Computation Offloading Strategy Based on Improved Auction Model in Mobile Edge Computing (Postprint)

Authors: Sheng Jinfang, Teng Xiaoyu, Li Weimin, Wang Bin

Date: 2019-05-10T00:00:00+00:00

Abstract

With the rapid development of mobile Internet services, mobile applications such as augmented reality, virtual reality, and ultra-high-definition video are gradually gaining popularity, while IoT applications continue to emerge. The insufficient computational capability and battery endurance have become primary bottlenecks preventing smart terminal devices from successfully supporting these applications. To address this challenge, this paper proposes a computation offloading strategy based on an improved auction algorithm in a multi-user multi-mobile-edge-server scenario, which comprehensively considers both smart device performance and server resources. The strategy consists of two main phases: the offloading decision phase, which establishes the basis for offloading decisions by comprehensively considering factors including computational task size, computational requirements, server computing capacity, and network bandwidth; and the task scheduling phase, which proposes a task scheduling model based on the improved auction algorithm by comprehensively considering the time requirements of computational tasks and the computing performance of MEC servers. Experimental results demonstrate that the proposed computation offloading strategy can effectively reduce service latency, decrease energy consumption of smart devices, and improve user experience.

Full Text

Preamble

Computational Offloading Strategy Based on Improved Auction Model in Mobile Edge Computing

Sheng Jinfang, Teng Xiaoyu, Li Weimin, Wang Bin (School of Information Science & Engineering, Central South University, Changsha 410000, China)

Abstract: With the rapid development of mobile Internet services, applications such as augmented reality, virtual reality, and ultra-high-definition video have become increasingly popular, while IoT applications continue to emerge. However, the limited computing power and battery life of smart terminal devices have become major bottlenecks preventing these applications from being successfully supported. To address this challenge, this paper proposes a computation offloading strategy based on an improved auction algorithm in a multi-user, multi-MEC server scenario, comprehensively considering both device performance and server resources. The strategy consists of two main phases: (1) In the offloading decision phase, we establish the basis for offloading decisions by comprehensively considering factors such as task size, computational requirements, server computing capacity, and network bandwidth; (2) In the task scheduling phase, we propose a task scheduling model based on an improved auction algorithm by jointly considering the time requirements of computational tasks and the computing performance of MEC servers. Experimental results demonstrate that the proposed offloading strategy can effectively reduce service latency, decrease smart device energy consumption, and improve user experience.

Keywords: mobile edge computing; computation offloading; auction algorithm; failure compensation

0 Introduction

The rapid development of mobile Internet services—including augmented reality, virtual reality, online gaming, and ultra-high-definition video—along with the emergence of IoT applications such as crowdsensing, intelligent surveillance, and environmental monitoring, has placed increasingly high demands on mobile device service capabilities. These applications require substantial computing resources while consuming significant amounts of energy. However, due to physical size constraints, mobile smart devices have limited computing power and battery life, making it difficult to support these applications at satisfactory levels for users.

A primary solution to this problem is to offload computationally intensive tasks to nearby servers with more abundant resources—a method known as computation offloading [**?**] or cyber foraging [**?**]. This approach involves six steps: migration environment awareness (server discovery), task partitioning, migration decision, task uploading, remote execution, and result retrieval. Among these, task partitioning and migration decision are the two most critical components. Task partitioning focuses on fine-grained division based on application code, dividing applications according to methods, functions, or other criteria, as seen in MAUI [**?**], ThinkAir [**?**], and Phone2Cloud [**?**]. Migration decision represents the core issue in computation offloading technology, focusing on whether to offload computation, with decisions depending on factors such as time overhead and energy consumption. References [**?**, **?**] consider offloading decisions

from an energy perspective, while [?] examines server resources and bandwidth.

