Hindawi
Mobile Information Systems
Volume 2019, Article ID 8172698, 12 pages
https://doi.org/10.1155/2019/8172698

Research Article

Markov Approximation for Task Offloading and Computation Scaling in Mobile Edge Computing

Wenchen Zhou,[1] Weiwei Fang,[1] Yangyang Li,[2] Bo Yuan,[1] Yiming Li,[1] and Tian Wang[3]

[1]*School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China*
[2]*Innovation Center & Mobile Internet Development and Research Center, China Academy of Electronics and Information Technology, Beijing 100041, China*
[3]*Department of Computer Science and Technology, Huaqiao University, Xiamen 362021, China*

Correspondence should be addressed to Weiwei Fang; wwfang@bjtu.edu.cn

Received 6 August 2018; Revised 22 October 2018; Accepted 12 December 2018; Published 23 January 2019

Academic Editor: Laurence T. Yang

Mobile edge computing (MEC) provides cloud-computing services for mobile devices to offload intensive computation tasks to the physically proximal MEC servers. In this paper, we consider a multiserver system where a single mobile device asks for computation offloading to multiple nearby servers. We formulate this offloading problem as the joint optimization of computation task assignment and CPU frequency scaling, in order to minimize a tradeoff between task execution time and mobile energy consumption. The resulting optimization problem is combinatorial in essence, and the optimal solution generally can only be obtained by exhaustive search with extremely high complexity. Leveraging the Markov approximation technique, we propose a light-weight algorithm that can provably converge to a bounded near-optimal solution. The simulation results show that the proposed algorithm is able to generate near-optimal solutions and outperform other benchmark algorithms.

1. Introduction

In recent years, mobile devices (MDs) have become an indispensable tool for communication, information, and entertainment in our daily life. However, finite battery capacities and limited computation resources pose intractable challenges for satisfying user-experience requirements. Computation offloading is recognized as a promising solution to cope with such a problem, by migrating computation tasks from mobile devices via wireless access to more powerful servers [1]. Mobile cloud computing (MCC) has been considered as one of the potential solutions. It is commonly assumed that the implementation of MCC relies on data exchange with a centralized cloud through wide area networks [2]. Nevertheless, MCC imposes huge traffic load on mobile networks and brings high communication latency due to the long distance from MDs to the cloud.

Mobile edge computing (MEC), which deploys MEC servers directly at the base stations using generic-computing platforms, is a newly proposed solution for the above problem. In this paradigm, IT and cloud-computing services are provided in close proximity to mobile devices [3]. By endowing ubiquitous wireless access networks (e.g., macrocell and small-cell base stations) with resource-rich computing infrastructures, MEC is envisioned to provide pervasive and agile computation augmenting services when and where are needed. Since the concept of MEC was proposed by European Telecommunications Standards Institute (ETSI), it has attracted increasing attentions from academic researchers. In particular, one of the key design issues in MEC is resource allocation [3]: should a computation task be processed locally by a MD's CPU or remotely by a MEC server, and if the latter is chosen, how many resources should be allocated to this task? This question stems from the basic tradeoff between the cost of task offloading and the reduction in task execution time brought by offloading. While conceptually simple, it is challenging to make optimal decisions since many factors are coupled and

the solution space is very large. Various approaches have been proposed to tackle this resource allocation problem in many kinds of scenarios [3, 4]. Some of them focused on single-user cases, while the others focused on multiuser cases. However, in these studies, the MD is assumed to be associated with only a single MEC server. With the expectation of small-cell/femtocell base stations been massively deployed in future networks, a MD can choose to offload its tasks to multiple nearby MEC servers with computational capabilities, other than only one MEC server [5].

This paper focuses on system optimization in scenarios where a single MD is capable of scaling its CPU frequency and allocating computation tasks to multiple MEC servers. Specifically, we exploit the diversity in terms of task assignment and CPU frequency to make optimal control decisions so as to minimize the tradeoff between task execution time and mobile energy consumption. We formulate such a problem as a combinatorial optimization, NP-hard problem. Inspired by the Markov approximation framework proposed in [6], we have devised an efficient approximation algorithm with near-optimal performance. In summary, the main contributions of this paper are as follows:

(1) We propose to exploit the task assignment decision and the computation scaling capability to jointly optimize the tradeoff between latency and energy in a multiserver MEC system and then formulate this as a nonlinear combinatorial optimization problem.

(2) We devise an approximation algorithm based on the Markov approximation framework [6] to solve the proposed problem efficiently. This algorithm can find the near-optimal solution by implementing a Markov chain over all feasible configurations and performing state transitions [6–8]. Then, we investigate key characteristics of the designed Markov chain and analyze the algorithm in terms of performance optimality, approximation gap, and error robustness.

(3) We conduct simulation experiments to demonstrate performance of our Markov approximation-based algorithm under various parameter settings. The simulation results show that this algorithm can generate near-optimal solutions and remarkably outperforms other benchmark algorithms.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the system model and the problem definition. Then, the proposed Markov approximation-based algorithm is introduced in Section 4. Section 5 demonstrates the simulation results. Finally, Section 6 summarizes the conclusions and outlines future work.

2. Related Work

From the computation perspective, MEC offers a new service environment characterized by proximity, efficiency, low latency, and high availability, making computation offloading a promising paradigm for MDs [3]. To this end, three important issues have to be taken into account, namely, resource allocation, data partition, and optimization objective.

Since offloading introduces additional communication overhead, a key technique challenge is how to allocate computation and communication resources so as to balance the energy-performance tradeoff and support the user-experience demands [4]. Recent years have seen increasing research progresses on resource allocation for both single-user [9–11] and multiuser MEC systems [12–14]. Wang et al. [9] investigated computation offloading in MEC by jointly optimizing a MD's CPU speed, transmit power, and offloading ratio to achieve two different design objectives, i.e., minimizing energy consumption and minimizing execution time, of the MD. Liu et al. [10] adopted a Markov decision process approach to solve the power-constrained latency minimization problem in MEC, where a MD schedules its computation tasks based on queue size, execution state, and channel information. You et al. [11] proposed a framework in which a MD can not only process computations tasks at local CPU or offload them to the MEC server but also harvest energy from the base station by microwave power transfer (MPT). The offloading problem in multiuser cases is more complex than that in single-user cases. You et al. [12] studied the optimal resource allocation for a multiuser MEC system with both time division multiple access (TDMA) and orthogonal frequency division multiple access (OFDMA). Chen et al. [13] proposed a game theoretic scheme for the computation offloading decision making problem in multiuser scenarios and demonstrated that the designed game always admits a Nash equilibrium. Sardellitti et al. [14] designed an iterative algorithm based on successive convex approximation techniques to minimize the overall users' energy consumption with latency constraints in a MIMO multicell system. However, the work introduced above only considered the scenario where a MD only associates with a single edge server. Since the future mobile networks will be heterogeneous due to dense deployment of base stations with different capabilities [3], we can exploit such a diversity to provide more offloading options and sufficient resource capacities to MDs for guaranteeing low service latency and satisfactory user experience [15].

Generally, computation offloading could be performed in two fashions, i.e., full offloading and partial offloading [9]. In full offloading, the mobile application has to be executed as a whole either locally at the MD or remotely at the MEC server [14]. Compared with full offloading, partial offloading takes advantage of parallelism between the MD and the MEC server, so it is much more capable to satisfy stringent latency requirement. However, existing work [9, 16] often assumed full granularity of data partition, i.e., the offloaded data could be partitioned as small as possible. In practice, a mobile application may contain some indivisible tasks or files that cannot be separated into parts of any size [5].

