selected
$P_i(j_k)$	The probability of the i-th possible offloading branch in the K-Best model
α	Mandatory candidate in the K-Best model
β	Edge candidate in the K-Best model
j_k	Stop nodes in the K-Best model
$V_{k,k'}(n)$	Total time delay in KBU model
$t_(\rho,k)$	Optimal threshold after time optimization in KBU model
P	The probability P in the p-model

4. Computational Model

4.1. General Model

Consider first the OST problem model in the $K = 1$ case, where the goal is to maximize the probability of offloading to the best node, thus achieving the effect that the total processing time of the system is as small as possible. In the previous BCP model [18], the offloading scenario considered in this paper can be equated to the scenario discussed in this paper $K = 1$ where the mobile nodes know in advance the number of MEC servers that are available for offloading as $n(n > 0, n \geq K)$. The value of the specific n can be defined by the user or estimated by the application based on the deadlines of the tasks to be offloaded. Each node can only check the servers in a sequential and random order and will rank them relatively based on the previously observed servers. At each observation, the node should decide whether to select the current candidate server, and once selected, it cannot revoke its decision. The node should maximize the probability of selecting the best candidate among n probability of choosing the best among the candidates. This is a best choice problem [12]. The offloading rules in BCP are described as follows.

Call the first i server as the candidate server, the $i = 1, \ldots, n$, assuming that it is the best in terms of X_i. We define a positive integer r_n in $\{1, \ldots, n\}$, which is defined as:

$$r_n = min\left\{r \geq 1 : \frac{1}{r} + \frac{1}{r+1} + \ldots + \frac{1}{n-1} \leq 1\right\}. \tag{1}$$

For $n \geq 2$, according to BCP, the best strategy is to reject the first $r_n - 1$ server and then select the first candidate server (if any) to offload the task. The maximum probability of selecting the best candidate for BCP in (1) is given by the following equation:

$$P_n(r_n) = \frac{r_n - 1}{n} \sum_{i=r_n}^{n} \frac{1}{i-1}. \tag{2}$$

For smaller values of n, the optimal stop node r_n can be calculated using (1). When $n \to \infty$, we obtain the well-known secretary problem, where the optimal probability tends to $\frac{1}{e}$ [26]. Figure 2 shows the values of the probability of offloading to the best candidate and R_n for different values of n. It can be seen that as n increases, there is at least a 36% probability of offloading to the best node [19].

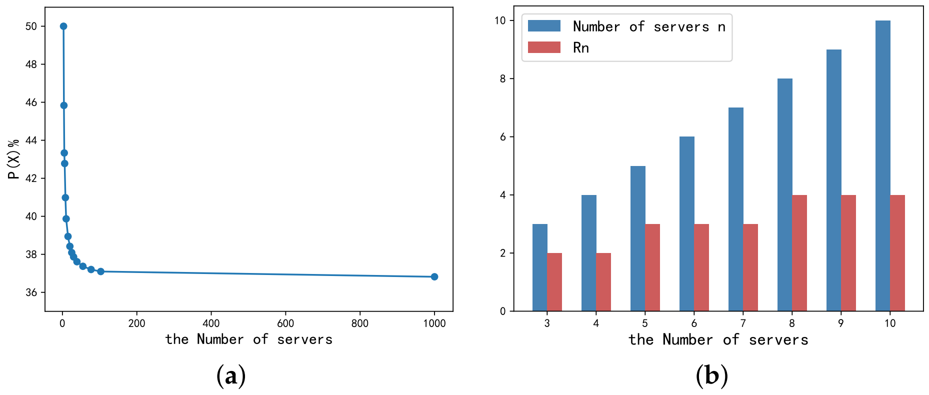


Figure 2. The probability of offloading to the best (**a**) and the value of R_n (**b**) for different numbers of servers n.

4.2. Time-Optimized Multi-Task Offloading Model

In this section, the situation that arises when a mobile node wants to offload a structured task into K subtasks to the MEC server and perform a data analysis task on the move is investigated. Data analysis tasks can be data correlation analysis, inference and predictive analysis [27], statistical learning model building, model selection [28,29] or data from HD maps, as shown in [21]. Data can be collected through different applications such as Mobile Crowd Sensing (MCS) or Vehicle Crowd Sensing (VCS) [30,31]. The mobile node sequentially observes the MEC server deployed in the RSU and decides whether to offload or not, and once rejected or selected, it cannot be reversed. The mobile node can perform a relative ranking of the historical observed X values, and within the observation time n, a total of K decisions are required to ensure that all K blocks of tasks are offloaded.

In addition, the tasks that need to be offloaded have contextual data time limits, so it is necessary to advance the offloading as much as possible to prevent data from being obsolete and not being processed. Using the observed server serial number i as the variable, where i can be interpreted as the time of the whole offload decision process, the $i = 1, 2, \ldots, n$. The time-optimized function T, adapted from [32], is constructed to consider the timeliness of the offloading decision:

- T is linear non-increasing in i.
- When $i = 1$, the observable time starts, the value of T is maximum, $T = 1$.
- When $i = n$, the observable time is over, the value of T is minimum.

The specific T function is defined as follows:

$$T_i = \begin{cases} \frac{n+1-i}{n}, & 1 \leq i \leq n. \\ 0, & i > n. \end{cases} \tag{3}$$

The following two multi-tasks offloading decisions are proposed for the structured task offloading problem:

1. For the general distribution scenarios of X, a time-optimized K-Best Choice (K-Best) strategy is proposed with the objective of maximizing the probability of selecting the best servers.
2. For some realistic road scenarios where the CPU utilization may have an approximately uniform distribution, a time-optimized K-Best selection based on Uniform distribution (KBU) strategy is proposed with the goal of minimizing the total processing time for K selections.

4.2.1. The K-Best Model

The offloading problem of such structured tasks can be abstracted as a K-best selection problem based on the OST; with the goal of maximizing the probability of selecting the best servers, the offloading algorithm is designed with the idea of K iterations. The description of K-Best model is given below: First, we define K stop nodes: $1 \leq j_1 < j_2 < \ldots < j_k \leq n$; here, c_i denotes the first i decision sequence number that has been selected. Then, we define two types of offload candidate servers, and the mobile node determines the type of candidate servers based on the relative ranking of the selected server utilization and the given threshold value θ. For example, if m offload selections have been made so far, the selected m servers are sorted in ascending order of utilization as: $1 < 2 < \ldots < m$. The candidate server types are defined follows:

1. Compulsory candidate server α: the current server utilization detected by the mobile node is X_i; if $X_i < \theta$ and the relative ranking $X_i \leq m$, then it is a compulsory candidate server α.
2. Marginal candidate server β: the current server utilization detected by the mobile node is X_i; if $X_i < \theta$ or the relative ranking $X_i \leq m + 1$, then it is a marginal candidate server β.