Computation offloading was first applied in Mobile Cloud Computing (MCC) [?, ?, ?], where mobile terminal tasks were offloaded to traditional cloud data centers for execution. However, the primary obstacle for traditional cloud computing—characterized by centralized computation and storage—is that the latency experienced when reaching remote cloud servers through Wide Area Networks (WAN) may offset the time saved by computation offloading. In many real-time mobile applications (e.g., online gaming, voice recognition, FaceTime), this can severely impact Quality of Experience (QoE). To address this limitation, researchers proposed bringing cloud servers closer to users, leading to the concept of cloudlet [?, ?], which enables computation offloading via Wi-Fi access to fixed servers [?, ?, ?]. Nevertheless, cloudlet servers have fixed locations and limited numbers, and the unavailability of fixed servers may restrict cloudlet applicability.

To overcome the limitations of MCC and cloudlet in computation offloading applications, the European organization TROPIC proposed providing cloud computing capabilities at small cell base stations. Subsequently, the European Telecommunications Standards Institute (ETSI) introduced the concept of Mobile Edge Computing (MEC) [?], combining MEC with small cell base station deployments to achieve true physical proximity between mobile devices and cloud servers providing computing services, thereby reducing application access latency.

As a key technology for future 5G communications, MEC' s primary application is computation offloading [?], and many scholars have made significant contributions to this research area. Reference [?] studied multi-user computation offloading in multi-channel wireless interference environments, using game theory to achieve effective channel allocation in a distributed manner. Reference [?] employed Orthogonal Frequency Division Multiple Access (OFDMA) for multi-user, multi-MEC server systems, using a bi-level optimization method to decouple the original NP-hard problem into lower-level problems seeking power and subcarrier allocation and upper-level task offloading problems. Reference [?] proposed a Heuristic Offloading Decision Algorithm (HODA), a semi-distributed approach that jointly optimizes offloading decisions and communication resources to maximize system utility. Reference [?] presented a task scheduling algorithm based on Lyapunov optimization theory, combining task offloading, device-base station association, and base station sleep scheduling to minimize overall energy consumption of devices and base stations. Reference [?] proposed an adaptive sequential task offloading method where mobile users make offloading decisions sequentially based on the current interference environment and available computing resources, achieving good results in reducing latency and energy consumption.

While these studies have significantly advanced computation offloading technology, each has its own focus. For instance, references [?, ?, ?] emphasize the impact of channel multiplexing and channel competition on offloading decisions

without considering the influence of server resource utilization on computation offloading performance. References [**?**, **?**, **?**] focus on MEC server resources and scheduling without considering how task characteristics (such as data volume and computational resource requirements) affect computation offloading. Addressing these gaps, this paper proposes a computation offloading strategy based on an improved auction algorithm. By jointly considering factors such as task size, computational requirements, server computing capacity, and network bandwidth, we establish the basis for offloading decisions. Additionally, by considering both task time requirements and MEC server computing performance, we propose a task scheduling model based on an improved auction algorithm. Experiments demonstrate that our offloading strategy can effectively reduce service latency, decrease smart device energy consumption, and improve user experience.

---

1 Improved Auction-Based Decision Model

To address the inability of mobile devices to meet the demands of computationally intensive applications, this paper adopts computation offloading as a solution, where mobile device tasks are offloaded to MEC servers. Computation offloading technology primarily addresses two questions: (1) Should a task be offloaded for execution? (2) When offloading is needed, which Virtual Machine (VM) should process the task? To answer these questions, we propose a computation offloading strategy based on an improved auction model, which consists of two phases:

a) **Offloading Decision Phase**: This phase jointly considers the time and energy consumption of tasks running locally versus being offloaded to determine whether task offloading is necessary.

b) **Task Scheduling Phase**: When tasks require offloading, we schedule them using an improved auction algorithm to execute on appropriate VMs, achieving global optimality—i.e., enabling MEC servers to process more tasks within a given time period. Additionally, we propose a bidding failure compensation model that ensures fairness in task scheduling by considering task deadline constraints through price compensation to buyers.