Computation offloading affects both mobile energy consumption and task execution time. On the one hand, in situations where the MD with latency-sensitive applications has a stringent requirement on energy consumption, it is essential to apply latency-oriented solutions [9, 10, 17] to

utilize limited energy efficiently, so as to shorten the execution time as much as possible. On the other hand, in order to prolong the MD's lifetime, energy-oriented solutions [9, 11, 12, 14] are proposed to minimize the overall energy consumption of MDs while guaranteeing the latency requirement of mobile applications. As far as we know, only a few studies [5, 13, 15] have focused on optimizing these two metrics simultaneously.

While recognizing their significance, our work is different from and complementary to existing studies. We investigate a joint optimization of task offloading and computation scaling problem in a multiserver MEC system, with the objective to minimize the tradeoff between latency and energy. Besides, in system modeling, we take into account many practical aspects that were missing or unconsidered in previous work investigating relevant problems. To our best knowledge, the proposed problem has not been explored by prior work.

3. System Model and Problem Formulation

3.1. System Model. Let us consider a MD that has a set of tasks $\mathcal{M}$ to be processed. It can choose to either process these tasks at local CPU or offload them to any one among the nearby MEC servers $\mathcal{N}$. We assume that the wireless base stations operate on orthogonal wireless channels so that any two of them will not interfere each other. We leverage the binary variables $x_m, y_{m,n} \in \{0,1\}$, $\forall m \in \mathcal{M}$, and $\forall n \in \mathcal{N}$, to represent the task assignment, i.e.,

$$x_m = \begin{cases} 1, & \text{if task } m \text{ is processed at local CPU,} \\ 0, & \text{if task } m \text{ is processed at a MEC server,} \end{cases}$$
$$y_{m,n} = \begin{cases} 1, & \text{if task } m \text{ is assigned to MEC server } n, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

A task must be processed locally or remotely, i.e., the aforementioned $|\mathcal{M}|$ tasks could be separated into $|\mathcal{N}|+1$ disjoint sets. To ensure this, we impose the following constraint:

$$x_m + \sum_{n\in\mathcal{N}} y_{m,n} = 1, \quad \forall m \in \mathcal{M}. \tag{2}$$

A computation task m is characterized by a three-tuple of parameters, (a_m, b_m, c_m), where a_m denotes the total number of CPU cycles needed to accomplish the task m, while b_m and c_m denote the size of computation input data (in bits) and output data (in bits), respectively. In this work, we assume that the MD can apply the methods such as offline measurements [18] and call-graph analysis [13] to obtain the values of a_m, b_m, and c_m. We now analyze the computation overhead in terms of both execution time and energy consumption for both local and offloading approaches.

(1) *Local computing*: If the MD chooses the local computing approach, it will execute the computation task m locally using its own CPU. Let ψ_{MD} be the computation capability (i.e., CPU cycles per second) of the MD. Given the decision profile $\mathbf{x} = \{x_m\} \in \{0,1\}^{|\mathcal{M}|}$ and ψ_{MD}, the execution time of computing a batch of tasks at local CPU is given by

$$T_{\mathrm{MD}}(\mathbf{x}, \psi_{\mathrm{MD}}) = \sum_{m\in\mathcal{M}} x_m \frac{a_m}{\psi_{\mathrm{MD}}}. \tag{3}$$

For the computational energy, we have that

$$E_{\mathrm{MD}}(\mathbf{x}, \psi_{\mathrm{MD}}) = \left(\Upsilon_1 \psi_{\mathrm{MD}}^{\theta} + \Upsilon_2\right) T_{\mathrm{MD}}(\mathbf{x}, \psi_{\mathrm{MD}}), \tag{4}$$

where $(\Upsilon_1 \psi_{\mathrm{MD}}^{\theta} + \Upsilon_2)$ denotes the computational power of the MD. As in [19], θ ranges from 2 to 3, while Υ_1 and Υ_2 are parameters depending on chip architecture. Their values can be obtained by the measurement approach in [19]. Using dynamic voltage frequency scaling (DVFS) technology [4, 14, 19], the MD could adaptively adjust ψ_{MD} to shorten execution time or reduce energy consumption. For example, the Nexus S smartphone has six levels of CPU speeds, each of which matches with some specific voltage [19]. In this work, we assume that ψ_{MD} takes value in some finite and discrete set $\Psi = \{\psi | \psi^{\min} \le \psi \le \psi^{\max}\}$, where $\psi^{\min}$ and $\psi^{\max}$ are minimum and maximum CPU frequency of the MD, respectively.

(2) *MEC offloading*: If the MD chooses the MEC offloading approach, it will offload the computation task m to one of the MEC servers via wireless access. This chosen server will execute task m on behalf of the MD. Such computation offloading would incur extra overhead in terms of time and energy for transmitting the computation input and output data. We assume that each MEC server n can provide the MD with fixed service rate (i.e., CPU cycles per second) ψ_n, which is determined according to the MEC computing service contract subscribed by the MD from its mobile operator [13]. The average uplink and downlink data rates, denoted by r_n^{UL} and r_n^{DL}, are also assumed to be known by the MD before task processing by applying the methods in [1, 5]. For simplicity, the data transmission and the task processing are assumed to be nonoverlapping and noninterfering with each other, which means the uploading, computing, and downloading steps are carried out sequentially [5]. Given the decision profile $\mathbf{y} = \{y_{m,n}\} \in \{0,1\}^{|\mathcal{M}|\times|\mathcal{N}|}$, the total execution time of computing a batch of tasks at MEC server n can be given as

$$T_n(\mathbf{y}) = \sum_{m\in\mathcal{M}} y_{m,n}\left(\frac{a_m}{\psi_n} + \frac{b_m}{r_n^{\mathrm{UL}}} + \frac{c_m}{r_n^{\mathrm{DL}}}\right). \tag{5}$$

The MD's energy consumption on wireless transmission can be calculated as

$$E_{\mathrm{MEC}}(\mathbf{y}) = P_{\mathrm{tx}} \sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} y_{m,n}\frac{b_m}{r_n^{\mathrm{UL}}} + P_{\mathrm{rx}} \sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} y_{m,n}\frac{c_m}{r_n^{\mathrm{DL}}}, \tag{6}$$

where P_{tx} and P_{rx} denote the transmitting and receiving power, respectively.

3.2. Problem Formulation. In this work, we hope to optimize two metrics, i.e., the tasks' execution time and the MD's energy consumption. Because the MD and MEC servers process tasks in parallel, the time metric can be given by

$$T(\mathbf{x}, \mathbf{y}, \psi_{\mathrm{MD}}) = \max\left\{T_{\mathrm{MD}}(\mathbf{x}, \psi_{\mathrm{MD}}), \max_{n\in\mathcal{N}}\{T_n(\mathbf{y})\}\right\}. \quad (7)$$

The energy consumption metric can be given by

$$E(\mathbf{x}, \mathbf{y}, \psi_{\mathrm{MD}}) = E_{\mathrm{MD}}(\mathbf{x}, \psi_{\mathrm{MD}}) + E_{\mathrm{MEC}}(\mathbf{y}). \quad (8)$$

However, these two objectives are coupled by x_m, $\{y_{m,n}\}$, and ψ_{MD}, so they cannot be optimized independently and contemporaneously. To investigate the tradeoff between them, we construct a unified objective function (or system utility) as

$$U(\mathbf{x}, \mathbf{y}, \psi_{\mathrm{MD}}) = \xi_{\mathrm{T}} T(\mathbf{x}, \mathbf{y}, \psi_{\mathrm{MD}}) + \xi_{\mathrm{E}} E(\mathbf{x}, \mathbf{y}, \psi_{\mathrm{MD}}), \quad (9)$$

where $\xi_{\mathrm{T}}, \xi_{\mathrm{E}} \in [0, 1]$ denote the weighting parameters of execution time and energy consumption for the MD. In this way, the latency and energy metrics can be taken into the decision making at the same time, while ξ_{T} and ξ_{E} reflect the relative importance between them. If the MD is running some application that is sensitive to the latency, it can set $\xi_{\mathrm{T}} \longrightarrow 1$ and $\xi_{\mathrm{E}} \longrightarrow 0$ when making decision. If the MD is at a low battery state and cares more about the energy consumption, it can set $\xi_{\mathrm{T}} \longrightarrow 0$ and $\xi_{\mathrm{E}} \longrightarrow 1$ when making decision. Such a weighted sum approach has been extensively used for modeling similar multiobjective optimization problems [2].

