Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 5485461, 10 pages
https://doi.org/10.1155/2018/5485461

Research Article

A Simulation-Based Approach of QoS-Aware Service Selection in Mobile Edge Computing

Jiwei Huang[1,2] **Yihan Lan**[3] **and Minfeng Xu**[3]

[1]*Department of Computer Science and Technology, China University of Petroleum - Beijing, Beijing 102249, China*
[2]*Beijing Key Laboratory of Petroleum Data Mining, Beijing 102249, China*
[3]*International School, Beijing University of Posts and Telecommunications, Beijing 100876, China*

Correspondence should be addressed to Jiwei Huang; huangjw05@gmail.com

Received 20 April 2018; Accepted 19 July 2018; Published 1 November 2018

Academic Editor: Kok-Seng Wong

Edge computing is an emerging computational model that enables efficient offloading of service requests to edge servers. By leveraging the well-developed technologies of cloud computing, the computing capabilities of mobile devices can be significantly enhanced in edge computing paradigm. However, upon the arrival of user requests, whether to dispatch them to the edge servers or cloud servers in order to guarantee the quality of service (QoS), i.e., the QoS-aware service selection problem, still remains an open problem. Due to the dynamic mobility of users and the variation of task arrivals and service processes, it is extremely costly to obtain the global optimal solution by both mathematical approaches and simulation-based schemes. To attack this challenge, this paper proposes a simulation-based approach of QoS-aware dynamic service selection for mobile edge computing systems. Stochastic system models are presented and mathematical analyses are provided. Based on the analytical results, the QoS-aware service selection problem is formulated by a dynamic optimization problem. Goal softening is applied to the original problem, and service selection algorithms are designed using ordinal optimization techniques. Simulation experiments are conducted to validate the efficacy of the approach presented in this paper.

1. Introduction

Edge computing is an emerging technique of optimizing cloud computing systems by performing data processing at the edge of the network near the source of the original data [1]. It pushes applications, data, and services away from centralized points (i.e., the cloud) to the logical extremes of a network. Consequently, the communications bandwidth needed between sensors and the central data center by performing analytics and knowledge generation can be significantly reduced. With the rapid development of mobile communications and mobile services, mobile edge computing (MEC) as a special type of edge computing has emerged. It is a network architecture that enables cloud computing capabilities and an IT service environment at the edge of the cellular network [2]. Due to its high performance and strong support for new personalized services for specific customers, MEC has become a hot topic in both industry and academia.

In an MEC system, cellular operators can efficiently deploy their services on the edge nodes which are usually cellular base stations. Since the cellular base stations combine elements of information technology and telecommunications networking in the same equipment, their performance has become a critical issue especially for some special applications such as Internet of Things (IoT). Comparing to the cloud data center which can be commonly regarded as a high-performance computing (HPC) system, the capacity of the base station usually meets some bottlenecks in reality. When the workload of a cellular base station gets heavy, the user requests have to be transferred to the cloud site in order to guarantee the quality of service (QoS) for both the services requested by users and the services already running on the edge node. Therefore, how to determine whether edge servers or cloud servers handle the user requests upon their arrival in order to meet the QoS requirement, i.e., the QoS-aware service selection problem, is critical for the performance of MEC systems.

Due to the high dynamic characteristics of both workload arrival and MEC systems, service selection can be theoretically formulated as a dynamic optimization problem. There have been several mathematical approaches for solving dynamic optimization problems. Nevertheless, most of them meet challenges in large-scale MEC systems. On one hand, dynamic programming based approaches such as Markov decision process (MDP) face a state explosion issue when the scale of MEC system grows reasonably large, resulting in the impossibility of solving the problems in acceptable time. On the other hand, queueing-based approaches, like Lyapunov optimization which is very popular in dynamic optimization nowadays, can only give a lower bound of the optimization algorithms. Furthermore, some of the approaches have to make assumptions to facilitate their optimization procedures, needing additional prior knowledge on the mathematical distributions of the dynamic processes, or resulting in an inaccuracy of performance evaluation. To attack these challenges, another type of approaches has emerged, whose basic idea is to design simulation models and conduct simulation experiments to obtain the optimal service selection policy [3]. However for the QoS-aware service selection problems, the performance of the simulations is a critical issue. Using the simulation to evaluate the output variables for a given setting of the input variables is already computationally expensive not even mention the search of the best policy provided that the input-variable space is huge, and furthermore variability is an integral part of the problem making the simulations more complex [4]. Thus, the simulation approaches sometimes face space explosion problem for both state space and action space. Therefore, how to design and implement efficient approaches for solving the QoS-aware service selection problem remains an open problem.

In this work, we make an attempt at filling this gap and propose an efficient scheme of simulation-based QoS-aware service selection for MEC systems. Simulation models capturing the dynamics and characteristics of MEC systems are carefully designed, and their corresponding mathematical analyses are provided. A framework of event-driven simulation is given, and algorithms of simulation-based service selection are proposed. For facilitating the long-term dynamic optimization, we slightly sacrifice some part of the optimality by softening our goal from "finding the best" to "selecting good enough solutions." Mathematical formulations are presented and quantitative analyses of both performance bound and convergence rate are conducted. By applying ordinal optimization (OO) techniques, we propose an efficient scheme of simulation-based optimization for QoS-aware service selection in MEC systems. Finally, we conduct empirical experiment based on real-life data to validate the efficacy of our approach.

The remainder of this paper is organized as follows. In Section 2, we discuss the related work most pertinent to this paper. In Section 3, we present detailed models for dynamic simulations and conduct quantitative analyses of the models. In Section 4, we design basic framework of the simulation-based approach of MEC service selection and propose an efficient scheme by applying OO techniques. In Section 5, we conduct real-life data based experiments to validate the efficacy of our scheme. Finally, we conclude the paper in Section 6.