1.1 Offloading Decision Model

The primary criteria for determining whether to perform computation offloading are: (1) whether it reduces task execution time, and (2) whether it saves device energy. This paper proposes corresponding models for time and energy consumption for both local execution and offloaded execution.

1) Local Execution Time and Energy Model

When a task runs locally, the time required to process the task is:

$$T_L^n = \frac{C_n}{V_L}$$

where C_n represents the computational resources required by task n, measured in CPU cycles needed to execute the task (which can be obtained through code implementation, such as using the `get_cycle_count()` function in C++), and V_L is the local CPU execution rate.

The energy consumed when processing the task locally is:

$$E_L^n = P_L \cdot T_L^n$$

where P_L is the mobile device' s processing power.

2) Computation Offloading Time and Energy Model

When a task runs on an MEC server, the total time consumption consists of three parts: the transmission time for uploading the task to the server (T_{up}), the task execution time on the server (T_{exe}), and the time for downloading results back to the mobile device (T_{down}). Therefore, the total offloading time is:

$$T_{unload} = T_{up} + T_{exe} + T_{down}$$

Many studies [**?**, **?**, **?**, **?**, **?**] ignore the impact of result download time T_{down} on total offloading time because the output of most applications is much smaller than the input, making the download time negligible. Reference [**?**] provides a detailed analysis of this. Therefore, this paper defines offloading time as:

$$T_{unload} = T_{up} + T_{exe}$$

The transmission time required for task upload can be expressed as:

$$T_{up} = \frac{D_n}{W \log_2\left(1 + \frac{P_{up} \cdot Los}{N_0}\right)}$$

where D_n is the amount of data to be uploaded for the computation task; N_0 represents the Gaussian noise power within the channel; W is the channel bandwidth; P_{up} is the mobile device' s upload power; and Los is the channel gain. Based on the wireless interference model for cellular environments [**?**], Los is defined as a distance-based function $Los = d^{-\alpha}$, where α is set to 4.

The task execution time on the MEC server is:

$$T_{exe} = \frac{C_n}{V_{cloud}}$$

where V_{cloud} is the server CPU execution rate.

From equations (1), (2), and (7), we derive the task offloading time as:

$$T_{unload} = \frac{D_n}{W \log_2\left(1 + \frac{P_{up} \cdot Los}{N_0}\right)} + \frac{C_n}{V_{cloud}}$$

When a task runs on an MEC server, the energy consumed by the mobile device is only due to task uploading. Therefore, the energy consumption for computation offloading is:

$$E_{up} = P_{up} \cdot T_{up} = P_{up} \cdot \frac{D_n}{W \log_2\left(1 + \frac{P_{up} \cdot Los}{N_0}\right)}$$

3) Offloading Decision Criteria

Computation offloading improves user experience quality when the offloaded task execution time is less than local execution time and the energy consumption is less than local energy consumption. That is, offloading can be performed when both conditions (9) and (10) are satisfied:

$$T_{unload} < T_L^n$$
$$E_{up} < E_L^n$$

From equations (1), (2), and (7)-(10), we can derive conditions (11) and (12):

$$V_{cloud} > \frac{C_n}{\frac{C_n}{V_L} - \frac{D_n}{W \log_2\left(1 + \frac{P_{up} \cdot Los}{N_0}\right)}}$$

$$W \log_2\left(1 + \frac{P_{up} \cdot Los}{N_0}\right) > \frac{D_n}{\frac{C_n}{V_L} - \frac{C_n}{V_{cloud}}}$$

The meaning of equations (11) and (12) is that computation offloading can be performed when the computing capacity V_{cloud} provided by the MEC server exceeds the value on the right side of inequality (11), and the network bandwidth in the user' s environment satisfies inequality (12).