In conclusion, the optimization problem is formulated as

$$\mathbf{P}1: \min U(\mathbf{x}, \mathbf{y}, \psi_{\mathrm{MD}}). \quad (10)$$

s.t. constraint (2),

$$\begin{aligned} &\text{var: } \psi_{\mathrm{MD}} \in \Psi, \\ &x_m, y_{m,n} \in \{0, 1\}, \quad \forall m \in \mathcal{M}, \forall n \in \mathcal{N}. \end{aligned} \quad (11)$$

The problem **P**1 is essentially a mixed-integer nonlinear programming, which is known as NP-hard. Furthermore, this problem is also a combinatorial optimization, in which the global optimal solution consists of decisions for each computation task and the MD's CPU. Since there is no computationally efficient way to get the exact optimal solution, we propose to develop a fast polynomial approximated algorithm that solves problem **P**1 based on the Markov approximation framework [6].

4. Markov Approximation and Algorithm Design

Markov approximation is a recently proposed technique for solving combinatorial network optimization problems [6]. Generally, this framework is consisted of two steps: log-sum-exp approximation and constructing problem-specific Markov chains that yield efficient parallel implementation for solving problem approximately. The proofs on optimality and convergence for Markov approximation have been presented in [6].

4.1. Log-Sum-Exp Approximation. Let $f = \{\mathbf{x}, \mathbf{y}, \psi_{\mathrm{MD}}\} \in \mathcal{F}$ be a feasible solution to problem **P**1, where $\mathcal{F}$ denotes the set of all feasible solutions that satisfy the constraints (2). Furthermore, we denote utility U_f as the system's objective function corresponding to a given configuration f, so problem **P**1 can be represented by $\min_{f\in\mathcal{F}} U_f$. Therefore, the equivalent minimum weight independent set (MWIS) problem of **P**1 is

$$\begin{aligned} \min_{\mathbf{p}\geq 0} \quad & \sum_{f\in\mathcal{F}} p_f U_f, \\ \text{s.t.} \quad & \sum_{f\in\mathcal{F}} p_f = 1, \end{aligned} \quad (12)$$

where the probability p_f indicates the percentage of time that the system is in configuration f. Regarding U_f as the weight of f, the problem is to search for a minimum weighted configuration. Following the Markov approximation framework [6], the log-sum-exp approximation of $\min_{f\in\mathcal{F}} U_f$ yields

$$U_{\min} \approx -\frac{1}{\beta}\log\left(\sum_{f\in\mathcal{F}} \exp\left(-\beta U_f\right)\right), \quad (13)$$

where β is a positive constant that affects the approximation performance. Let $|\mathcal{F}|$ be the size of set $\mathcal{F}$, the approximation accuracy is known as follows:

$$\min_{f\in\mathcal{F}} U_f - \frac{1}{\beta}\log|\mathcal{F}| \leq -\frac{1}{\beta}\log\left(\sum_{f\in\mathcal{F}} \exp\left(-\beta U_f\right)\right) \leq \min_{f\in\mathcal{F}} U_f. \quad (14)$$

It is clear that, as $\beta \longrightarrow \infty$, the approximation gap approaches 0 and thus the approximation becomes exact.

According to [6], the log-sum-exp approximation in (13) is equivalent to solve the following problem **P**2:

$$\begin{aligned} \mathbf{P}2: \min_{\mathbf{p}\geq 0} \quad & \sum_{f\in\mathcal{F}} p_f U_f + \frac{1}{\beta}\sum_{f\in\mathcal{F}} p_f \log\, p_f, \\ \text{s.t.} \quad & \sum_{f\in\mathcal{F}} p_f = 1. \end{aligned} \quad (15)$$

Since **P**2 is a convex problem, we can solve the Karush–Kuhn–Tucker (KKT) conditions [20] and obtain the optimal solution as

$$p_f^* = \frac{\exp\left(-\beta U_f\right)}{\sum_{f'\in\mathcal{F}} \exp\left(-\beta U_{f'}\right)}, \quad \forall f \in \mathcal{F}. \quad (16)$$

However, it is difficult to solve problem **P**2 directly because this requires complete information on $\mathcal{F}$ which is typically unknown due to the large solution space. However, if we can sample the configuration space $\mathcal{F}$ from the distribution p_f^* in (16), i.e., time-sharing among different configurations f according to their portions p_f^*, we actually solve problem **P**2 and thus problem **P**1 approximately [7]. Toward this, the key is to design a problem-specific Markov chain, which models feasible configurations as states,

achieves stationary distribution p_f^*, and allows parallel construction among the $|\mathcal{M}|$ tasks.

4.2. Markov Chain Design. It has been proved that there exists at least one continuous-time time-reversible ergodic Markov chain with stationary distribution p_f^* in (16) and [6]. To construct such a time-reversible Markov chain, we let $f, f' \in \mathcal{F}$ be two states of the Markov chain and use $q_{f,f'}$ as the transition rate from state f to f'. It suffices to design $q_{f,f'}$ to ensure the following two conditions: (a) any two states are reachable from each other, and (b) the detailed balance equation is satisfied, i.e., $p_f^* q_{f,f'} = p_{f'}^* q_{f',f}, \forall f, f' \in \mathcal{F}$. The above two sufficient requirements allow a large space to design a Markov chain in terms of the state-space structure and transition-matrix design.

First, the transition rate between any two states can be set to zero, if they are still reachable from any other states. That is, we only allow direct links between two states that can be reached by performing only one task migration.

Second, for two assignments with direct transitions, e.g., f and f', we design the transition rate between them as

$$q_{f,f'} = \frac{1}{|\mathcal{M}|\gamma} \frac{\exp\left(-\beta U_{f'}\right)}{\max\left\{\exp\left(-\beta U_{f'}\right), \exp\left(-\beta U_f\right)\right\}}, \quad (17)$$

where $\gamma > 0$ is a constant denoting the mean time of state transition [6].

4.3. Markov Approximation-Based Algorithm. According to Markov approximation [6], in our system, a configuration f consists of $|\mathcal{M}|$ computation tasks using one of its local configurations. If all tasks run individual continuous-time clocks and wait for performance-dependent amounts of time before switching their local configurations, we can implement the target Markov chain such that transitions only happen between two configurations f and f' when they differ from each other by only one task's local configuration.

The implementation based on our designed Markov chain is shown in Algorithm 1. This algorithm is explained as follows. The MD initializes a dedicated thread for each of its computation tasks. At first, each thread randomly selects a feasible target device that satisfies constraint (2) for $m \in \mathcal{M}$. Besides, the CPU frequency ψ_{MD} is also randomly picked from the candidate set Ψ. In stage 1, each thread is associated with an exponentially distributed random number with a mean equal to γ and counts down according to this number. In stage 2, when the timer of task m expires, the dedicated thread first sends RESET signals to other threads for notifying them the upcoming potential transition f' and then randomly generates a new configuration f' on task assignment and frequency scaling. The thread m will transit to f' with the probability $p_{f,f'}$ or stay at the current configuration f with the probability $1 - p_{f,f'}$. In stage 3, when the dedicated thread serving for a task receives a RESET signal, it terminates its current countdown process and then transits to stage 1 again. Due to the properties of the underlying Markov chain, this algorithm will converge to near-optimal configuration in probability after a sufficient number of time periods.