2. Related Work

Service selection has been a hot topic in services computing community. The classical service selection was often a question of retrieving functional descriptions from service repositories and the ensuring that the described and required interfaces match a technical level [6]. Solutions regarding the functional aspects of service selection usually studied this problem from modeling or semantics angle [7, 8]. With the rapidly growing number of functional similar services being available on the Internet, the research on service selection has focused on the QoS issues. The very foundation of QoS-aware service selection is to evaluate the QoS of the services. The most straightforward type of the approaches is to design and implement some hardware equipment or computer programs and deploy them in the real-life running systems or emulators to obtain QoS metrics. Existing approaches took full advantage of measurement and feedback techniques for improving the accuracy of the QoS evaluation [9]. Another type of existing approaches is prediction-based scheme, which uses the historical QoS data from previous users of invoked services to predict the QoS metrics by a current user on the particular service before the invoking. Since the prediction-based approaches allow for data missing, they appear powerful strength in QoS evaluation especially in large-scale services computing systems. Several theories and techniques have been applied for improving the predictive quality of the QoS values, and examples include Group Decision Theory [10], collaborative filtering [11], and neural network [12]. The third type of QoS evaluation is model-based method, which is to build a mathematical model for formulating the dynamics of the services and conduct quantitative QoS analysis according to the model parameters. Services can be modeled by Markov chains [13] or queueing models [14] and QoS attributes can be calculated even before services being implemented and deployed, according to which optimal service selection policies can be obtained. Finally, some of the researchers studied this problem from a simulation point of view. System dynamics were analyzed, based on which simulation experiments were designed and conducted for achieving the performance evaluation [15].

With the analytical results, service selection can be studied by solving its correlated optimization problem. Several aspects could be paid attention to while selecting optimal service for users. For example, the diversity of user demands and preferences should be carefully considered especially in the formulation of optimization objectives and constraints [16]. Trustworthiness should be formulated and optimized for better meeting the QoS requirements of the users [17]. Sometimes, the service selection can be formulated as a multidimensional, multiobjective, and multichoice problem, which is so complex to be NP-Hard and can only be solved approximately by heuristic algorithms [18].

Service selection in mobile edge computing may face new characteristics and challenges. Due to the emergence of the edge layer, several additional aspects may affect the

performance of the system, bringing in more uncertain factors to end-to-end QoS. For instance, task arrivals to the MEC system [19], data processing rates in both edge layer and cloud site [20], and service scheduling strategies [21, 22] would all affect the dynamics of the systems, making service selection problem more complex. Therefore, how to design and implement efficient and practical solutions of QoS-aware service selection capturing the dynamics and characteristics of the MEC systems remains largely unexplored. Our simulation-based approach, to be described next, is designed to fill this gap.

3. System Models

In order to solve the service selection problem in mobile edge computing, we have to firstly formulate the problem theoretically. To do so, we study the dynamic behaviours of both mobile users and servers. Mathematical models that capture the dynamics of the systems are presented, and then quantitative analyses are provided. With the analytical results, the problem is hence formulated by an optimization model.

3.1. User Mobility Model. In a mobile edge computing system, we suppose that there are N users, each of which requests for certain services that are deployed on either an edge server or the cloud server. For one of the users $i \in \{1, 2, \ldots, N\}$, let $p_i(t)$ represent the geographical position of the user at the time point t. Since the physical location of a user in an MEC system is usually changing with time, $p_i(t)$ is defined as a stochastic variable, whose distribution varies for different users in different systems.

In an MEC system, a user connects to the system via a cellular base station, which is also an edge server. We suppose that there are M edge servers in the mobile edge computing system. Each of the servers runs several services fulfilling requests from its accessible users. Actually, we usually do not care about the physical location of a user. Instead, we concern which edge server the user connects. Therefore, the value of the stochastic variable $p_i(t)$ can be set by the ID of the accessed edge server, i.e., $p_i(t) \in \{1, 2, \ldots, M\}$.

Computer systems always operate in discrete-time fashion, since each of them is equipped with an internal discrete-time clock. Therefore, the time line can be thought of as a sequence of intervals defined by a sequence of points $t_0 < t_1 < \ldots < t_k < \ldots$. Thus, the collection of $p_i(t)$ is a stochastic process representing the mobility of a user in an MEC system. This can be regarded as a general user mobility model, which can be applied in most types of MEC systems in reality.

3.2. Queueing Model for Edge Servers. The dynamics of an edge server include the following three fundamental parts. Firstly, requests arrive at the server for completing certain tasks according to their requirements. Secondly, since the computational/storage resources of the server are not unlimited, the requests sometimes have to wait in queues until the service is available. Otherwise, arriving requests immediately proceed to the service without queueing. Thirdly, the requests get served and finally depart from the server. Such dynamics can be well captured by a queueing model, which can be represented graphically by Figure 1. The circle represents an edge server, and an open box represents a buffer (queue) preceding this server, where the slots in the queue are meant to indicate waiting requests. The requests are thought of as arriving at the queue and departing from the server, and it is assumed that the service process normally takes a strictly positive amount of time.

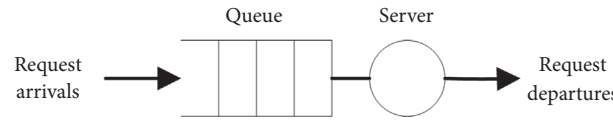

Figure 1: Queueing model of edge server.