For a given task n, parameters including the mobile device' s computing rate V_L, the task' s computational resource requirement C_n, the task' s data size D_n, and the device' s upload power P_{up} are all obtainable. Los is a distance-based function that can be calculated once the distance between the mobile device and the base station is known. Therefore, the only unknown parameters in equations (11) and (12) are V_{cloud} and W. In this paper, we define N_0 as -100

dBm based on [**?**]. Thus, the right-hand side values of inequalities (11) and (12) can be calculated.

The first step in computation offloading is migration environment awareness, during which we can obtain the MEC server' s computing capacity V_{cloud} and the network bandwidth W of the environment. These values are then compared with the calculated right-hand side results. If conditions (11) and (12) are satisfied, computation offloading can be performed. In scenarios where multiple base stations are present in the user' s geographic environment, we calculate the time and energy consumption for offloading to each base station separately and select the base station that yields the minimum values for offloaded execution.

1.2 Improved Auction-Based Task Scheduling

When tasks are to be executed via offloading, two key questions arise: (1) Which VM should process each task? (2) When the number of offloaded tasks is large but server computing resources are limited, how should tasks be scheduled? To address these questions, this paper proposes a task scheduling strategy based on an improved auction algorithm.

1) Auction Model and Problem Formulation

The auction model is illustrated in [Figure 1: see original paper]. In this model, there are two roles: sellers and buyers. Sellers provide goods, while buyers participate in auctions for products that meet their needs, with the highest bidder winning.

In our model, the sellers are MEC servers, and the commodity being sold is rental time for computing resources. These computing resources are refined to idle VMs in MEC servers—i.e., sellers auction off rental time for idle VMs. The buyers are tasks that need to be offloaded. A buyer' s demand is the time required to run the task on a VM, denoted as T_{exe}. The bidding criterion is:

$$price = \frac{1}{T_{unload} - T_L}$$

where T_{unload} is the time consumed by offloading the task, and T_L is the time consumed by running the task locally. The difference represents time saved by offloading. This difference can also be viewed as the task' s deadline, representing the urgency of task completion. Since we aim to minimize offloading time T_{unload}, the closer a task is to its deadline, the more urgent it is. Therefore, we use the inverse of the difference: the less remaining time, the larger its inverse, indicating a higher willingness to pay.

In our auction model, we assume all VMs have identical computing capabilities, meaning the only factor users need to consider when renting a VM is the time required to complete their tasks.

We define a time slot duration as s. Both VM rental time and server leasing time are integer multiples of s. An auction is conducted every time slot s, with participants including both users who failed in the previous auction and newly arriving users in the current round.

When an auction begins, a VM' s rental time is $vm.t$, and a task' s required VM rental time is $task.t$. Only tasks satisfying $task.t < vm.t$ can participate in the bidding. The highest bidder gains VM usage rights, and the VM executes the task. When the rental time expires, the user' s access to the VM is revoked, making the VM available for the next round of competition.

If competition follows only these rules, in unfavorable situations, certain users may satisfy $task.t < vm.t$ for every VM auction but lose due to low bids, potentially participating in up to m rounds. In the worst case, this algorithm has a time complexity of $O(nm)$. To address this high complexity, we propose an improved auction algorithm.

2) Improved Auction Algorithm

Let the buyer set be $TASK = \{task_1, task_2, \dots, task_n\}$, where each $task_n$ has two attributes: $task_n.t$ representing the time needed to complete task n, and $task_n.p$ representing the task' s auction bid. Sort the elements in $TASK$ in ascending order based on $task.t$.

Let the seller set be $VM = \{vm_1, vm_2, \dots, vm_m\}$, where each vm_n has an attribute $vm_n.t$ representing VM n' s rental time. Sort the elements in VM in ascending order based on $vm.t$.

Define a set $CA = \{task_1, task_2, \dots, task_m\}$ to store tasks that meet auction conditions and participate in bidding. This set has a limited size, storing at most m elements (equal to the number of competing VMs). The insertion rule is: for each inserted element, use binary insertion sort to arrange the set elements in descending order of task bids. When the number of competing tasks exceeds m, only the top m highest-bidding elements are retained in CA; remaining tasks lack price advantages when competing for the same VM (as there are only m VMs) and thus fail the auction.