According to $p_{f,f'}$ in Algorithm 1, we have two typical transition scenarios as follows. First, if $U_f \geq U_{f'}$, then $q_{f,f'} = 1$. Second, if $U_f < U_{f'}$ and $\beta \longrightarrow \infty$, then $q_{f,f'} \longrightarrow 0$. Therefore, the design of $p_{f,f'}$ in (17) is likely to lead the system to a configuration that minimizes the objective in problem **P**1.

Theorem 1. *Algorithm 1 realizes a time-reversible Markov chain with the stationary distribution shown in (16).*

Proof. From the construction rule for the state-space structure of a Markov chain, we can know that all configurations can reach each other within a finite number of transitions, so the generated Markov chain is irreducible. Moreover, it is a finite-state ergodic Markov chain with a unique stationary distribution. Now we show that the stationary distribution of this Markov chain is exactly (16).

In Algorithm 1, the transition probability of task m from f to f' is $p_{f,f'}$. Then from a system-wide view, the probability of transition due to the selection of a single task m from the task set $\mathcal{M}$ is $p_{f,f'}/|\mathcal{M}|$. Furthermore, because task assignment is activated according to the countdown timer mechanism with a rate of γ, it follows that the transition rate from state f to f' is

$$q_{f,f'} = \frac{p_{f,f'}}{|\mathcal{M}|\gamma} = \frac{1}{|\mathcal{M}|\gamma} \frac{\exp\left(-\beta U_{f'}\right)}{\max\left\{\exp\left(-\beta U_{f'}\right), \exp\left(-\beta U_f\right)\right\}}. \quad (18)$$

Using (16) and (18), we can obtain that $p_f^* q_{f,f'} = p_{f'}^* q_{f',f}, \forall f, f' \in \mathcal{F}$; that is, the detailed balance equations hold. Finally, we know that the constructed Markov chain is time-reversible, and its stationary distribution is exactly (16) according to Theorem 1.3 and Theorem 1.14 in [21]. □

4.4. Performance Guarantee. In practice, it is possible that we can obtain only an inaccurate measurement or estimate of U_f for any $f \in \mathcal{F}$, due to imprecise measurements or local estimates [8]. As a result, the perturbed Markov chain may not converge to the desired stationary distribution p_f^*, but a suboptimal steady-state distribution. This observation motivates us to study on the performance gap in the presence of the perturbation errors.

For each configuration $f \in \mathcal{F}$ with U_f, we assume that the corresponding perturbation error belongs to the bounded region $[-\Theta_f, \Theta_f]$, where Θ_f is the inaccuracy bound. Then, the perturbed U_f takes only one from the $2n_f + 1$ discrete values $[U_f - \Theta_f, \ldots, U_f - (1/n_f)\Theta_f, U_f, U_f + (1/n_f)\Theta_f, \ldots, U_f + \Theta_f]$, where n_f is a positive constant. Moreover, with probability $\delta_{(f,j)}$, the perturbed U_f takes the value $U_f + (1/n_f)\Theta_f$, $\forall j \in \{-n_f, \ldots, n_f\}$ and $\sum_{j=-n_f}^{n_f} \delta_{(f,j)} = 1$.

(1) The following procedures execute on the MD.
(2) **procedure** *Initialization*
(3) **for** $m \in \mathcal{M}$ **do**
(4) $x_m \longleftarrow 0$, and $y_{m,n} \longleftarrow 0$, $\forall n \in \mathcal{N}$
(5) Randomly assign m to the MD (i.e., $x_m \longleftarrow 1$) or a MEC server $n \in \mathcal{N}$ (i.e., $y_{m,n} \longleftarrow 1$)
(6) **end for**
(7) Randomly pick a feasible ψ_{MD} from Ψ
(8) Transit to *STAGE 1*
(9) **end procedure**
(10) **procedure** *Stage 1: Wait (m)*
(11) Generate an exponentially distributed timer t_m with mean equal to γ
(12) Begin counting down by t_m
(13) **end procedure**
(14) **procedure** *Stage 2: Hop (m)*
(15) **If** t_m expires **then**
(16) Send **RESET** signals to m', $\forall m' \in \mathcal{M} \backslash \{m\}$.
(17) Generate a new configuration f' by randomly selecting another feasible device for m
(18) **if** the target device is the MD **then**
(19) $x_m \longleftarrow 1$, and $y_{m,n} \longleftarrow 0$, $\forall n \in \mathcal{N}$
(20) Randomly pick another feasible ψ_{MD}' from Ψ
(21) else **if** the target device is a MEC server n' **then**
(22) $x_m \longleftarrow 0$, $y_{m,n'} \longleftarrow 1$, and $y_{m,n} \longleftarrow 0$, $\forall n \in \mathcal{N} \backslash \{n'\}$
(23) **end if**
(24) Migrate to the new configuration f' by the probability of $p_{f,f'} = ((\exp(-\beta U_{f'}))/(\max\{\exp(-\beta U_{f'}), \exp(-\beta U_f)\}))$
(25) **end if**
(26) **end procedure**
(27) **procedure** *Stage 3: Reset (m)*
(28) **if** t_m does not expire **and** m receives a **RESET** message **then**
(29) Terminate its current countdown timer
(30) Transit to *STAGE 1*
(31) **end if**
(32) **end procedure**

Algorithm 1: The Markov approximation-based algorithm.

Based on the analysis on perturbation errors in [22], we have the following results.

Theorem 2(a). *The stationary distribution of the perturbed Markov chain is*

$$\overline{p}_f = \frac{\varphi_f \exp\left(-\beta U_f\right)}{\sum_{f' \in \mathcal{F}} \varphi_{f'} \exp\left(-\beta U_{f'}\right)}, \tag{19}$$

where $\varphi_f = \sum_{j=-n_f}^{n_f} \delta_{(f,j)} \exp(\beta((j\Theta_f)/n_f))$.

Theorem 2(b). *The optimality gaps without and with perturbation errors are shown as follows:*

$$0 \le U_{\mathrm{avg}} - U_{\min} \le \frac{\log|\Psi| + |\mathcal{M}|\log(|\mathcal{N}| + 1)}{\beta}, \tag{20}$$

$$0 \le \overline{U}_{\mathrm{avg}} - U_{\min} \le \frac{\log|\Psi| + |\mathcal{M}|\log(|\mathcal{N}| + 1)}{\beta} + \Theta_{\max}, \tag{21}$$

where $U_{min} = min_{f \in \mathcal{F}} U_f$ *is the optimal system utility for problem* **P1**, $U_{avg} = \sum_{f \in \mathcal{F}} p_f^* U_f$ *is the expected system utility with the original perfect Markov chain,* $\overline{U}_{avg} = \sum_{f \in \mathcal{F}} \overline{p}_f U_f$ *is the expected system utility with the perturbed Markov chain, and* $\Theta_{max} = max_{f \in \mathcal{F}} \Theta_f$ *is the maximum perturbation error.*

Proof (a). We consider an extended Markov chain in which each state f is expanded to $2n_f + 1$ states (f, j), $\forall j = -n_f, \ldots, 0, 1, \ldots, n_f$ with the following transition rates:

$$\overline{q}_{(f,j),(f',j')} = \frac{\delta_{(f',j')}}{|\mathcal{M}|\gamma} \frac{\exp\left(-\beta\left(U_{f'} + \left(j'/n_{f'}\right)\Theta_{f'}\right)\right)}{\max\left\{\exp\left(-\beta\left(U_{f'} + \left(j'/n_{f'}\right)\Theta_{f'}\right)\right), \exp\left(-\beta\left(U_f + \left(j/n_f\right)\Theta_f\right)\right)\right\}}, \tag{22}$$

where $\delta_{(f',j')}$, $\forall j' = -n_f', \ldots, 0, 1, \ldots, n_f'$ is the probability scale on expanded states and $\sum_{j=-n_{f'}}^{n_{f'}} \delta_{(f',j')} = 1$. This extended Markov chain has a unique stationary distribution since it is irreducible and only has a finite number of states. Denote by $p_{(f,j)}$, $\forall j = -n_f, \ldots, 0, 1, \ldots, n_f$, $f \in \mathcal{F}$ is the stationary distribution of the states in this extended Markov chain. According to (22) and the detailed balance equations $p_{(f,j)}\overline{q}_{(f,j),(f',j')} = p_{(f',j')}\overline{q}_{(f',j'),(f,j)}$, we have:

$$\frac{p_{(f,0)}}{\delta_{(f,0)}\exp\left(-\beta U_f\right)} = \frac{p_{(f',j')}}{\delta_{(f',j')}\exp\left(-\beta\left(U_{f'} + \left(j'/n_{f'}\right)\Theta_{f'}\right)\right)} = \text{constant}, \tag{23}$$

where $(f, 0)$ refers to state f with no error.