The queueing model can be formulated by a discrete event system (DES), where its "events" consist of a sequence of task arrivals and departures. Specifically, let $w_j(t)$ denote the workload (number of requests in the queue) of server $j \in \{1, 2, \ldots, M\}$ at time t, and thus $w_j(t)$ is a stochastic variable which can be defined as the state of the queueing model. In addition, we define $a_j(t, t+1)$ to represent the number of arrival requests at server j in the time interval from t to $t+1$, and $d_j(t, t+1)$ to express the number of departures in the same time interval. Suppose that the edge server has a limited buffer and its buffer size is denoted by B_j. Therefore, the dynamics of the states of the queueing model can be captured by the following expression:

$$w_j(t+1) = \min\left\{B_j, \max\left\{0, w_j(t) + a_j(t, t+1) - d_j(t, t+1)\right\}\right\} \quad (1)$$

The average queue length of the edge server j can be obtained by calculating the mean of $w_j(t)$, as follows.

$$q_j \equiv \mathbf{E}\left[W_j\right] = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} w_j(t) \quad (2)$$

With *Little's law* which provides an all-purpose steady-state performance analysis of queueing systems with any stochastic clock structure, the average response time is proportional to the mean queue length with $1/\lambda_j$ being the constant of proportionality where λ_j is the average arrival rate of requests at edge server j.

$$T_j = \frac{1}{\lambda_j} \mathbf{E}\left[W_j\right] = \frac{q}{\lambda_j} \quad (3)$$

$$\lambda_j = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} a_j(t, t+1) \quad (4)$$

For each of the user requests handled by an edge server, the response time, also known as the sojourn time, can be calculated by the total time that the request spends in the queueing system, which is the summation of its waiting time and its service time. Suppose the arrival time of a request $k \in \{1, 2, \ldots\}$ is denoted by $t(A_k)$ while its departure time

is expressed as $t(D_k)$, and thus the response time can be obtained by (5). We should note that although intuitive, this expression is with value especially for simulation-based schemes, referring readers to Section 4 for details.

$$t_k = t\left(D_k\right) - t\left(A_k\right) \tag{5}$$

3.3. Service Selection Model. The service selection is to determine either the local service located on an edge server or the remote one deployed on cloud site will handle the user requests. Since an edge server can be regarded as an intermediate layer between users and cloud that is able to process in part workload and services locally [23], it usually brings in a performance enhancement especially for some lightweight realtime tasks. However, an edge server is commonly a virtualized lightweight cloud server deployed in a base station, and thus its capacity is much lower than a cloud server. If the workload of an edge server exceeds a certain level resulting in a heavy workload or request congestion, the end-to-end performance decreases dramatically. As a result, the service selection strategy has to be dynamic according to the workload status of the edge servers.

We let a stochastic variable $x_i(t)$ denote the decision variable of service selection in the time epoch of t. We suppose that $x_i(t) = 1$ indicates that the requests submitted by user i will be handled by the accessed edge server $p_i(t)$, while $x_i(t) = 0$ represents that the requests in this time epoch will be submitted to the cloud site. With such definitions, one can obtain that if $x_i(t) = 0$ then there will be nothing submitted to the edge server during the time epoch, and hence (1) can be expressed more specifically by

$$w_j\left(t+1\right) = \begin{cases} \min\left\{B_j, \max\left\{0, w_j\left(t\right) + \sum\limits_{p_i(t)=j} a_i\left(t, t+1\right) - d_j\left(t, t+1\right)\right\}\right\}, & x_i\left(t\right) = 1; \\ \max\left\{0, w_j\left(t\right) - d_j\left(t, t+1\right)\right\}, & x_i\left(t\right) = 0. \end{cases} \tag{6}$$

Considering that the services deployed on the cloud usually have guaranteed QoS via SLA, we assume that the response time of a cloud service to a user request is nearly deterministic. However, as cloud services are often deployed on a remote cloud site, there should be an additional part of the end-to-end response time for a cloud service, i.e., the network communication delay. Comparing with the response time of the services on the edge server, this delay is usually longer. We use (7) to express the end-to-end response time of cloud service for request submitted by user i, where T_i^C is the response time at the cloud site and T_i^N is the communication delay.

$$T_i = T_i^C + T_i^N \tag{7}$$

The service selection scheme is to minimize the response time of all the requests submitted by users, which is one of the most popular approaches for guaranteeing the end-to-end QoS. The decision of service selection is made periodically, and during each time epoch the service selection policy remains the same. We should note that such scheme is time-driven; i.e., the decisions are made at certain time points. One may also design and implement the service selection scheme as an event-driven one; i.e., a decision has to be made upon the occurrence of any event (e.g., arrival and departure). This is a special case of time-driven scheme when we set the time points of making decision to the exact time when events occur. However, the event-driven approaches are usually too costly that the dispatchers have to obtain the optimal solution upon every event resulting in significant overhead. Thus, in the following parts of this paper, we focus on the time-driven schemes which are much more practical in reality.

In a steady-state point of view, the service selection scheme should minimize the average response time of all the user requests. Since the states of edge servers (numbers of requests in the queues) vary with time, the scheme should also be dynamic. Therefore, the dynamic service selection is formulated by a dynamic optimization problem as follows.

Decision Epoch. A decision epoch is indexed by $n \in \{0, 1, \ldots\}$ and correspondingly a decision is executed at time $t = t_0, t_1, \ldots$. If each decision epoch takes equivalent time unites (i.e., τ), then the decisions are made at time $t \in \{0, \tau, 2\tau, \ldots\}$.

States and State Space. The state $S(n)$ at the beginning of epoch n is defined by the state of the system. Since the system consists of M edge servers, we define

$$\begin{aligned} S\left(n\right) &= \left[w_1\left(t_n\right), w_2\left(t_n\right), \ldots, w_M\left(t_n\right)\right] \\ &\in S = B_1 \times B_2 \times \ldots \times B_M \end{aligned} \tag{8}$$

Actions and Action Space. The action of service selection at time epoch n is expressed by

$$a_n = \left[x_1\left(t_n\right), x_2\left(t_n\right), \ldots, x_N\left(t_n\right)\right] \in A = \{0, 1\}^N \tag{9}$$

Optimization Objective. The objective of QoS-aware service selection is to minimize the average response time of all the user requests submitted by users to the system, by tuning the decision variables $x_i(t)$. Mathematically, the objective function is expressed as follows:

$$\underset{a \in A}{\text{maximize}} \quad \mathbf{E}\left[t_k\right] = \lim_{K \longrightarrow \infty} \frac{1}{K} \sum_{k=1}^{K} t_k. \tag{10}$$

We should note that the response time t_k of request k is closely related to the decision made upon its arrival. If the service on the edge server is selected, t_k should be calculated by (5). Otherwise if being served by cloud servers, the request will be responded in T_i time expressed by (7).