The Algorithm 1 is described as follows:

Algorithm 1 the K-Best algorithm.

```
Require: stop node $1 \leq j_1 < j_2 < \ldots < j_K \leq n$, random variables $X$, time optimization
   function $T_i$.
Ensure: total time delay.
   for $p = 1, 2, 3, \ldots, n$ do
      for $m = 1, 2, \ldots, K$ do
         if $m == 1$ then
            if $p \geq j_m$ and $p = \alpha$ or $\beta$ then
               perform offload;
               $j_m = p$;
            end if
         else if $m > 1$ and $p = \alpha$ and $p \in (j_m - 1, j_m)$ then
            perform offload;
            $j_m = p$;
         else if $m > 1$ and no $\alpha$ is in $(j_{m-1}, j_m)$ then
            if $p \geq j_m$ and $p = \alpha$ or $\beta$ then
               perform offload;
               $j_m = p$;
            end if
         end if
      end for
   end for
```

According to [33], it is difficult to give the general closed forms of the stopping nodes when $K > 1$, but the structure of the process of calculating the solution and the asymptotic form of the solution can be given.

With the consideration of time-optimized function T, taking $K = 2$ as an example, there are two possible decisions:

$c_1 < j_1 \leq c_2$ or $c_1 < c_2 \leq j_2$, the probability of the first case is:

$$\begin{aligned} P_1(j_2) &= \sum_{c_2=j_2}^{n} \prod_{k=c_1+1}^{j_2-1} T_k\left(1-\frac{1}{k}\right) \prod_{l=j_2}^{c_2-1} T_l\left(1-\frac{1}{l}\right)\frac{2}{c_2} \prod_{s=c_2+1}^{n} T_s\left(1-\frac{2}{s}\right) \\ &= \sum_{c_2=j_2}^{n} \prod_{i=c_1+1}^{n} T_i \frac{c_1}{n}\frac{2}{c_2-2}\frac{j_2-2}{n-1}. \end{aligned} \tag{4}$$

Let $P_1(j_2) = \frac{c_1(n-c_1)!}{n^{n-c_1}} F_1(c_1, j_2)$, the probability of the second case is:

$$\begin{aligned} P_2(j_2) &= \sum_{c_2=c_1+1}^{j_2-1} \prod_{k=c_1+1}^{c_2-1} T_k\left(1-\frac{1}{k}\right)\frac{1}{c_2} T_{c_2} \prod_{l=c_2+1}^{n} T_l\left(1-\frac{2}{l}\right) \\ &= \sum_{c_2=c_1}^{j_2-1} \prod_{i=c_1+1}^{n} T_i \frac{c_1}{n(n-1)}. \end{aligned} \tag{5}$$

Let $P_2(j_2) = \frac{c_1(n-c_1)!}{n^{(n-c_1)}} F_2(c_1, j_2)$, the total probability can be expressed as:

$$P(j_2) = P_1(j_2) + P_2(j_2). \tag{6}$$

When $n \to \infty$, we use the Euler–McLaughlin formula to replace the sum of probabilities; then, the total probability can be expressed as:

$$\begin{aligned} V_1(c_1, j_2) &= \frac{c_1(n-c_1)!}{n^{(n-c_1)}}\left(\int_{j_2}^{1} 2\frac{j_1}{c_2}dc_2 + \int_{c_1}^{j_2} 1dc_2\right) \\ &= \frac{c_1(n-c_1)!}{n^{(n-c_1)}}(-c_1 + j_2 - 2j_2 ln(j_2)). \end{aligned} \tag{7}$$

Since the function $V_1(c_1, j_2)$ is unimodal in j_2, the maximum P value is given when $\frac{dV_1}{dj_2} = 0$:

$$\frac{dV_1}{dj_2} = c_1(1 - 2lnj_2 - 2) = 0, \tag{8}$$

$$j_2^* = e^{-\frac{1}{2}} = 0.6065306596\ldots \tag{9}$$

Next, we find the value of j_1; again, there are two cases: $j_2 = j_2^*(if c_1 < j_2^*)$ or $j_2 = c_1(if c_1 \geq j_2^*)$; the probability of the first case is:

$$\begin{aligned} P_1(j_1) &= \sum_{c_1=j_1}^{j_2^*-1}\prod_{i=j_1}^{c_1-1} T_i\left(1-\frac{1}{i}\right)\frac{1}{c_1}c_2[F_1(c_1, j_2^*) + F_2(c_1, j_2^*)] \\ &= \sum_{c_1=j_1}^{j_2^*-1} \frac{\prod_{i=j_1}^{c_1-1} n+1-i}{n^{(c_1-j_1+2)}}\frac{j_1-1}{c_1-1}[F_1(c_1, j_2^*) + F_2(c_1, j_2^*)]. \end{aligned} \tag{10}$$

$P_1(j_1)$ is unimodal in j_1; the probability of the second case is:

$$\begin{aligned} P_2(j_1) &= \sum_{c_1=j_2^*}^{n-1}\sum_{c_2=c_1+1}^{n}\prod_{i=j_1}^{c_1-1} T_i\left(1-\frac{1}{i}\right)\frac{1}{c_1}T_{c_1}\prod_{l=c_1+1}^{c_2-1}\left(1-\frac{2}{l}\right)\frac{2}{c_2}T_{c_2}\prod_{s=c_2+1}^{n}\left(1-\frac{2}{s}\right) \\ &= \frac{(n+1-j_1)!}{n^{(n+1-j_1)}}\sum_{c_1=j_2^*}^{n-1}\sum_{c_2=c_1+1}^{n}\frac{2}{c_2-2}\frac{j_1-1}{n}\frac{1}{n-2}, \end{aligned} \tag{11}$$

$P_2(j_1)$ is monotonic in j_1; the total probability can be expressed as:

$$P(j_2) = P_1(j_2) + P_2(j_2). \tag{12}$$

Because $P_1(j_1)$ is unimodal in j_1, $P_2(j_1)$ is monotonic in j_1, so $P(j_1)$ is unimodal in j_1. Similarly, when $n \to \infty$, we use the Euler–Maclaurin formula to replace the sum of probabilities; then, the total probability can be expressed as:

$$V_2(j_1) = \int_{j_1}^{j_2^*}\frac{1}{c_1}V_1(c_1, j_2^*)dc_1 + j_1\int_{c_1=j_2^*}^{1}\int_{c_2=c_1}^{1}\frac{2}{c_2}dc_2dc_1. \tag{13}$$

Since $V_2(j_1)$ is unimodal in j_1, the maximum P value is given when $\frac{dV_2}{dj_1} = 0$:

$$j_1^* = e^{-\frac{1}{2}}W\left(-e^{(-3+e^{\frac{1}{2}})}\right) = 0.2291147286\ldots. \tag{14}$$

Here, $W(x)$ is the Lambert function. Based on a similar solution structure, we can derive in advance the form of the asymptotic solution for the stop nodes at $K = 3$ for the subsequent experiments:

$$\begin{aligned}
&\frac{j_1}{n} \to 0.1666171752\ldots. \\
&\frac{j_2}{n} \to -e^{-\frac{1}{3}} W\left(-exp\left(-\frac{2}{5} + e^{\frac{1}{3}}\right)\right) = 0.4369818602\ldots. \\
&\frac{j_3}{n} \to 0.7165313106\ldots.
\end{aligned} \tag{15}$$

The stop nodes can be calculated in advance and stored in the initialization parameters of the model without delaying the unloading. The space complexity of the K-Best algorithm is $S(n) = O(n)$ and the time complexity is $T(n) = O(n)$.