Iterate through set VM, auctioning VMs sequentially. When auctioning vm_1, iterate through set $TASK$ and add elements satisfying $task_n.t < vm_n.t$ to set CA until encountering a task that does not satisfy the condition. Then, assign vm_1 to the first task in CA. Continue this process for $vm_2, vm_3, \dots, vm_m$. Finally, m VMs are allocated to m users, while remaining users wait for the next round.

3) Auction Compensation Strategy

In our model, the bidding price is defined as $\frac{1}{T_{unload} - T_L}$, which considers task urgency but may introduce unfairness. Tasks that save more time through offloading lose price advantages, which is unfair in auctions. Therefore, we

propose a bidding failure compensation strategy that reprices tasks after auction failure:

$$price = \frac{1}{(T_{unload} - T_L) - s \cdot X}$$

where s is the duration of a time slot, and X is the number of times the task has failed in auctions. This formula has two implications: (1) Each failed auction round reduces the task' s remaining deadline by one time slot (since auctions occur per time slot); (2) Taking the derivative of price with respect to failure count X shows that as X increases, the derivative increases, meaning the price grows faster. Thus, the formula provides greater price compensation as auction failures accumulate.

The complete auction process is described in the following algorithm:

Algorithm: Multi-Task Multi-VM Single-Round Auction *Input:* Tasks and VMs participating in the auction
Output: Tasks that failed competition

1. Calculate the time and bid for each task to rent a VM, obtaining set $TASK$.
2. Set VM lease times according to task requirements, obtaining set VM.
3. Sort elements in $TASK$ in ascending order based on required VM lease time.
4. Sort elements in VM in ascending order based on rental time.
5. For each vm in set VM:
   - For each $task$ in set $TASK$:
     - If $task_n.t < vm_n.t$:
       - Add the task to set CA
       - Add the task to set EL (auction-failed task set)
   - Assign the first task in CA to VM vm
6. Add remaining tasks in CA to set EL

The core of our multi-task multi-VM auction model is the competition phase. For each VM auctioned, binary insertion sort is performed on set CA, which has complexity $O(\log m)$. With m auction rounds total, the algorithm' s time complexity is $O(m \log m)$—a significant improvement over the original auction algorithm' s $O(nm)$ complexity. As the number of mobile smart devices grows explosively, the number of tasks n participating in computation offloading will far exceed the number of VMs m. Furthermore, sorting both VM and task sets before competition ensures that requested times and VM rental times are similar, enabling VMs to be quickly recycled after task completion and allowing servers to process more tasks within the same timeframe.

2 Simulation Experiments and Results Analysis

This section presents MATLAB simulations of our proposed offloading strategy. The simulation scenario is defined as follows: 1-2 base stations, each with an MEC server running 30 VMs with computing capacity set to 10 GHz [**?**]. Communication bandwidth between users and base stations is 5 MHz [**?**], and each user has only one computation task at any given time.

We adopt a face recognition algorithm [**?**] as the computation task, with data sizes randomly distributed between 1,000 KB and 5,000 KB, and computational resource requirements set between 200 and 1,000 Megacycles. Based on surveys, mobile device computing capacity is randomly distributed between 1.5 GHz and 2.5 GHz. VM rental times are uniformly distributed between minimum and maximum values based on task requirements. All experimental results are averaged over 20 runs.

2.1 Energy Consumption Comparison Between Offloaded and Local Execution

This experiment uses a scenario with 1 base station and 30 VMs, with 20-70 tasks participating in offloading decisions. Figures 2 and 3 show the number of offloaded tasks and mobile device energy consumption at distances of 50m, 70m, and 100m from the base station.