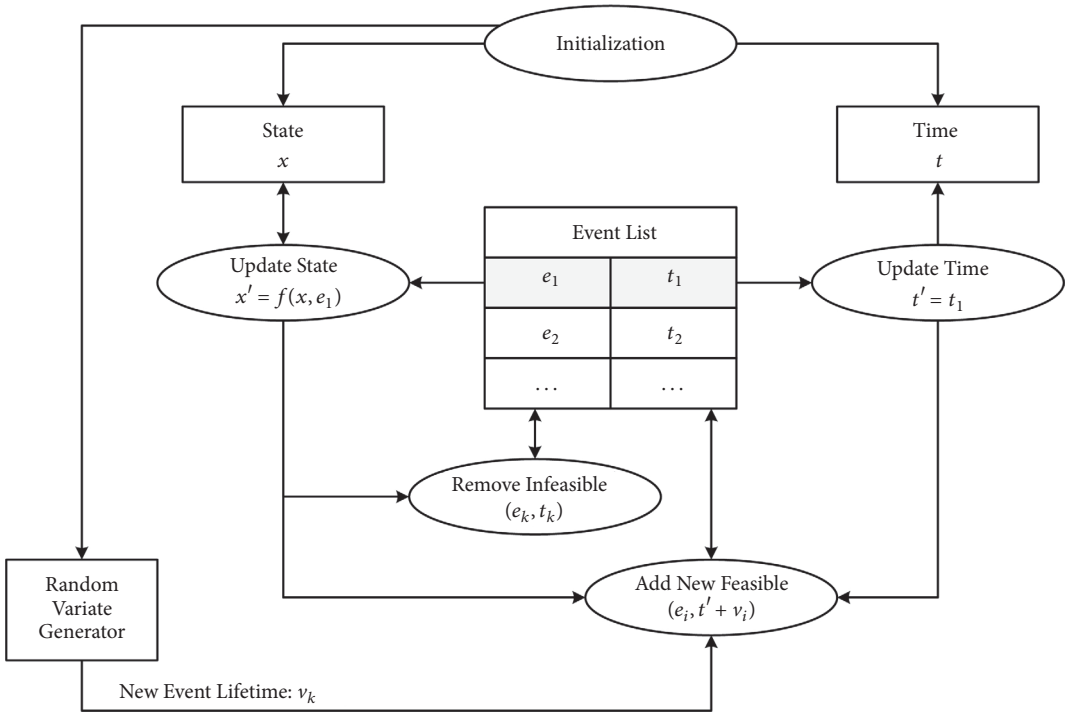

Figure 2: Framework of event-driven simulation.

4. Approach

4.1. Simulation-Based Optimization. With the models presented in the previous section, the service selection problem has been well formulated and can be furthermore solved by optimization theories and techniques. The optimization problem defined by (10) is similar to a Markov Decision Process (MDP) which is a well-known dynamic optimization problem being studied for decades. However, there is a significant difference. The MDP always assigns a reward to each of the states, and the optimization objective is to maximize or minimize the expected value of the reward, either discounted or undiscounted. But for our problem shown in (10), the objective is to minimize the average of response time which is a transient time-varying variable. The transient response time is correlated to both of the status of queues and policies after decisions having been made. Therefore, the Markovian (memoryless) property sometimes may not hold, resulting in significant difficulty in optimization procedures. Furthermore, real-world systems do not conform to some assumptions (e.g., Poisson arrivals, exponential service rates) we make in order to simplify a model, and they are too complex to yield analytical solutions. Since both analytical and algorithmic solutions may fail for solving this problem, we design a simulation-based approach.

The basic idea of simulation is to conduct a series of experimental processes through which the system models are evaluated numerically, and the data from the processes are used to estimate various quantities of our interests. The states of the physical system are represented by state variables, and the simulation program modifies them to reproduce the evolution of the physical systems over time. With different policies, the performances of the system are evaluated and compared, based on which the optimal solution will finally surface. The most appealing advantage of simulation-based optimization is its relative simplicity and wide applicability, especially comparing to old-fashioned laboratory experiments in which the real physical systems have to be implemented. The only hardware involved is a computer, and instead of physical servers connected with each other we have software programs capturing all such interactions and activities. Randomness is also fully considered, and replaced by appropriate software driven by random variate generator.

Although the service selection scheme is time-driven, we implement our simulation using an event-driven fashion, in order to capture all the dynamics of the whole MEC system. The overall framework of the simulation is shown by Figure 2. The basic procedure of a simulation experiment is to continuously repeat the following five steps. (1) The first entry (e_1, t_1) from the *event list* is selected and removed. (2) We update the simulation *time* by advancing it to the new event time t_1. (3) The *state* of the system x is updated according to the state transition function $x' = f(x, e_1)$. More specifically, if e_1 is an arrival event, we have $x' = f(x, e_1) = x + 1$; otherwise if e_1 is a departure, and thus we obtain $x' = f(x, e_1) = x - 1$. (4) If there exist any infeasible events in the new state, we remove their corresponding entries from the event list. (5) The new feasible events triggered by e_1 will be added into the event list. The scheduled event time for event e_i is given by $t_1 + v_i$, where v_i is a lifetime obtained from the *random variate generator*.

The simulation-based approach of QoS-aware service selection in MEC systems is designed and implemented as

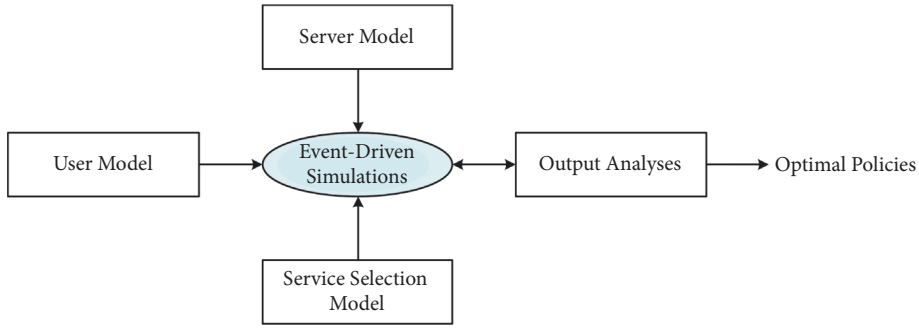

Figure 3: Framework of simulation-based service selection.