4.2.2. The KBU Model

In some realistic road scenarios, the distribution of utilization X of edge servers deployed in the RSU may be closer to a uniform distribution. In this case, the threshold corresponding to each observation node can be bounded in reverse by using the function of uniform distribution [34]. We propose the KBU model with the objective of minimizing the total K processing delays, and the thresholds are bounded by the time optimization function T_i at the same time.

In this model, nodes are required to select online K servers, and in order to tolerate a certain rate of offloading failures, we define a parameter $K^{'}$ that represents the expectation of the number of global offloads. $K^{'}$ does not have to be an integer, $K^{'} \geq K$. A set of metrics $I_1, I_2, \ldots, I_n$ is defined, which indicate whether the ith server is selected or not. $I_i = 0$ indicates that the ith server is not selected, and $I_i = 1$ shows the opposite, while δ_n denotes the set of all I_i indicators.

K and $K^{'}$ satisfy the following two constraints:

Constraint 1:

$$\sum_{i=1}^{n} I_i \geq K, K \in N, 0 \leq K \leq n. \tag{16}$$

Constraint 2:

$$E\left(\sum_{i=1}^{n} I_i\right) \geq K^{'}, K^{'} \in R, 0 \leq K^{'} \leq n. \tag{17}$$

The total time delay under different constraints is denoted by $V_{K,K'}(n)$; the objective of the strategy is to minimize the total time delay $V_{K,K'}(n)$:

$$V_{K,K'}(n) = \min_{(I_1,\ldots,I_n)\in\Gamma_n} E\left(\sum_{i=1}^{n} I_i X_i\right), n \geq K^{'} \geq K, r, n \in N. \tag{18}$$

$H_k = (X_1, I_1), (X_2, I_2) \ldots, (X_k, I_k)$ is the offload history, there are N_k tasks that have been offloaded, $0 \leq N - k \leq K$; then, there is the structure of the basic solution as:

$$V_{K,K'}(n) = V_{K-N_k,K'-N_k}(n-k) + \sum_{j=1}^{k} I_j X_j. \tag{19}$$

Let δ be the minimum time to satisfy constraint 1, so that:

$$V_{K,K'}(n) = V_{0,K'-K}(n-\delta) + \sum_{j=1}^{\delta} I_j X_j. \tag{20}$$

Among them:

$$V_{0,K'-K}(i) = \frac{\left(K' - K\right)^2}{2i}. \tag{21}$$

Now, we can get the double recursive solution form of $V_{K,K'}(n)$:

$$V_{K,K'}(n) = V_{K,K'}(n-1) - \frac{1}{2}\left[V_{K,K'}(n-1) - V_{K-1,K'-1}(n-1)\right]^2. \tag{22}$$

The optimal threshold $t^*(\rho, k)$ denotes the threshold value for the kth server detected in the offload decision for the ρth task. $t^*(\rho, k)$ can be given by the inverse of the structure of the optimal total time cost $V_{K,K'}(n)$:

$$\begin{aligned} t^*(\rho, k) &= t^*_{K,K'}(\rho, k) \\ &= V_{K-\rho,K'-\rho}(k-1) - V_{K-\rho-1,K'-\rho-1}(k-1). \end{aligned} \tag{23}$$

That is, for the first ρth task, if the currently observed $X_i \leq t^*(\rho, n-k)$, the server i should be taken into account.

Considering that offloading should be performed as far as possible before the task data become obsolete, the time-optimized function is further used to constrain the threshold to obtain a new threshold defined as $t(\rho, k)$:

$$t(\rho, k) = t^*(\rho, k) * T_{(n-k)}. \tag{24}$$

For example, when $K = 3$, $n = 10$, the KBU model gives different thresholds for the three subtasks, and a comparison of $t(\rho, k)$ and $t^*(\rho, k)$ before and after optimization is shown in Figure 3.

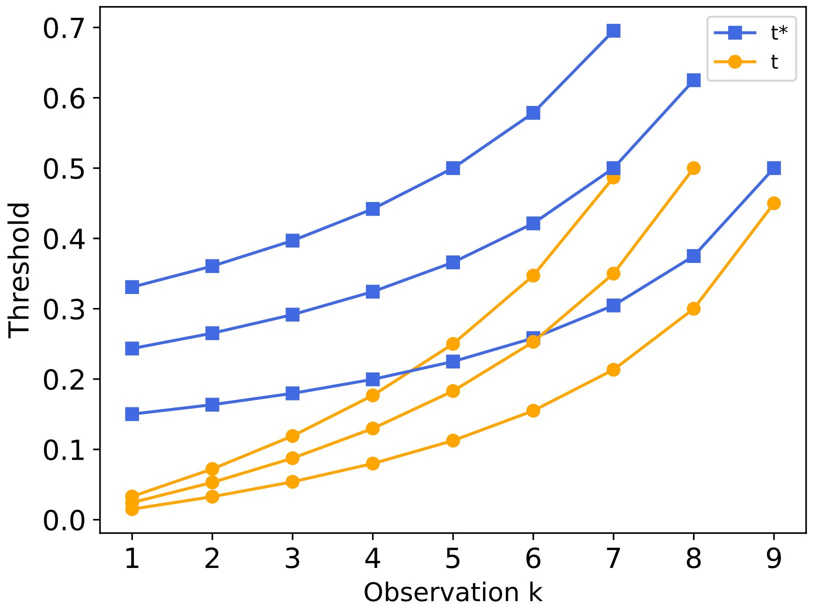


Figure 3. Comparison of the threshold t before and after the time-optimized function constraint.