Using $\sum_{f\in\mathcal{F}}\sum_{j=-n_f}^{n_f} p_{(f,j)} = 1$, it follows that

$$p_{(f,j)} = \frac{\delta_{(f,j)}\exp\left(-\beta\left(U_f + \left(j/n_f\right)\Theta_f\right)\right)}{\sum_{f'\in\mathcal{F}}\sum_{j'=-n_{f'}}^{n_{f'}}\delta_{(f',j')}\exp\left(-\beta\left(U_{f'} + \left(j'/n_{f'}\right)\Theta_{f'}\right)\right)}. \tag{24}$$

Denote by $\varphi_f = \sum_{j=-n_f}^{n_f}\delta_{(f,j)}\exp\left(\beta\left(\left(j\Theta_f\right)/n_f\right)\right) = \sum_{j=-n_f}^{n_f}\delta_{(f,j)}\exp\left(-\beta\left(\left(j\Theta_f\right)/n_f\right)\right)$. Using (24), we finally have

$$\overline{p}_f = \sum_{j=-n_f}^{n_f} p_{(f,j)} = \frac{\left[\sum_{j=-n_f}^{n_f}\delta_{(f,j)}\exp\left(-\beta\left(j/n_f\right)\Theta_f\right)\right]\exp\left(-\beta U_f\right)}{\sum_{f'\in\mathcal{F}}\left[\sum_{j'=-n_{f'}}^{n_{f'}}\delta_{(f',j')}\exp\left(-\beta\left(j'/n_{f'}\right)\Theta_{f'}\right)\right]\exp\left(-\beta U_{f'}\right)}, = \frac{\varphi_f\exp\left(-\beta U_f\right)}{\sum_{f'\in\mathcal{F}}\varphi_{f'}\exp\left(-\beta U_{f'}\right)}. \tag{25}$$

□

Proof (b). First, we prove the bounds on optimality gap for the original Markov chain. Let $f_{\min} \in \operatorname{argmin}_{f\in\mathcal{F}} U_f$, and by the Dirac distribution, we have

$$\widetilde{p}_f = \begin{cases} 1, & \text{if } f = f_{\min}, \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

Since p_f^* in (16) is the optimal distribution for problem **P**2, we have

$$\sum_{f\in\mathcal{F}} p_f^* U_f + \frac{1}{\beta}\sum_{f\in\mathcal{F}} p_f^* \log\, p_f^* \le \sum_{f\in\mathcal{F}}\widetilde{p}_f U_f + \frac{1}{\beta}\sum_{f\in\mathcal{F}}\widetilde{p}_f \log\, \widetilde{p}_f = U_{\min}. \tag{27}$$

With Jensen's inequality, we can obtain

$$\sum_{f\in\mathcal{F}} p_f^* \log\, p_f^* = -\sum_{f\in\mathcal{F}} p_f^* \log\frac{1}{p_f^*} \ge -\log\left(\sum_{f\in\mathcal{F}} p_f^*\frac{1}{p_f^*}\right) = -\log|\mathcal{F}|. \tag{28}$$

Combining (27) and (28) yields

$$U_{\text{avg}} = \sum_{f\in\mathcal{F}} p_f^* U_f \ge \sum_{f\in\mathcal{F}} p_f^* U_{\min} = U_{\min} \ge U_{\text{avg}} + \frac{1}{\beta}\sum_{f\in\mathcal{F}} p_f^* \log\, p_f^* \ge U_{\text{avg}} - \frac{1}{\beta}\log|\mathcal{F}|. \tag{29}$$

Thus,

$$0 \le U_{\text{avg}} - U_{\min} \le \frac{1}{\beta}\log|\mathcal{F}|. \tag{30}$$

Next, we prove the bounds on optimality gap for the perturbed Markov chain. By part (a), probability distribution $\overline{p}$ can be regarded as the optimal solution to problem **P**1 by replacing the utility function U_f by $U_{f'} = U_f - (\log\varphi_f/\beta)$. From (20), we have

$$\sum_{f'\in\mathcal{F}}\overline{p}_{f'}U_{f'} - \min_{f'\in\mathcal{F}} U_{f'} \le \frac{1}{\beta}\log|\mathcal{F}|. \tag{31}$$

Plugging the values of $U_{f'}$ into (31), we obtain

$$\sum_{f\in\mathcal{F}}\overline{p}_f\left(U_f - \frac{\log\varphi_f}{\beta}\right) - \min_{f\in\mathcal{F}}\left(U_f - \frac{\log\varphi_f}{\beta}\right) \le \frac{1}{\beta}\log|\mathcal{F}|. \tag{32}$$

According to the definition of φ_f, it is easy to know that

$$\exp\left(-\beta\Theta_f\right) \le \varphi_f \le \exp\left(\beta\Theta_f\right), \quad \forall f \in \mathcal{F}. \tag{33}$$

Therefore,

$$-\Theta_f \le -\frac{\log\varphi_f}{\beta} \le \Theta_f. \tag{34}$$

Combining (32) and (34) yields

Table 1: Simulation parameters.

Parameter	Definition	Default value
$\lvert\mathcal{M}\rvert$	Total number of computation tasks	100
$\lvert\mathcal{N}\rvert$	Total number of MEC servers	4
a_m	Required number of CPU cycles for processing task m	$330 \times b_m$ cycles
b_m	Size of the input data of task m	0.5 MB
c_m	Size of the output data of task m	$0.2 \times b_m$
θ	Power parameter in CPU energy consumption model	3
(Υ_1, Υ_2)	Power parameter in CPU energy consumption model	$(1.25 \times 10^{-26}, 0)$
$[\psi^{\min}, \psi^{\max}]$	CPU frequency range of the MD	$[2.0, 8.0] \times 10^8$ cycles/sec
$\lvert\Psi\rvert$	Total number of CPU DVFS levels	7
ψ_n	Service rates provided by MEC servers	$\{2, 2.2, 2.4, 2.6\} \times 10^9$ cycles/sec
r_n^{UL}	Uplink data rate between the MD and MEC server n	10 ~ 20 Mbps
r_n^{DL}	Downlink data rate between the MD and MEC server n	10 ~ 20 Mbps
P_{tx}	Transmitting power of MD	1.26 Watt
P_{rx}	Receiving power of MD	1.18 Watt
ξ_{T}	Scalar weight of tasks' execution time	0.5
ξ_{E}	Scalar weight of the MD's energy consumption	$(1 - \xi_{\mathrm{T}})$
β	Positive constant that controls approximation accuracy	50
γ	Mean value for the countdown timer	200

$$\begin{aligned} U_{\min} &= \min_{f\in\mathcal{F}} U_f \geq \min_{f\in\mathcal{F}}\left(U_f - \frac{\log\varphi_f}{\beta}\right) \\ &\geq \sum_{f\in\mathcal{F}} \overline{p}_f\left(U_f - \frac{\log\varphi_f}{\beta}\right) - \frac{1}{\beta}\log|\mathcal{F}| \geq \sum_{f\in\mathcal{F}} \overline{p}_f U_f \\ &\quad - \Theta_f - \frac{1}{\beta}\log|\mathcal{F}|. \end{aligned} \tag{35}$$