Figure 3. Based on the user mobility model, server queueing model, and service selection model presented in the previous section, we conduct simulation experiments for certain parameter settings and analyze the output (e.g., average response time of user requests). With different policies, the output is analyzed and compared, among which the optimal solution can be obtained.

4.2. Goal Softening. The conventional simulation-based optimization is to experimentally evaluate all the candidate policies and select the one with the optimal objective. However, such type of approach meets challenges especially in the long-term optimization of the service selection problem discussed in this paper.

First and foremost, the search space is extremely large. From the discussions in Section 3.3, we obtain that the number of feasible policies in only one decision epoch is 2^N. Furthermore, in one of the simulation experiments, we have to make decisions in a series of decision epoches, resulting in a further exponential increase of the search space. The number will become extremely large especially when the scale of the MEC systems grows up. Consequently, there is a critical limit of the scalability of the conventional simulation-based optimization approaches.

Secondly, the number of random variables is also extremely large. For each of the events in the simulations, i.e., task arrivals and service processes, the random variate generator has to generate a new random variable in order to obtain new feasible events. In each of the decision interval, there might be a number of event having occurred, and thus more events are generated. In an MEC system with multiple edge nodes, the event table shown in Figure 2 grows very fast. Since plenty of correlated data has to be recorded and calculated, both the overall performance and the memory space will face critical challenges in the simulation programming.

Last but not least, since the workload in reality is usually dynamic and varies with time, we have to simulate quite a long time period in order to capture the characteristics of the workload for guaranteeing the accuracy of the simulation results. However, such is quite time-consuming and resource-consuming. For some large-scale MEC system, it is impossible for us to conduct such long-term simulations for every feasible policy with existing computer systems.

Therefore, in order to attack these challenges, we have to sacrifice some part of the optimality for solving the optimization problem in reasonable time. Here, we borrow the idea from ordinal optimization (OO) which was firstly proposed by Ho et al. [24] for solving extremely complex search-based optimization problems. The basic idea of OO is "soft optimization for hard problems," which means that one can solve the hard problem within an acceptable time after softening the optimization objective.

Since we have known that finding the global optimal solution is practically infeasible, we switch our goal to a reasonable one which is to find a good enough solution with high probability. Mathematically, the goal is expressed as

$$\Pr\left(|G \cap S| \geq k\right) \geq \alpha \tag{11}$$

Here, G is the actual good enough set which is usually the top-g feasible solutions for the optimization problem, where $g = |G|$. S is the set of selected solutions by OO which is usually the estimated top-s solutions selected by simulation experiments, where $s = |S|$. $\Pr(|G \cap S| \geq k)$ is called *alignment probability* which indicates the probability that there are actually k truly good enough solutions in S, and k is called the alignment level. We should note that, in most cases where the users usually select the best one solution in S, k can be set to 1.

The first basic idea of OO is that ordinal optimization is much easier than cardinal optimization. A traditional cardinal problem usually asks us to estimate the difference in performance between two policies. In simulation-based optimizations, since all the simulation experiments are identical and independent, the overall performance (e.g., average response time) is calculated using a mean estimator by averaging all the experimental results obtained from the simulations. We suppose that there are l identical and independent simulation experiments obtaining a sequence of i.i.d. samples of the estimate observations expressed by $X(i)$ $(i = 1, 2, \ldots, l)$. Consequently, the estimator is expressed as $\widetilde{X} = (1/l)\sum_{i=1}^{l} X(i)$. We let X denote the actual value of the overall performance. With *Strong Law of Large Numbers*, we have $\lim_{l\longrightarrow\infty}\widetilde{X} = \lim_{l\longrightarrow\infty}(1/l)\sum_{i=1}^{l} X(i) = X$, indicating that the estimator is unbiased. Furthermore, we can conclude that the standard deviation of the mean estimator can be calculated by $\sigma_{\widetilde{X}} = (1/\sqrt{l})\sigma_X$, which shows that such estimate's convergence rate of no faster than $l^{-1/2}$ is unsatisfactorily slow. On the other hand, the *Central Limit Theorem* illustrates that the sample mean $\widetilde{X}$ converges to

1: Use a crude and computationally fast model to estimate the performance of all feasible policies.
2: Estimate the Ordered Performance Curve (OPC) class of the problem and the noise level of the crude model.
3: Use the table presented in [5] to calculate the size of selected set s.
4: Select the estimated top-s policies by the crude model as the selected set S.
5: Simulate each policy in the selection set with a precise model.
6: Apply the best policy from the simulation results in step 5.

Algorithm 1: OO-Based Service Selection Scheme for MEC.

Gaussian distribution when l is large enough. Therefore, if we only want to determine whether a policy is better than another one by comparing their simulation output $\widetilde{X}_1$ and $\widetilde{X}_2$ following OO spirit, we find that the differential $\widetilde{X}_2 - \widetilde{X}_1$ also conforms to Gaussian distribution. Suppose that the actual values have the relationship as $\Delta X = X_2 - X_1 > 0$, and thus we have $\Pr(\widetilde{X}_2 - \widetilde{X}_1 > 0) = \Phi(\sqrt{l}\Delta X) = 1 - O(l^{-1/2} \cdot e^{-\Delta X^2 l})$, showing the order between estimates of the two results agreeing the true order converges exponentially to 1 with rate no slower than ΔX^2. In conclusion, with much higher convergence rate, OO is easier than conventional cardinal optimization.