The Algorithm 2 is given as follows:

Algorithm 2 the KBU algorithm.

```
Require: distribution of random variable $X$, time-optimized function $T_i$.
Ensure: total time delay $V_{K,K'}(n)$.
  A Obtain the minimum time delay:
  (A1)
  for $i = \lceil K' \rceil - K, \lceil K' \rceil - K + 1, \ldots, n$ do
      $V_{0,K'-K}(i) = (K' - K)^2/(2i)$;
  end for
  (A2)
  for $i = \lceil K' \rceil - K, \lceil K' \rceil - K + 1, \ldots, n - k; s = 1, 2, \ldots, k$ do
      $V_{s,i}(i) = i/2$;
  end for
  (A3)
  for $s = 1, 2, \ldots, K$ and the initial conditions (A1) and (A2) do
      $V_{s,K'-K+s}(i) = K' - K + s, \ldots, n - K + 1, n - K, \ldots, n$;
  end for
  B Obtain the optimal threshold:
  (B1)
  for $i = \lceil K' \rceil - K, \lceil K' \rceil - K + 1, \ldots, n + K$ do
      $t(K, i) = V_{0,K'-K}(i) = (K' - K)^2/(2i)$;
  end for
  (B2)
  for $\rho = 0, 1, \ldots, K - 1$ do
      for $i = K - \rho, K - \rho + 1, \ldots, n$ do
          $t^*(\rho, i) = V_{K-\rho,K'-\rho}(i-1) - V_{K-\rho-1,K'-\rho-1}(i-1)$;
      end for
  end for
  $t(\rho, k) = V^*(\rho, k)V_{n-k}$;
  for $p = 1, 2, 3, \ldots, n$ do
      for $m = 1, 2, 3, \ldots, K - 1$ do
          if number of nodes left $\leq$ number of unloaded tasks left then
              perform offload;
          else if $X_p < t(m, n - p)$ then
              perform offload;
          end if
      end for
  end for
```

Same as the K-Best model, the K thresholds here can be calculated in advance and stored in the initialization parameters of the model without delaying the unloading. The space complexity of the KBU algorithm is $S(n) = O(n)$, and the time complexity is $T(n) = O(n)$.

5. Experimental Evaluation

We use two settings to evaluate the two proposed OST-based multi-task offloading models: simulation-based evaluation and real-world data sets based. In both settings, we compare our models: namely, K-Best (Section 4.2.1) and KBU (Section 4.2.2) with the Optimal model [19], BCP model [17], Random selection model (Random) and the p-stochastic model (p-model). We will discuss these models next.

The four offloading models used for comparison in a multitask offloading scenario are described as follows:

The optimal model takes the number of observations n, the probability distribution function of X, the quality-aware function and the threshold θ as inputs, and outputs the index s where nodes start checking the MEC servers. If the subsequent observed X does not exceed θ, i.e., $X \leq \theta$, then we perform offloading; otherwise, we continue to observe. When the number of remaining unselected servers m does not exceed the number of remaining unloaded tasks, i.e., $m \leq K - K_p$, then the remaining tasks are offloaded to the corresponding servers sequentially without comparing so that all tasks are guaranteed to be offloaded.

The BCP model takes the number of observations n and the probability distribution function of X as inputs and calculates the stop node r_n, rejects the first $r_n - 1$ servers, and then selects the first server with the best relative previous ranking (if any) to offload the task. When there are K_p tasks that have been offloaded, there are still $K - K_p$ tasks left to be offloaded. If the number of remaining unselected servers m does not exceed the number of remaining unloaded tasks, i.e., $m \leq K - K_p$, then the remaining tasks are offloaded to the corresponding servers in turn to ensure that all tasks can be offloaded.

In the p-model model, we use $p = 0.8$ as the offload probability for each server in the following experiments, i.e., the node connects to the MEC server during the move and has an 80% probability of offloading to the server it is currently observing. In the random model, the mobile node randomly selects K servers for task offloading, and again, when the number of remaining unselected servers m does not exceed the number of remaining unloaded tasks, i.e., $m \leq K - K_p$, then the remaining tasks are offloaded to the corresponding servers sequentially without further observation, so that all tasks are guaranteed to be offloaded.

To simulate the MEC environment, we use Simpy [35] in Python. Simpy is a process-based framework for discrete-event simulation. Each MEC server is modeled as a resource, and during the simulation, the server publishes its processing time X, and the connected mobile node can receive the information. The mobile node is modeled as a process that detects the processing time of the server in a one-direction mobile model and chooses whether to offload it or not. The values of the simulation experiment are shown in Table 2.

Table 2. Simulation experiment parameters' values.

Parameters	**Value/Rang**
X	N (50,10), U (0,1)
Number of mobile nodes	1000
K	3
n	10
Threshold θ of K-Best	48
Threshold θ of Optimal	50
P for the p-modle	0.8

5.1. *Evaluation Based on Simulated Data*

In the evaluation based on simulated data, the node would know in advance the distribution function of the random variable X, i.e., server utilization or CPU utilization. We consider experiments in the case of $K = 3$, $n = 10$. A series of random numbers will be generated by Python functions, which obey a specific distribution, such as normal or uniform distribution. Consider first the case where X follows a normal distribution.

As shown in Figure 4, the results between the K-Best model and the four comparison models are compared when the distribution of X presents a case of normal distribution. The K-Best model clearly performs best in terms of total delay of offloading. We can observe a clear overlap between the K-best model and Optimal model in Figure 4a, and this overlap is significantly reduced in the comparison plots with the other three models. It can be seen that the K-Best model works significantly better than the last three models and presents Optimal. Overall, the OST-based models (K-Best, Optimal and BCP) are significantly more effective than several other models that are not based on OST. It can be seen in Figure 5 that the OST-based models achieve a lower expected total processing delay, and the K-Best model has similar results to the Optimal and BCP models, while it differs more from the *p*-model and random model.