For $\overline{U}_{\mathrm{avg}}$, we have

$$\begin{aligned} \overline{U}_{\mathrm{avg}} &= \sum_{f\in\mathcal{F}} \overline{p}_f U_f \geq \sum_{f\in\mathcal{F}} \overline{p}_f U_{\min} = U_{\min} \geq \overline{U}_{\mathrm{avg}} - \Theta_f \\ &\quad - \frac{1}{\beta}\log|\mathcal{F}|. \end{aligned} \tag{36}$$

Thus,

$$0 \leq \overline{U}_{\mathrm{avg}} - U_{\min} \leq \frac{1}{\beta}\log|\mathcal{F}| + \Theta_f \leq \frac{1}{\beta}\log|\mathcal{F}| + \Theta_{\max}. \tag{37}$$

In general, given $|\mathcal{M}|$ computation tasks, $|\mathcal{N}|$ MEC servers, and the MD's local CPU with $|\Psi|$ frequency levels, the total number of available configurations is bounded by

$$|\mathcal{F}| \leq |\Psi|\,(|\mathcal{N}| + 1)^{|\mathcal{M}|}. \tag{38}$$

Finally, by plugging (38) into (30) and (37), we prove part (b) of Theorem 2. □

5. Simulation Results and Analysis

5.1. Simulation Setup. In this section, we evaluate the performance of our Markov approximation-based algorithm (denoted by MA) by simulations based on parameters from references [5, 23, 24]. Table 1 lists the default values for simulation parameters in our experiments, if not otherwise specified.

In order to evaluate the MA algorithm performance, we compare it with the following three baseline approaches:

(1) Exhaustive Search (ES): the optimal solution is found by enumerating all possible configurations, which leads to high computation complexity and only feasible for small-sized problems.

(2) Random Assignment (RA): the solutions are obtained by randomly assigning computation tasks and setting CPU frequency.

(3) Local Processing (LP): all computation tasks are processed by the CPU of MD, and the frequency is optimal to the assignment.

To eliminate the influence of randomness, the experiments of MA, RA, and LP are independently carried out multiple times for each given setting. For clear comparison, we choose to show either the average of results or the range of results in the figures. Similar to [7], the convergence time of MA is defined as the time at which the difference between two consecutive values does not exceed 0.5%.

5.2. Performance Evaluation

5.2.1. Convergence of the MA Algorithm. At first, we examine the convergence of MA under different settings. Figure 1 shows the convergence of our MA approach in a low-load scenario with the number of tasks $|\mathcal{M}| = 10$. The results show that MA converges quickly to near optimal after 50 iterations, independent of problem size. The computation complexity is far less than that of enumerating all 7×5^{10} combinatorial possibilities in the ES approach. We further investigate the performance of MA under different ratios of

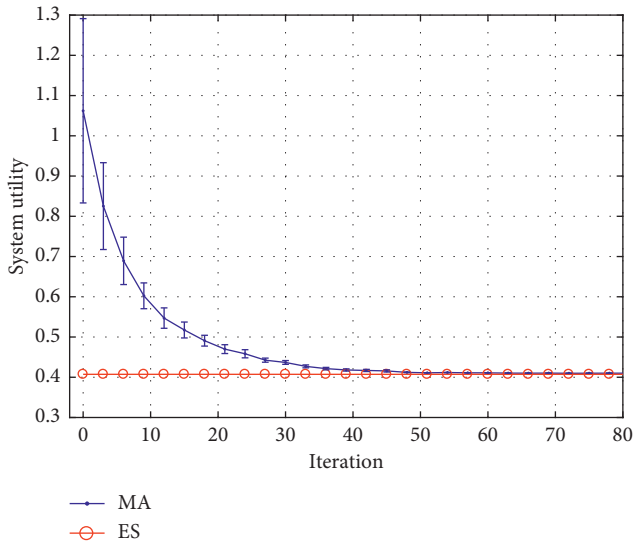

Figure 1: Convergence of MA algorithm in a low-load scenario.

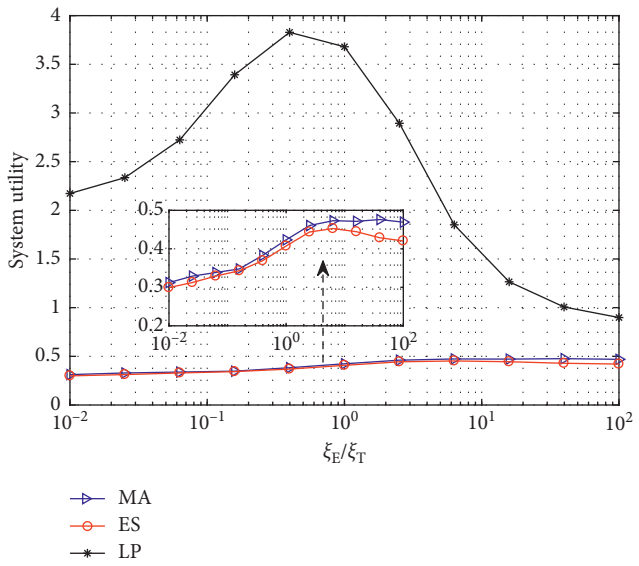

Figure 2: Impact of ξ_E/ξ_T on convergence of MA algorithm.

ξ_E and ξ_T. The value of ξ_E is chosen such that $\xi_E = 10^{\omega} \times \xi_T$, where $\omega \in \{0, \pm 0.4, \pm 0.8, \pm 1.2, \pm 1.6, \pm 2\}$. Figure 2 shows that changing these two scalar weights has little impact on the convergence of MA and confirms the existence of approximation gap in (14).

As the problem size becomes even larger (e.g., $|\mathcal{M}| = 100$), ES will take considerable time (e.g., several hours or even more) to search for the optimal solution from 7×5^{100} options, while MA converges after about 1100 iterations. In such a high-load scenario, we compare MA with ES, as well as the best results obtained by RA and LP from multiple independent runs. As shown in Figure 3, no matter how many MEC servers are used (i.e, $|\mathcal{N}| = 2$ or 4), MA can gradually converge to optimal as iteration grows. It is obvious that more MEC servers can provide more offloading options and further improve the energy and latency

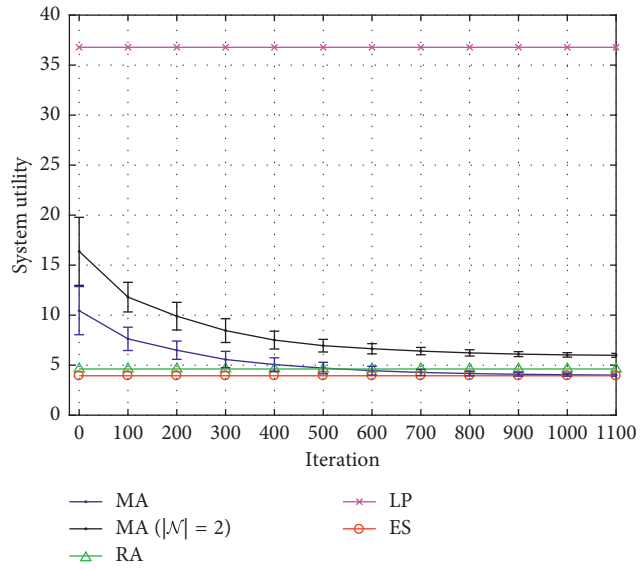

Figure 3: System utilities of different algorithms.

performance. Although the best result of RA in Figure 3 approaches the optimal utility found by ES, RA is of no practical value because it is generally not able to guarantee deterministic outputs. Among the four approaches, LP has the largest utility value, indicating that MEC offloading is essential for shortening execution time and reducing energy consumption in such a case.