The second idea of OO is that the optimization with a softer goal of being good enough is much easier than trying to find the exact one best solution. This relaxing of the goal can buy us a lot in the easing of the computational and memory burden, meanwhile guaranteeing the optimality in an acceptable level. We discuss a blind picking scheme which is able to provide a lower bound analysis for the alignment probability. Such scheme is to just randomly pick s policies from the search space to obtain the selected set S, and thus no prior knowledge is used in the selection. Therefore, we have the alignment probability when $k = 1$ given by the following expression, where Θ represents the overall search space.

$$\begin{aligned} \Pr(|G \cap S| \geq 1) &= 1 - \frac{\binom{|\Theta|-g}{s}}{\binom{|\Theta|}{s}} \geq 1 - \left(1 - \frac{g}{|\Theta|}\right)^s \\ &\geq 1 - e^{-gs/|\Theta|} \end{aligned} \tag{12}$$

Therefore, the alignment probability converges exponentially with respect to the size of the set G and S. Furthermore, the lower bound of the alignment probability has a general form as follows:

$$\Pr(|G \cap S| \geq k) \geq \sum_{i=k}^{\min(g,s)} \frac{\binom{g}{i}\binom{|\Theta|-g}{s-i}}{\binom{|\Theta|}{s}} \tag{13}$$

4.3. Service Selection Scheme. We take advantage of OO techniques to solve the dynamic service selection problem. After goal softening, an efficient simulation-based scheme for dynamic service selection in MEC systems is carefully designed. The basic procedures of our approach are shown by Algorithm 1.

We firstly use a rough model to get the top-s candidate policies. The crude model will provide a rough estimation of the performance of each policy, but it is quite efficient in both computational operations and memory space. In the optimization problem defined by service selection in MEC presented in Section 3, simulation parameters are carefully controlled. Since the resource consumption is closely correlated to the simulation time, we implement our crude model by estimating the performance of the feasible policies using simulation experiments for only a relatively small time period. In other words, although the parameter K in (10) should be large enough to obtain an accurate estimate of the average response time, we set it a very small number in our crude model, which makes the simulations complete in a short time resulting in small resource consumption. Consequently, step 1 can be completed in a reasonable time.

In the next step we determine the parameters of ordinal optimization. The user specifies the size of good enough set g and the required alignment level k. With well-developed theoretical and practical results, after estimating the Ordered Performance Curve (OPC) class of the problem and the noise level of the crude model by our approach, we are able to obtain the appropriate size of selected set s by looking up a precalculated table [5].

After the top-s feasible policies have been selected by the crude model, we apply the precise model simulation on the s candidate policies to find the optimal one. We notice that the precise model simulations are much more expensive than the previous crude version, which takes much more time and computational resources. However, we only need to run the precise simulations on the s policies, and thus comparing to the original search space Θ the overhead is quite acceptable. Also, we will illustrate such issue in the next section by the experimental results obtained from simulation experiments we conduct in reality.

5. Evaluation

5.1. Experimental Setup. We conduct experiments based on real-life data to validate the efficacy of our approach. The QoS-aware service selection scheme introduced in the previous sections is implemented in simulation experiments, where workload is generated according to real-world trace data and OO theory is applied to find a good enough solution in a reasonable time.

We apply a real-world data set released by Microsoft Research, namely, "T-Drive" to generate the workload in our experiments. It contains the GPS trajectories of 10,357 taxis collected by GPS loggers and GPS phones within the city of Beijing during a period of one week in the year of 2008 [25, 26]. The exact time of each piece of data being submitted to

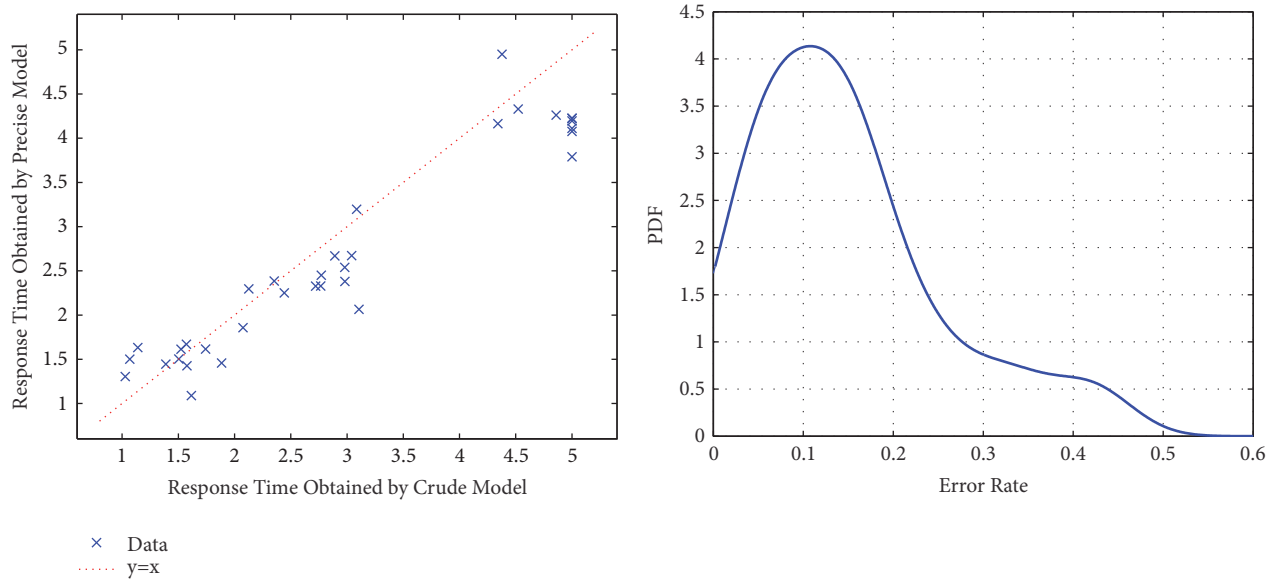

Figure 4: Estimation error of crude model from precise model.

the system has been recorded, which is of valuable reference of the task arrivals from different users. In our experiments, the time when the users initiate a request is followed by the timestamp of each piece of taxi GPS data. We assume that the users in our MEC system basically follow a random walk mobility model within different coverage areas of different edge servers, and we implement an exponentially distributed random variable generator for the service processing events in the edge servers. The service selection scheme is designed and implemented in a time-driven fashion, following the procedures presented in Section 4.