[Figure 2: see original paper] demonstrates that more tasks can be offloaded when the device is closer to the base station, as shorter distances result in less wireless communication bandwidth loss. This illustrates that edge computing is more suitable for computation offloading than traditional centralized cloud computing. [Figure 3: see original paper] shows that offloading saves at least 50% of device energy compared to local execution, significantly extending battery life.

2.2 Performance Comparison Between Improved and Classic Auction Algorithms

This experiment uses a scenario with 2 base stations and 60 VMs total. Each time slot generates 20-70 tasks, and each experiment runs for 10 time slots.

[Figure 4: see original paper] compares VM idle time between the improved and classic auction algorithms. We define VM idle time as the difference between when a VM completes a task and when the next time slot begins. The results show that the improved auction algorithm effectively reduces VM idle time. In the classic auction algorithm, when a VM is auctioned, all tasks satisfying the runtime condition compete, potentially binding a long-rental VM with a short-runtime, high-bidding task and creating longer idle periods. In contrast, our improved algorithm sorts both VMs and tasks by time before auctioning, matching short-rental VMs with short-runtime tasks and long-rental VMs with long-runtime tasks, thereby minimizing idle time. Thus, our algorithm not only offers time complexity advantages but also improves VM efficiency.

[Figure 5: see original paper] shows the number of failed tasks (defined as tasks where offloading takes longer than local execution) when task volume far exceeds VM count. The improved auction algorithm produces fewer failed tasks due to lower VM idle time and higher VM utilization.

2.3 Time Consumption Comparison with Other Approaches

This experiment uses a scenario with 1 base station, 30 VMs, and 20-70 tasks per time slot. [Figure 6: see original paper] compares total system time overhead across different algorithms, defined as the sum of time consumed by local tasks, offloaded tasks, and response time (which increases when auctions fail). The results show that computation offloading provides clear time advantages over local execution, reducing time overhead by nearly 50% as user numbers increase. The improved auction algorithm further reduces system computational overhead. Experimental data proves that our improved auction-based offloading strategy effectively reduces service latency and enhances user satisfaction.

---

3 Conclusion

To address the inability of mobile devices to support computationally intensive applications, this paper proposes a computation offloading strategy based on an improved auction algorithm. By considering task requirements, server computing capacity, and network bandwidth, we establish the basis for offloading decisions. By jointly considering task time requirements and MEC server performance, we propose a task scheduling model based on an improved auction algorithm. Experimental results demonstrate that our offloading strategy effectively improves task execution speed, reduces service latency, extends mobile device battery life, and enhances user satisfaction.

References

[1] Kumar K, Liu Jibang, Lu Y H, et al. A survey of computation offloading for mobile systems [J]. Mobile Networks & Applications, 2013, 18(1): 129-140.

[2] Sharifi M, Kafaie S, Kashefi O. A survey and taxonomy of cyber foraging of mobile devices [J]. IEEE Communications Surveys & Tutorials, 2012, 14(4): 1232-1243.

[3] Cuervo E, Balasubramanian A, Cho D, et al. MAUI: making smartphones last longer with code offload [C]//Proc of International Conference on Mobile Systems, Applications, and Services. 2010: 49-62.

[4] Kosta S, Aucinas A, Hui Pan, et al. ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading [C]//Proc of IEEE INFOCOM. 2012: 945-953.

[5] Xia Feng, Ding Fangwei, Li Jie, et al. Phone2Cloud: exploiting computation offloading for energy saving on smartphones in mobile cloud computing [J]. Information Systems Frontiers, 2014, 16(1): 95-111.

[6] Mao Yuyi, Zhang Jun, Letaief K B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices [J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3590-3605.

[7] You Changsheng, Huang Kaibin, Chae H, et al. Energy-efficient resource allocation for mobile-edge computation offloading [J]. IEEE Trans on Wireless Communications, 2017, 16(3): 1397-1411.