5.2.2. Impact of Link Rates. In Figure 4, we compare the performance of algorithms with regard to different data transmission conditions. The link rates are controlled in three distinct ranges, namely, low rate (0.5 Mbps~1 Mbps), medium rate (2 Mbps~10 Mbps), and high rate (20 Mbps~50 Mbps). From Figure 4, the MA algorithm can always converge, and its convergence time decreases as the link rates increase. We can also observe from Figure 4(a) that MA converges to the same result as that of LP, no matter how many MEC servers are used. That is because when the link rates are low, almost all tasks are processed locally by the MD's CPU since the offloading costs on energy and latency are much higher than those of local computation. As the link rates increase, the offloading costs decreases because of lower transmission duration, making it more preferable to offload tasks to MEC servers by using our MA algorithm (as shown in Figures 4(b) and 4(c)).

5.2.3. Impact of Parameter β. Figure 5 demonstrates the impact of β on system utility and convergence time. The dashed arrows with different colors in Figure 5 indicate the iterative time when the convergence is achieved by MA with a given parameter β. It is shown that as β increases from 4 to 50, and MA will eventually obtain an approximate utility closer to the optimal value obtained by ES, but the corresponding convergence time will become longer. This allows us to make flexible choices among various tradeoff points between system optimization and convergence time. These

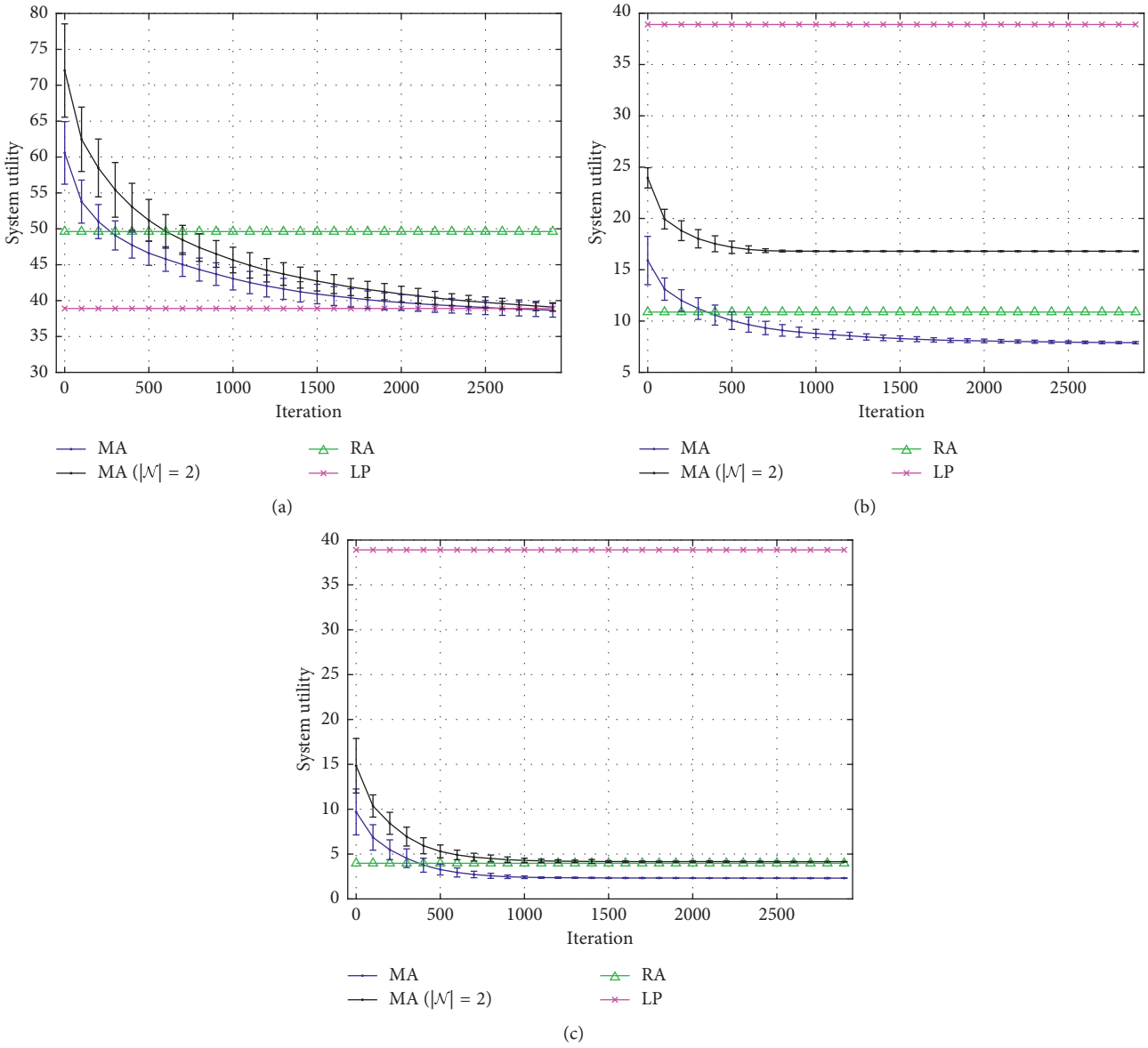

Figure 4: Algorithm comparison under different link rates. (a) 0.5 Mbps~1 Mbps. (b) 2 Mbps~10 Mbps. (c) 20 Mbps~50 Mbps.

observations in Figure 5 are consistent with (20) in Theorem 2.

5.2.4. Impact of Perturbation Errors. As mentioned in Section 4.4, in practice, we usually obtain inaccurate U_f due to estimation errors. To verify (21) in Theorem 2, we carry out experiments to study the impact of perturbation errors on the performance of MA. To do this, we add different degrees of random errors ($\pm 10\% \sim \pm 50\%$, uniformly distributed) to r_n^{UL} and r_n^{DL} for all n. In the experiments, we let MA make all control decisions based on the perturbed link rates but use the actual values to calculate the objective utility. We find that MA can indeed converge to a suboptimal solution even when perturbation errors exist. To characterize the optimality gap, we choose the optimal result obtained by ES without perturbation errors as the baseline and use this baseline to examine the worst-case (i.e., maximum) utility achieved by MA in multiple independent runs. Figure 6 shows the difference ratios of utilities under different error rates. From the figure, we find that as the error rate varies from 10% to 50%, the optimal gap keeps increasing from 2.15% to 33.0%. These results corroborate Theorem 2 that the further the estimations are from the true values, the poorer the performance MA achieves. However, when the error rate is smaller than some threshold value (e.g., 20%), the optimal gap is very small.