An MEC system is simulated on a PC environment with an Intel quad-core CPU and 8GB memory. A simulator is designed and implemented capturing the basic dynamics of the system behaviors, including request arrivals, task scheduling, and service procedures. With our approach, experimental data is collected and analyzed, which will validate the effectiveness and efficiency of our approach. We defer readers to the following subsection for details.

5.2. Experimental Results. We conduct the simulation experiments several times by tuning the parameters such as the number of decision epoches that we consider during the long-term optimization (i.e., K) and the number of edge servers in the MEC system (i.e., M). Also, both crude model and precise model are implemented. After running the simulation experiments, we illustrate the experimental results. We should note that, during our experiments, we find that the conventional simulation-based optimization approach takes so long time that we are not able to complete all the experiments in a reasonable time. Therefore, we illustrate the comparison between these two approaches within the limited cases.

Firstly, we evaluate the effectiveness of our approach. The response times obtained by the crude model and the precise model are shown by the first subfigure of Figure 4. The x-axis indicates the response time obtained using the crude model in our simulations, while y-axis illustrates the response time calculated by the simulation results using the precise model. We also draw a dot line of $y = x$ for clear demonstration. The closer the data points are to the dot line, the less estimating error the crude model has. We obtain from Figure 4 that most of the data points are close to the line. The second subfigure further analyzes the distribution of the error rate by illustrating its probability density function (PDF). It has been shown that the majority of the data is within the error rate below 20%. Quantitatively, we calculate the mean error rate of the crude model, showing that such value is around 16.3%. Considering the significant reward in reducing the search space of the crude model, such error rate is quite satisfactory.

Secondly, we tune the experimental parameters to discuss the size of search space with the increase of the scale of the optimization problem. The experimental results are shown by Figure 5. On one hand, we find that the search space of the precise model increases exponentially with the increase of the decision period time, indicating that the conventional simulation-based approaches have to spend plenty of time and resources for obtaining a long-term optimality. However, the search space of the crude model basically remains the same, since we only conduct a fixed short-term dynamic programming optimization for obtaining the selected set. Consequently, using crude model is robust to the increase of the long-term optimization. On the other hand, we find from the second subfigure that the search spaces of both crude model and precise model grow exponentially with the increase number of the edge servers. The reason is that the action space increases exponentially in this case, and both of the two models have to look for the optimal solutions in a much wider scope. However, it is clear that there is a significant difference between their increase rates. The size of search space in the crude model grows slower than precise

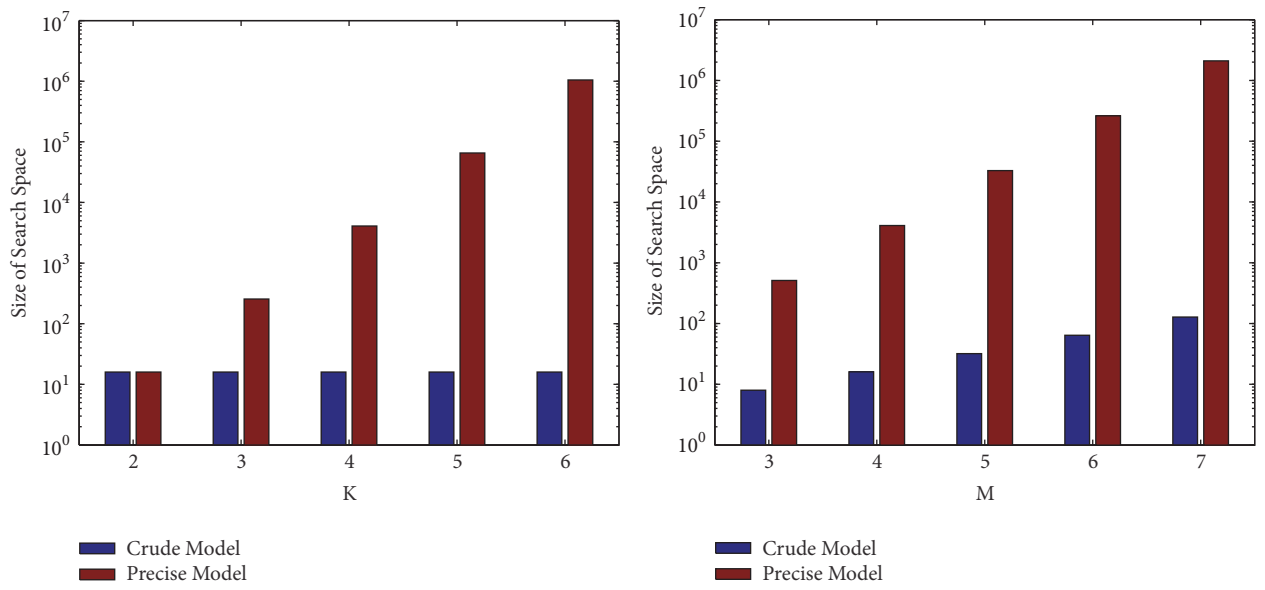

Figure 5: Size of search space of the two models.

Table 1: Experimental results of running time.

Scheme		Running Time
Traditional Approach		161min36s
OO	Crude Model	3s
	Precise Model	20min12s
	Total	20min15s

model. In summary, the effectiveness of our approach in reducing the search space can be validated.

Finally, we evaluate the running time of our approach comparing with the traditional simulation-based optimization. We show a group of experimental results for a small-sized MEC system, since the traditional approach is not able to complete the task in acceptable time if the search space grows larger. Shown as Table 1, one can obtain that the computational complexity of the crude model is significantly low. Applying ordinal optimization in search-based service selection optimization is able to nearly reduce the overhead by one order of magnitude, which validates the efficiency of our approach.