[8] Deng Maofei, Tian Hui, Lyu Xinchen. Adaptive sequential offloading game for multi-cell Mobile Edge Computing [C]//Proc of IEEE International Conference on Telecommunications. 2016: 1-5.

[9] Fernando N, Fernando N, Loke S W, et al. Mobile cloud computing: a survey [J]. Future Generation Computer Systems, 2013, 29(1): 84-106.

[10] Li Jirui, Li Xiaoyong, Gao Yunquan, et al. Research on energy saving measures of mobile cloud computing under 5G network [J]. Chinese Journal of Computers, 2017, 40(7): 1491-1516.

[11] Sanaei Z, Abolfazli S, Gani A, et al. Heterogeneity in mobile cloud computing: taxonomy and open challenges [J]. IEEE Communications Surveys & Tutorials, 2013, 16(1): 369-392.

[12] Satyanarayanan M, Bahl P, Caceres R, et al. The case for VM-based cloudlets in mobile computing [J]. IEEE Pervasive Computing, 2009, 8(4): 14-23.

[13] Zhao Ziming, Liu Fang, Cai Zhiping, et al. Edge computing: platform, application and challenges [J]. Journal of Computer Research and Development, 2018, 55(2): 327-337.

[14] Shi Yusong, Sun Hui, Cao Jie, et al. Edge computing: a new computing model in the age of internet of things [J]. Journal of Computer Research and Development, 2017, 54(5): 907-924.

[15] Zhang Yang, Niyato D, Wang Ping. Offloading in mobile cloudlet systems with intermittent connectivity [J]. IEEE Trans on Mobile Computing, 2015, 14(12): 2516-2529.

[16] Jia Mike, Cao Jiannong, Liang Weifa. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks [J]. IEEE Trans on Cloud Computing, 2017, PP(99): 1-1.

[17] Sardellitti S, Scutari G, Barbarossa S. Joint optimization of radio and computational resources for multicell mobile-edge computing [J]. IEEE Trans on Signal and Information Processing over Networks, 2015, 1(2): 89-103.

[18] Tian Hui, Fan Shaoshuai, Lyu Yichen, et al. Mobile edge computing for 5G demand [J]. Journal of Beijing University of Posts and Telecommunications, 2017, 40(2): 1-10.

[19] Chen Xu, Jiao Lei, Li Wenzhong, et al. Efficient multi-user computation offloading for mobile-edge cloud computing [J]. IEEE/ACM Trans on Networking, 2016, 24(5): 2795-2808.

[20] Cheng Kang, Teng Yinglei, Sun Weiqi, et al. Energy-efficient joint offloading and wireless resource allocation strategy in multi-mec server systems [EB/OL]. https://arxiv.org/abs/1803.07243. (2018-03-20).

[21] Lyu Xinchen, Tian Hui, Zhang Pin, et al. Multiuser joint task offloading and resource optimization in proximate clouds [J]. IEEE Trans on Vehicular Technology, 2017, 66(4): 3435-3447.

[22] Yu Bowen, Pu Lingjun, Xie Yuting, et al. Research on mobile edge computing task offloading and base station association collaborative decision making problem [J]. Journal of Computer Research and Development, 2018, 55(3): 537-550.

[23] Lin Xiaopeng, Zhang Heli, Ji Hong, et al. Joint computation and communication resource allocation in mobile-edge cloud computing networks [C]//Proc of IEEE International Conference on Network Infrastructure and Digital Content. 2017: 190-195.

[24] Huang Dong, Wang Ping, Niyato D. A dynamic offloading algorithm for mobile computing [J]. IEEE Trans on Wireless Communications, 2012, 11(6): 1991-1995.

[25] Rappaport T. Wireless communications: principles and practice [M]. Publishing House of Electronics Industry, 2013.

[26] Soyata T, Muraleedharan R, Funai C, et al. Cloud-vision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture [C]//Proc of IEEE Computers and Communications. 2012: 59-66.

Note: Figure translations are in progress. See original paper for figures.