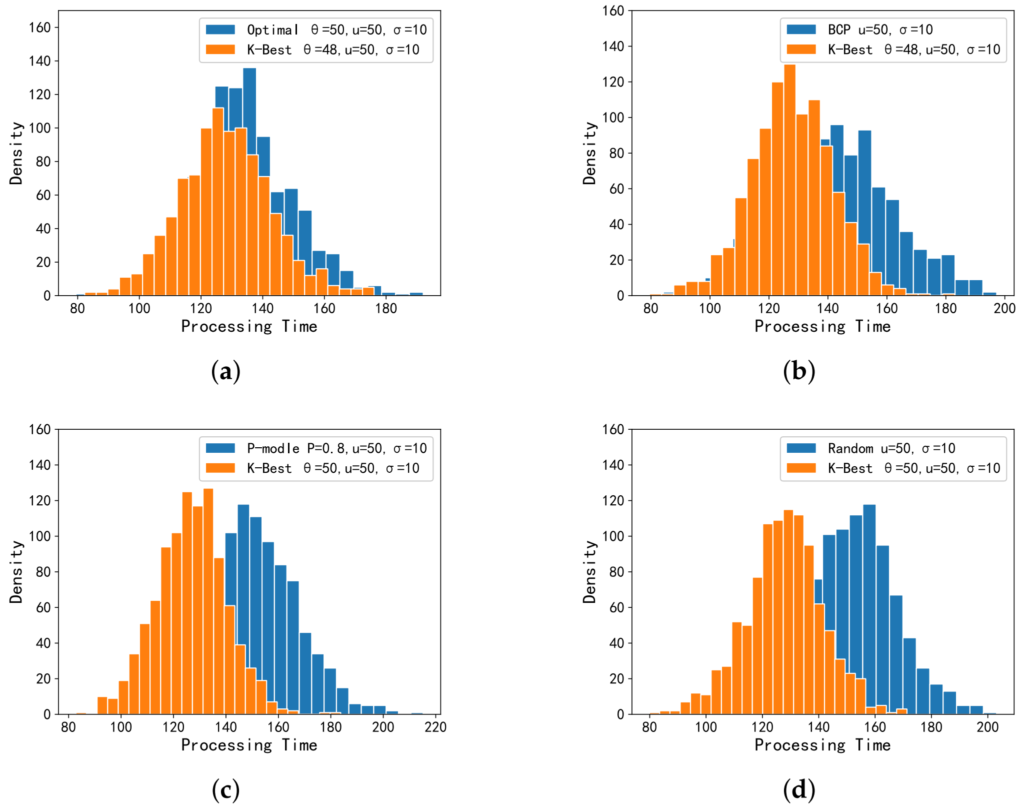


Figure 4. Simulation results for all the models when X is normally distributed. (**a**) Optimal and K-Best selections. (**b**) BCP and K-Best selections. (**c**) P-model and K-Best selections. (**d**) Random and K-Best selections.

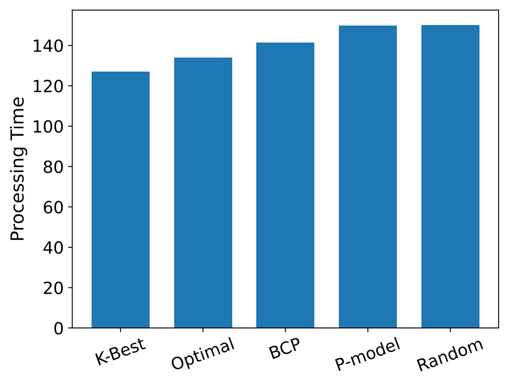


Figure 5. Confidence interval of simulation results for each model when X is normally distributed.

In the previous experiments, the random variable X observed by the mobile nodes followed a normal distribution. Now, let X be uniformly distributed scaled in [0,1]; X is the server utilization or CPU utilization. For example, $X = 0.5$ means that the CPU utilization of the server is 50%. For all models, the following experiments follow the same steps as in the previous experiments.

As shown in Figure 6, the KBU model clearly performs the best compared to the remaining four models, and it can be seen in Figure 6a that the KBU model overlaps significantly with the K-Best and Optimal models. Overall, the four OST-based models perform better than the other models. The KBU model achieves the smallest expected processing delay with X following a uniform distribution, and the K-Best model also achieves a more desirable expected processing delay, which is significantly better than the p-model and the Random model, as can be seen in Figure 7.

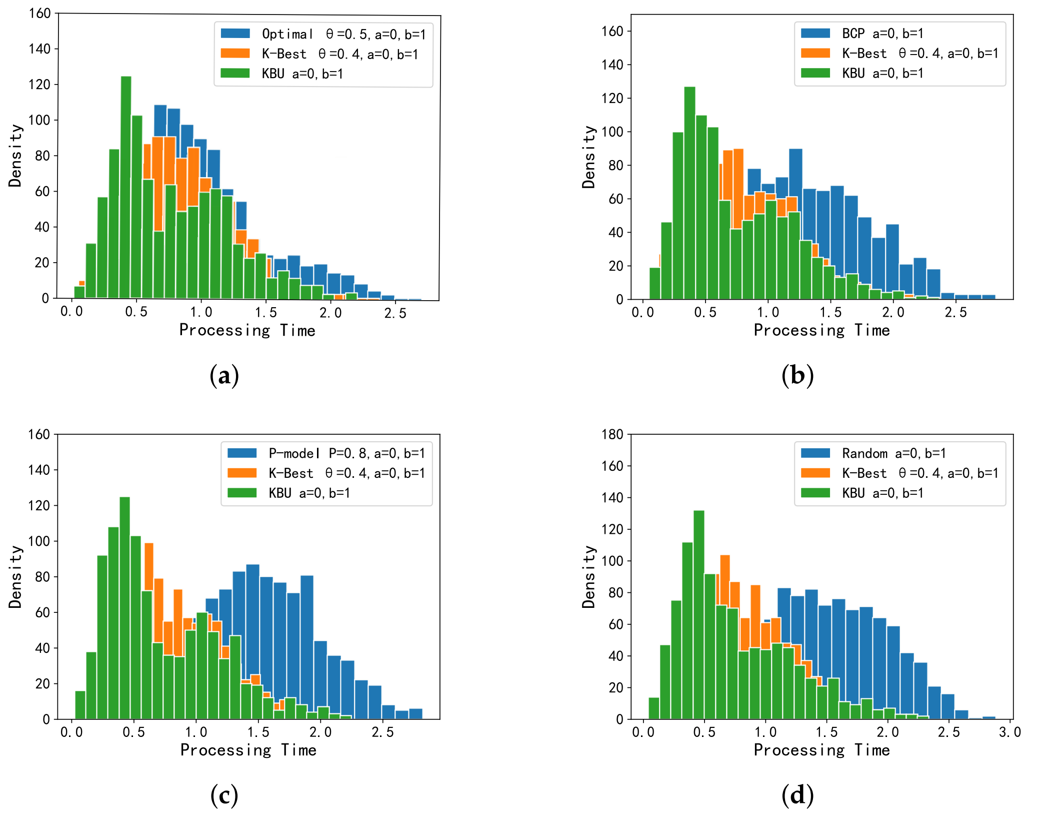


Figure 6. Simulation results for all the models when X is uniformly distributed. (**a**) Optimal, K-Best and KBU selections. (**b**) BCP, K-Best and KBU selections. (**c**) P-model, K-Best and KBU selections. (**d**) Random, K-Best and KBU selections.

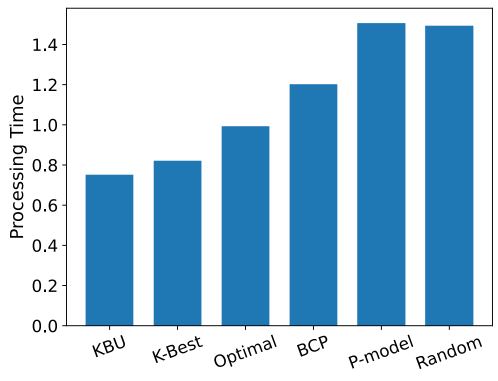
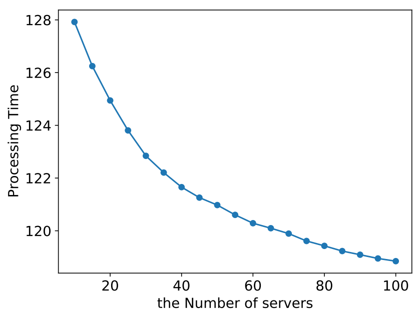


Figure 7. Confidence interval of simulation results for each model when X is uniformly distributed.