5.2.5. Tradeoff between Latency and Energy. In Figure 7, we study the tradeoff between task execution time and mobile energy consumption in the MA algorithm. As we did in the case of Figure 2, totally 11 different combinations of ξ_{E} and ξ_{T} are chosen for comparison in Figure 7. For each curve in

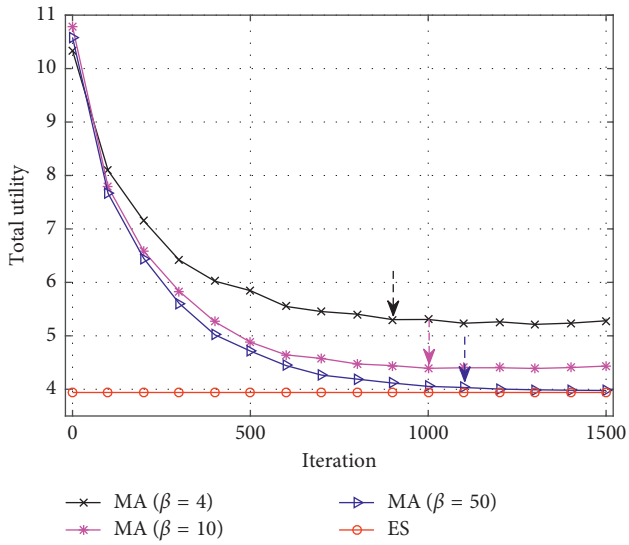

Figure 5: Impact of β on system utility and convergence time.

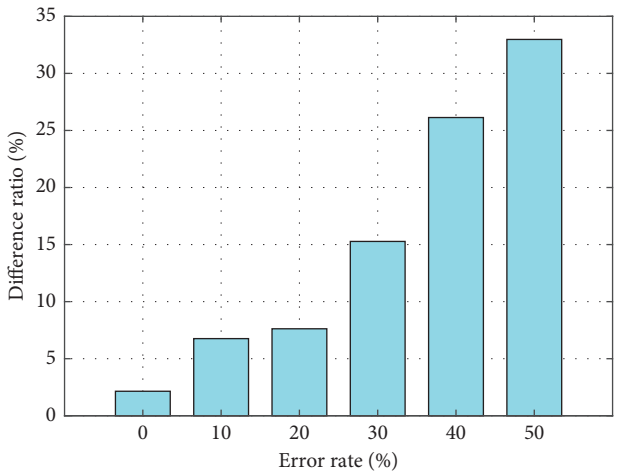

Figure 6: Impact of perturbation errors on optimality gap.

Figure 7, the markers are laid out from left to right in the order of ω. Correspondingly, the left-most marker corresponds to the condition $\omega = -2$, and the right-most marker corresponds to the condition $\omega = 2$. From this figure, it is feasible to tune the two scalar weights for achieving a desirable tradeoff between latency and energy. However, we can notice that when ω grows greater than some threshold (e.g., $\omega > 0$ when $|\mathcal{N}| = 2$), the decreasing trend of energy consumption becomes smooth while execution latency keeps increasing. Therefore, the values of ξ_E and ξ_T should be carefully chosen to achieve a satisfactory energy-latency tradeoff. On the other hand, increasing $|\mathcal{N}|$ leads to significant reductions in both latency and energy, as the curve shifts down to the bottom-left corner of the plot. This observation confirms again that it is beneficial to provide more than one MEC server to the MD for task offloading. Nevertheless, the performance improvement gradually diminishes when $|\mathcal{N}|$ becomes larger.

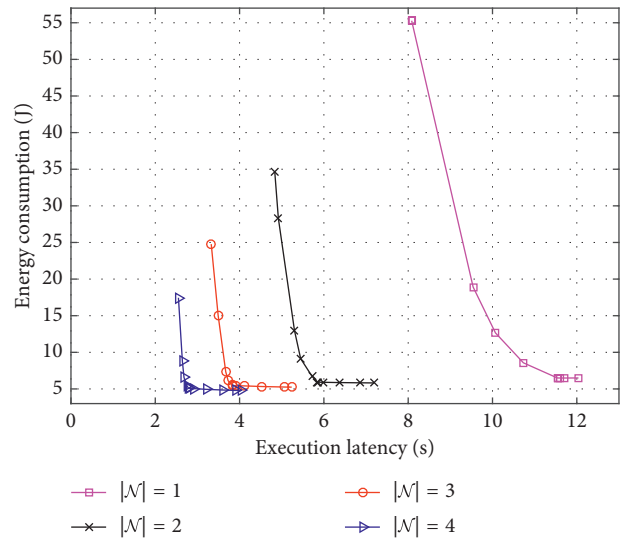

Figure 7: Energy-latency tradeoff under different values of ω and $|\mathcal{N}|$.

6. Conclusion

This paper addresses the computation offloading issue in a multiserver mobile edge computing scenario, where a MD can offload its computation tasks to multiple MEC servers. We formulate this problem as the joint optimization of task assignment and frequency scaling, aimed at minimizing the tradeoff between the MD's energy consumption and total tasks' execution time. In order to solve this NP-hard problem, a Markov approximation-based algorithm is devised to find a near-optimal solution, where the result has only a small and bounded gap with the optimal one. Through simulation experiments with many practical concerns, we verify that our algorithm can converge to near-optimal performance and ensure desirable robustness and scalability. Furthermore, this algorithm significantly outperforms other benchmark algorithms such as exhaustive search and local processing.

For the future work, we are going to examine our Markov approximation-based approach with multiple MDs to obtain more general conclusions. Further, we shall extend our approach to an online case, where the problem can be solved with dynamic arrival and leave of MDs. It is also worthwhile to evaluate the work in a real test-bed of mobile edge computing.

Data Availability

The experiment data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities of China under Grant 2016JBZ006 and CETC Joint Fund under Grant 6141B08020101.

References

[1] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi, "Using bandwidth data to make computation offloading decisions," in *Proceedings of 2008 IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–8, Miami, FL, USA, April 2008.

[2] W. Fang, Y. Li, H. Zhang, N. Xiong, J. Lai, and A. V. Vasilakos, "On the throughput-energy tradeoff for data transmission between cloud and mobile devices," *Information Sciences*, vol. 283, pp. 79–93, 2014.

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[4] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[5] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Adaptive computation scaling and task offloading in mobile edge computing," in *Proceedings of 2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, San Francisco, CA, USA, March 2017.

[6] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6301–6327, 2013.

[7] Z. Shao, H. Zhang, M. Chen, and K. Ramchandran, "Reverse-engineering bittorrent: a markov approximation perspective," in *Proceedings of 2012 IEEE INFOCOM*, pp. 2996–3000, Orlando, FL, USA, March 2012.

[8] Z. Shao, X. Jin, W. Jiang, M. Chen, and M. Chiang, "Intra-data-center traffic engineering with ensemble routing," in *Proceedings of 2013 IEEE INFOCOM*, pp. 2148–2156, Turin, Italy, April 2013.

[9] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[10] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proceedings of 2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1451–1455, Barcelona, Spain, July 2016.

[11] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1757–1771, 2016.

[12] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[13] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[14] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[15] G. Wu, J. Chen, W. Bao, X. Zhu, W. Xiao, and J. Wang, "Towards collaborative storage scheduling using alternating direction method of multipliers for mobile edge cloud," *Journal of Systems and Software*, vol. 134, pp. 29–43, 2017.

[16] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2015.

[17] J. Ren, G. Yu, Y. Cai, Y. He, and F. Qu, "Partial offloading for latency minimization in mobile-edge computing," in *Proceedings of GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, Singapore, December 2017.

[18] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proceedings of 2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, San Francisco, CA, USA, March 2017.

[19] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Dynamic speed scaling for energy minimization in delay-tolerant smartphone applications," in *Proceedings of IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 2292–2300, Toronto, Canada, April 2014.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[21] F. P. Kelly, *Reversibility and Stochastic Networks*, Cambridge University Press, New York, NY, USA, 2011.

[22] S. Zhang, Z. Shao, M. Chen, and L. Jiang, "Optimal distributed p2p streaming under node degree bounds," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 717–730, 2014.

[23] Y. Xiao, P. Savolainen, A. Karppanen, M. Siekkinen, and A. Ylä-Jääski, "Practical power modeling of data transmission over 802.11g for wireless applications," in *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pp. 75–84, New York, NY, USA, 2010.

[24] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, p. 4, Berkeley, CA, USA, 2010.