6. Conclusion

Service selection is an important and open problem in mobile edge computing for guaranteeing the quality of service. This paper proposes an efficient simulation-based service selection approach for MEC systems, tackling the challenges of state explosion and high variability in simulation-based optimization. Frameworks, models, analyses, and algorithms are discussed in detail, and empirical validation is conducted based on trace data obtained from reality. It has been proved both theoretically and experimentally that the performance can be significantly improved by slightly softening the optimization objective with a sacrifice of global optimality of the obtained solutions. This work is expected to bring a new idea for solving the optimization problems in large-scale MEC systems and provide with a practically efficient solution for optimal QoS-aware service selection.

Data Availability

Previously reported T-Drive dataset was used to support this study and is available at https://www.microsoft.com/en-us/research/publication/t-drive-driving-directions-based-on-taxi-trajectories/. This prior dataset is cited at relevant places within the text as [25, 26]. Most of the simulation experimental data used to support the findings of this study are included within the article. Further additional data are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by Beijing Natural Science Foundation (no. 4162042), National Natural Science Foundation of China (no. 61502043), and National Key Research and Development Plan (no. 2016YFC0303700).

References

[1] P. G. Lopez, A. Montresor, D. Epema et al., "Edge-centric computing: vision and challenges," *Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[2] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proceedings of the 10th International Conference on Intelligent Systems and Control, ISCO 2016*, India, January 2016.

[3] F. Patrizi, "An Introduction to Simulation-Based Techniques for Automated Service Composition," *Electronic Proceedings in Theoretical Computer Science*, vol. 2, pp. 37–49, 2009.

[4] S.-C. Horng and S.-S. Lin, "An ordinal optimization theory-based algorithm for a class of simulation optimization problems and application," *Expert Systems with Applications*, vol. 36, no. 5, pp. 9340–9349, 2009.

[5] T. W. E. Lau and Y.-C. Ho, "Universal alignment probabilities and subset selection for ordinal optimization," *Journal of Optimization Theory and Applications*, vol. 93, no. 3, pp. 455–489, 1997.

[6] S. Reiff-Marganiec and M. Tilly, "Non-functional property based service selection: A survey and classification of approaches," in *Proceedings of the 2008 Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop*, 2008.

[7] P. C. Xiong, C. Pu, and M. C. Zhou, "Protocol-level service composition mismatches: A petri net siphon based solution," *International Journal of Web Services Research*, vol. 7, no. 4, pp. 1–20, 2010.

[8] M. Rouached, W. Fdhila, and C. Godart, "Web services compositions modelling and choreographies analysis," *International Journal of Web Services Research*, vol. 7, no. 2, pp. 87–110, 2010.

[9] S. Wang, A. Zhou, W. Lei, Z. Yu, C. Hsu, and F. Yang, "Enhanced User Context-Aware Reputation Measurement of Multimedia Service," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 4s, pp. 1–18, 2016.

[10] Y. Ma, S. Wang, P. C. Hung, C. H. Hsu, Q. Sun, and F. Yang, "A highly accurate prediction algorithm for unknown web service QoS value," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 511–523, 2016.

[11] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, and T. Zhang, "Collaborative web service quality prediction via exploiting matrix factorization and network map," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 126–137, 2016.

[12] W. Xiong, Z. Wu, B. Li, and Q. Gu, "A Learning Approach to QoS Prediction via Multi-Dimensional Context," in *Proceedings of the 24th IEEE International Conference on Web Services, ICWS 2017*, pp. 164–171, USA, June 2017.

[13] R. S. Matos, P. R. M. Maciel, and R. M. A. Silva, "QoS-driven optimisation of composite web services: An approach based on GRASP and analytical models," *International Journal of Web and Grid Services*, vol. 9, no. 3, pp. 304–321, 2013.

[14] L. Li, S. Li, and S. Zhao, "QoS-Aware scheduling of services-oriented internet of things," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1497–1507, 2014.

[15] A. P. T. Hsu, W. T. Lee, A. J. C. Trappey, C. V. Trappey, and A.-C. Chang, "Using System Dynamics Analysis for Performance Evaluation of IoT Enabled One-Stop Logistic Services," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, SMC 2015*, pp. 1291–1296, Hong Kong, October 2015.

[16] L. Yang, L. Liu, and Q. Fan, "A Survey of User Preferences Oriented Service Selection and Deployment in Multi-Cloud Environment," in *Proceedings of the 2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pp. 354–359, Taipei, December 2017.

[17] H. Ma, Z. Hu, and M. Cai, "Trustworthy service selection integrating cloud model and possibility degree ranking of interval numbers," *Journal of Electronics*, vol. 26, no. 6, pp. 1177–1183, 2017.

[18] M. C. Jaeger, G. Mühl, and S. Golze, "QoS-aware composition of web services: A look at selection algorithms," in *Proceedings of the 2005 IEEE International Conference on Web Services, ICWS 2005*, pp. 807–810, USA, July 2005.

[19] Z. Wang, Q. Zhao, F. Xu, H. Dai, and Y. Zhang, "Detection performance of packet arrival under downclocking for mobile edge computing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 9641712, 7 pages, 2018.

[20] G. Li, J. Wang, J. Wu, and J. Song, "Data Processing Delay Optimization in Mobile Edge Computing," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[21] X. Lyu, H. Tian, L. Jiang et al., "Selective Offloading in Mobile Edge Computing for the Green Internet of Things," *IEEE Network*, vol. 32, no. 1, pp. 54–60, 2018.

[22] Li. W, Y. Xia, M. Zhou, X. Sun, and Q. Zhu, "Fluctuation-aware and predictive workflow scheduling in cost-effective Infrastructure-as-a-Service clouds," *IEEE Access*, 2018.

[23] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[24] Y. C. Ho, R. S. Sreenivas, and P. Vakili, "Ordinal optimization of DEDS," *Discrete Event Dynamic Systems*, vol. 2, no. 1, pp. 61–88, 1992.

[25] J. Yuan, Y. Zheng, C. Zhang et al., "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th International Conference on Advances in Geographic Information Systems ACM SIGSPATIAL (GIS '10)*, pp. 99–108, November 2010.

[26] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*, pp. 316–324, August 2011.