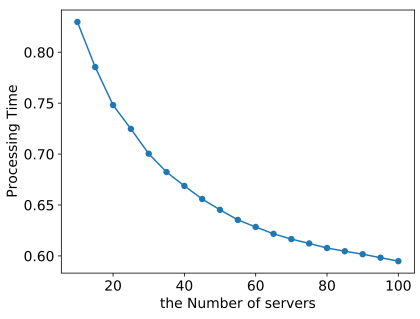
Sensitivity Analysis

As shown in Figure 8, the total processing delay of the K-Best model tends to decrease as the number of servers increases with a constant number of tasks $K = 3$. Figure 8a shows the case when the random variable X obeys a normal distribution, and the case when the random variable X obeys a uniform distribution is shown in Figure 8b. We can see that when the number of observable servers increases, the K-Best model allows mobile nodes to select the best possible server for task offloading and has general applicability to the distribution of X.

As shown in Figure 9, as the number of servers increases, the total processing latency of the KBU model tends to decrease with a constant number of tasks $K = 3$. It means that the KBU model enables mobile nodes to select the best possible server for task offloading as the number of observable servers increases.



(**a**)



(**b**)

Figure 8. Total latency of the K-Best model varies with the number of servers n. (**a**) X normally dstributed. (**b**) X uniformly distributed.

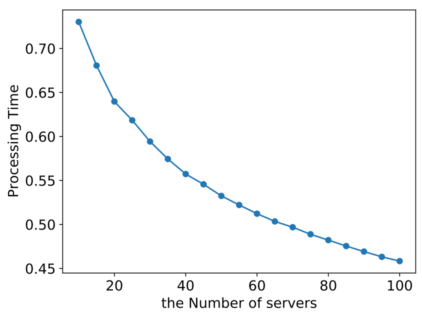


Figure 9. Total latency of the KBU model varies with the number of servers n.

5.2. Evaluation Based on Real Data

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. We use the CABS data set provided by the Shenzhen Smart City project [36] to simulate the movements of the mobile nodes. The data set contains four attributes of the vehicles: vehicle ID, GPS location, movement time and movement speed. The use of mobility trace here is not for studying the mobility of users; in our experiment, for each movement, the car picks a server from the servers' data set, checks that server utilization, and makes a decision of whether the car should offload at that time or continue observing based on the decision suggested by the models, as explained earlier in the Simulation Evaluation section. An example of one connection observation is shown in Table 3; we can see that Cab 178 made connections to three servers at different times, where two connections were made to server m_1938 at almost the same location but at different times, while obtaining server information, i.e., CPU utilization.

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

Taxi ID	Movement ID	Location	Machine Name	CPU Utilization
178	2014/10/22 8:00:20	(22.5965, 114.114601)	m_1935	(23)
178	2014/10/22 8:00:50	(22.5966, 114.1182 02)	m_1937	(35)
178	2014/10/22 8:01:20	(22.5968, 114.1194)	m_1938	(54)
178	2014/10/22 8:01:50	(22.5965, 114.119598)	m_1938	(26)

The processing time of the servers is provided by the actual CPU utilization obtained in the Alibaba Cluster Tracking Program [37]. There are more than one billion rows of CPU utilization data from about 150 servers that are recorded in the data set. One million of these data are used in the experiment; Figure 10 shows the probability distribution of the CPU utilization of all servers in the data set. We can see in Figure 10 that the CPU utilization follows a normal distribution with $\mu = 36$, $\sigma = 16$. The values of the key parameters' values in the experiment are given in Table 4.

Table 4. Real data set experiment parameters' values.

Taxi ID	Movement ID
X	Real servers CPU utilization in $N(36, 16)$
Number of mobile nodes	1000
K	3
n	10
Threshold θ of K-Best θ	48
Threshold θ of Optimal θ	50
P for the p-modle	0.8

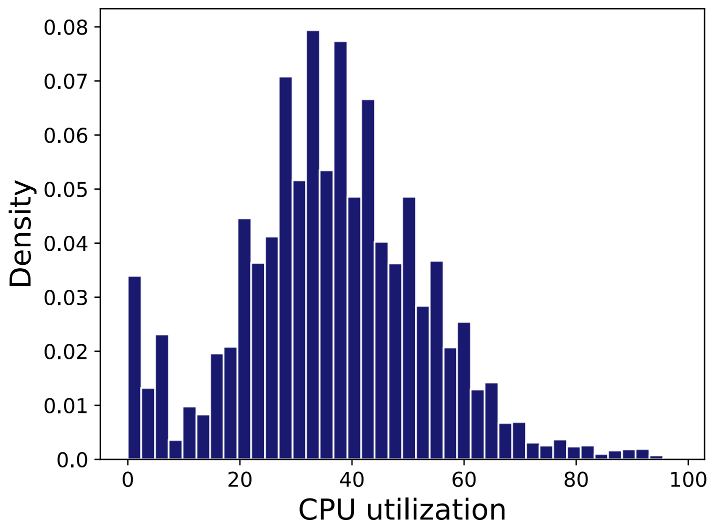


Figure 10. The distribution of the servers' CPU utilization.

At the beginning of the experiment, the mean and the standard deviation were taken once for the whole servers' utilization data set to feed the models (K-Best, KBU, Optimal and BCP). In a realistic scenario, the mobile nodes do not know the mean and standard deviation of a particular MEC server, but they can obtain the information about the historical data of the MEC servers in one area in a specific time with the help of MEC servers operators. Therefore, we take this information once when we start the experiment.

Similar to the experiments based on simulated data, the results of each model are aggregated and compared in the experiments based on real data sets. Each model selects K servers for mobile nodes to offload to minimize the total offload delay. The total delay is expressed in terms of the total server utilization.

Figure 11 shows the average server utilization for the offloading decisions suggested by each model. We can see that in the results based on real data sets, the OST-based model still performs better at minimizing the total offloading delay, and the KBU model and the K-Best model perform the best. Since in real data sets, the distribution of CPU utilization is closer to a normal distribution than a uniform distribution, the KBU model fails to show a more significant advantage over the K-Best model.

Figure 12 shows the average waiting time for the offloading decision suggested by each model. We can see that the p-model with $p = 0.8$ suggests the smallest average waiting time. However, the average offloading time is too long, so that choosing an instant server is not a wise offload strategy. Moreover, the optimal server is unknown, and the mobile node cannot know exactly which server is the best choice, so using the OST-based model can achieve a near-optimal server utilization.

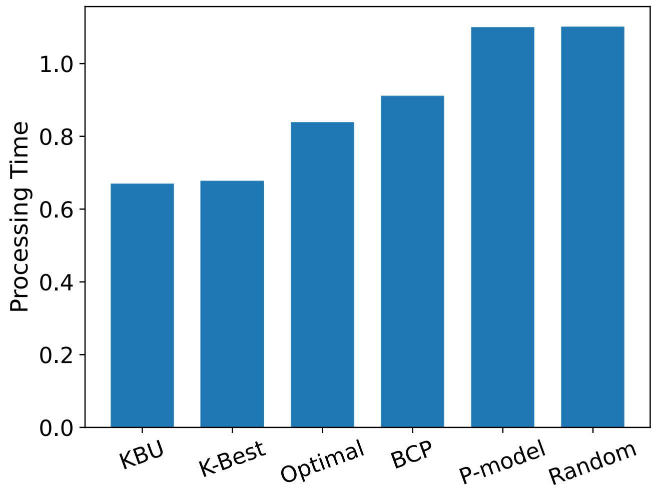


Figure 11. Average offloading time.

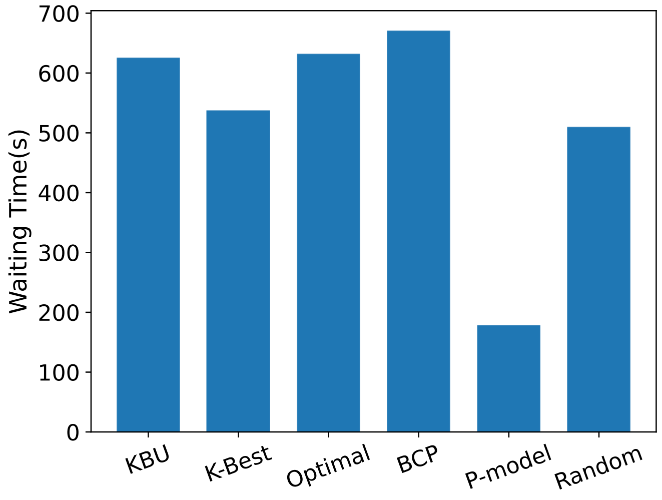


Figure 12. Average waiting time.

Sensitivity Analysis

We use the number of successful offloading for different threshold requirements as a performance metric for the models. The number of successful offloading is the number of offload decisions suggested by each model that satisfy a specific requirement. Suppose we have three different MEC applications x, y, and z, all with specific requirements. For example, application x requires a total CPU utilization ≤ 0.4, application y requires a total CPU utilization ≤ 0.6, and application z requires a total CPU utilization ≤ 0.8.

Figure 13 shows the number of successful offloading for all models for different requirements. For the first case requiring a total CPU utilization ≤ 0.4, the KBU model achieves 89 successful offloads, and the K-Best model achieves 90 successful offloads; for the second case where the required total CPU utilization ≤ 0.6, the KBU model achieves 250 successful offloads and the K-Best model achieves 230 successful offloads; for the third case that the total CPU utilization should ≤ 0.4, the KBU model achieves 385 successful offloads, and the K-Best model achieves 408 successful offloads. Overall, we can see that the OST-based offloading models (KBU, K-Best and Optimal) are significantly more effective than the other models.

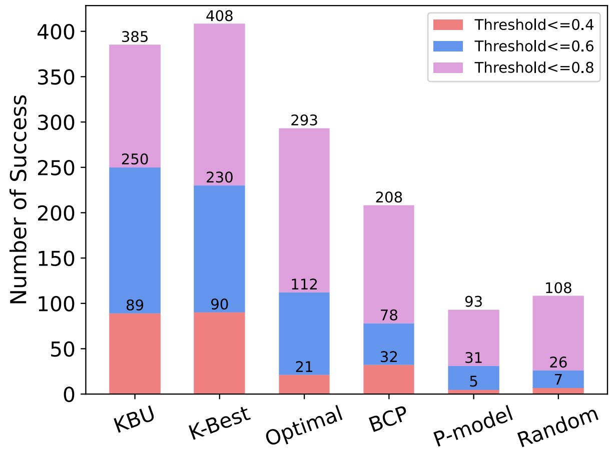


Figure 13. The number of successful offloadings for each model based on different threshold values.

6. Conclusions

In this paper, we focus on a time-optimized multi-tasking offloading model based on OST in an IoV environment. A time-optimized multi-task offloading model adapted to a general X distribution and a time-optimized multi-task offloading model that performs better for the case of uniform X distribution are proposed, and detailed evaluations are provided. The experimental evaluations show that the OST-based models outperform other offloading methods, achieve more desirable expected processing latencies, are efficient to use in mobile nodes, and do not require significant resources. In addition, the OST-based models are suitable for scenarios where mobile nodes need to make local and independent decisions in the MEC environment. In realistic scenarios, mobile nodes can obtain the required information with the help of MEC service providers and thus can make decisions autonomously. As future work, it is considered to study the K-Best model when K keeps increasing and may give a feasible closed-form solution. Since the observed recall of the MEC server is allowed in different mobility models, the optimization of the mult-itasking offloading model based on OST theory can be further investigated in the recallable server as well as in the multi-node competition scenario.

Author Contributions: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation and visualization, L.M.; writing—review and editing, B.G. and C.X.; supervision, C.W.; project administration and funding acquisition, B.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key R&D Program of China (2020YFB1314103); the National Natural Science Foundation of China (6210071479); and the Natural Science Foundation of Anhui Province (2108085QF258).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

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

References

1. Liyanage, M.; Porambage, P.; Ding, A.Y.; Kalla, A. Driving forces for Multi-Access Edge Computing (MEC) IoT integration in 5G. *ICT Express* **2021**, *7*, 127–137. [CrossRef]
2. Kekki, S.; Featherstone, W.; Fang, Y.; Kuure, P.; Li, A.; Ranjan, A.; Purkayastha, D.; Jiangping, F.; Frydman, D.; Verin, G.; et al. MEC in 5G networks. *ETSI White Pap.* **2018**, *28*, 1–28.
3. Pham, Q.V.; Fang, F.; Ha, V.N.; Piran, M.J.; Le, M.; Le, L.B.; Hwang, W.J.; Ding, Z. A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art. *IEEE Access* **2020**, *8*, 116974–117017. [CrossRef]

4. Zhang, K.; Mao, Y.; Leng, S.; He, Y.; Zhang, Y. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading. *IEEE Veh. Technol. Mag.* **2017**, *12*, 36–44. [CrossRef]
5. Sabella, D.; Moustafa, H.; Kuure, P.; Kekki, S.; Zhou, Z.; Li, A.; Thein, C.; Fischer, E.; Vukovic, I.; Cardillo, J.; et al. *Toward Fully Connected Vehicles: Edge Computing for Advanced Automotive Communications*; 5GAA: Munich, Germany, 2017.
6. Dziyauddin, R.A.; Niyato, D.; Luong, N.C.; Izhar, M.A.M.; Hadhari, M.; Daud, S. Computation offloading and content caching delivery in vehicular edge computing: A survey. *arXiv* **2019**, arXiv:1912.07803.
7. Feng, J.; Liu, Z.; Wu, C.; Ji, Y. Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling. *IEEE Veh. Technol. Mag.* **2018**, *14*, 28–36. [CrossRef]
8. Gao, H.; Huang, W.; Yang, X. Applying probabilistic model checking to path planning in an intelligent transportation system using mobility trajectories and their statistical data. *Intell. Automat. Soft Comput.* **2019**, *25*, 547–559. [CrossRef]
9. Deng, S.; Huang, L.; Taheri, J.; Zomaya, A.Y. Computation offloading for service workflow in mobile cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *26*, 3317–3329. [CrossRef]
10. Tang, W.; Zhao, X.; Rafique, W.; Qi, L.; Dou, W.; Ni, Q. An offloading method using decentralized P2P-enabled mobile edge servers in edge computing. *J. Syst. Archit.* **2019**, *94*, 1–13. [CrossRef]
11. Li, M.; Si, P.; Zhang, Y. Delay-tolerant data traffic to software-defined vehicular networks with mobile edge computing in smart city. *IEEE Trans. Veh. Technol.* **2018**, *67*, 9073–9086. [CrossRef]
12. Ferguson, T. *Optimal Stopping and Applications*; UCLA Department of Mathematics: Los Angeles, CA, USA, 2020.
13. Ko, H.; Lee, J.; Pack, S. Spatial and temporal computation offloading decision algorithm in edge cloud-enabled heterogeneous networks. *IEEE Access* **2018**, *6*, 18920–18932. [CrossRef]
14. Li, Z.; Hu, H.; Hu, H.; Huang, B.; Ge, J.; Chang, V. Security and Energy-aware Collaborative Task Offloading in D2D communication. *Future Gener. Comput. Syst.* **2021**, *118*, 358–373. [CrossRef]
15. Ouyang, T.; Chen, X.; Zeng, L.; Zhou, Z. Cost-aware edge resource probing for infrastructure-free edge computing: From optimal stopping to layered learning. In Proceedings of the 2019 IEEE Real-Time Systems Symposium (RTSS), Hong Kong, China, 3–6 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 380–391.
16. Alghamdi, I.; Anagnostopoulos, C.; Pezaros, D.P. Time-optimized task offloading decision making in mobile edge computing. In Proceedings of the 2019 Wireless Days (WD), Manchester, UK, 24–26 April 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–8.
17. Alghamdi, I.; Anagnostopoulos, C.; P Pezaros, D. Delay-tolerant sequential decision making for task offloading in mobile edge computing environments. *Information* **2019**, *10*, 312. [CrossRef]
18. Alghamdi, I.; Anagnostopoulos, C.; Pezaros, D.P. On the optimality of task offloading in mobile edge computing environments. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Big Island, HI, USA, 9–13 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
19. Alghamdi, I.; Anagnostopoulos, C.; Pezaros, D.P. Data quality-aware task offloading in mobile edge computing: An optimal stopping theory approach. *Future Gener. Comput. Syst.* **2021**, *117*, 462–479. [CrossRef]
20. Kuang, L.; Gong, T.; OuYang, S.; Gao, H.; Deng, S. Offloading decision methods for multiple users with structured tasks in edge computing for smart cities. *Future Gener. Comput. Syst.* **2020**, *105*, 717–729. [CrossRef]
21. Zhang, J.; Letaief, K.B. Mobile edge intelligence and computing for the internet of vehicles. *Proc. IEEE* **2019**, *108*, 246–261. [CrossRef]
22. Le Tan, C.N.; Klein, C.; Elmroth, E. Location-aware load prediction in edge data centers. In Proceedings of the 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, Spain, 8–11 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 25–31.
23. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 1628–1656. [CrossRef]
24. Plachy, J.; Becvar, Z.; Mach, P. Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network. *Computer Net.* **2016**, *108*, 357–370. [CrossRef]
25. Silva, B.; Junior, W.; Dias, K.L. Network and cloudlet selection for computation offloading on a software-defined edge architecture. In Proceedings of the International Conference on Green, Pervasive, and Cloud Computing, Uberlândia, Brazil, 26–28 May 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 147–161.
26. Ferguson, T.S. Who solved the secretary problem? *Stat. Sci.* **1989**, *4*, 282–289. [CrossRef]
27. Harth, N.; Anagnostopoulos, C.; Pezaros, D. Predictive intelligence to the edge: Impact on edge analytics. *Evol. Syst.* **2018**, *9*, 95–118. [CrossRef]
28. Harth, N.; Anagnostopoulos, C. Edge-centric efficient regression analytics. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 93–100.
29. Anagnostopoulos, C.; Kolomvatsos, K. Predictive intelligence to the edge through approximate collaborative context reasoning. *Appl. Intell.* **2018**, *48*, 966–991. [CrossRef]
30. Marjanović, M.; Antonić, A.; Žarko, I.P. Edge computing architecture for mobile crowdsensing. *IEEE Access* **2018**, *6*, 10662–10674. [CrossRef]
31. Pu, L.; Chen, X.; Mao, G.; Xie, Q.; Xu, J. Chimera: An energy-efficient and deadline-aware hybrid edge computing framework for vehicular crowdsensing applications. *IEEE Internet Things J.* **2018**, *6*, 84–99. [CrossRef]

32. Anagnostopoulos, C. Time-optimized contextual information forwarding in mobile sensor networks. *J. Parallel Distrib. Comput.* **2014**, *74*, 2317–2332. [CrossRef]
33. Louchard, G.; Bruss, F.T. Finding the k Best Out of n Rankable Objects. A Consecutive Thresholds Algorithm. 2015. Available online: https://www.researchgate.net/publication/282580746_Finding_the_k_best_out_of_n_rankable_objects_A_consecutive_thresholds_Algorithm (accessed on 9 October 2021).
34. Bruss, T. On a Class of Optimal Stopping Problems with Mixed Constraints. *Discret. Math. Theor. Comput. Sci.* **2010**, *12*, 363–380. [CrossRef]
35. SimPy, T. Simpy: Discrete event simulation for python, Python Package Version 3 (9). 2017. Available online: https://simpy.readthedocs.io/en/latest/ (accessed on 1 August 2021).
36. Zhang, D.; Zhao, J.; Zhang, F.; He, T. UrbanCPS: A cyber-physical system based on multi-source big infrastructure data for heterogeneous model integration. In Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems, Seattle, WA, USA, 14–16 April 2015; pp. 238–247.
37. Alibaba Cluster Trace Program Cluster-Trace-v2018. 2018. Available online: https://github.com/alibaba/clusterdata/blob/master/cluster-trace-v2018/trace_2018.md (accessed on 9 October 2021).