Wireless Networks (2021) 27:609–620
https://doi.org/10.1007/s11276-020-02474-1

Energy-efficient computation offloading strategy with tasks scheduling in edge computing

Yue Zhang[1] · Jingqi Fu[1]

Accepted: 25 September 2020 / Published online: 7 October 2020

Abstract
In mobile edge computing systems, the energy consumption and execution delay can be reduced dramatically by mobile edge computation offloading (MECO) . However, due to the limited computing capacity of edge cloud, an energy-efficient offloading strategy plays a significant role. In this paper, the offloading decision problem for multi-device edge computing systems based on time-division multiple access is studied. The scheduling of offloading devices at the edge cloud is considered when modelling the edge computing system. Then, the offloading decision problem is formulated as an energy consumption minimization problem with the constraint of latency tolerance. It is a mixed integer programming problem of NP-hardness. To address the problem, a Dynamic Programming-based Energy Saving Offloading (DPESO) algorithm is designed to obtain the offloading strategy including the offloading option, offloading sequence and transmission power. First, the MECO with infinite edge cloud capacity is solved by device classification and transmission power decision. Then, we sort and adjust the offloading devices to meet the latency tolerance for the MECO with finite edge cloud capacity. Finally, simulation results demonstrate that the DPESO algorithm achieves better energy efficiency than the baseline strategies and has good scalability.

Keywords Mobile edge computing · Computation offloading · Resource competition · Dynamic programming · Energy-efficient

1 Introduction

With the dramatical development of the Internet of Things (IoT) technology, more and more devices running computation-intensive applications access the Internet. However, devices have limited battery lifetime and computation resources in general, making them unqualified for processing resource-intensive services [1].

Cloud computing has been envisioned as an efficient way to address the above challenge. By offloading tasks to cloud infrastructures, cloud computing can enhance the computation capabilities of devices [2]. Nevertheless, the weaknesses of cloud computing are revealed with the emergence of the Internet of Everything (IoE). More than 50 billion terminal devices will access the Internet by 2020 [3]. As a result, cloud resources are not infinite any more, and the bandwidth limitation may lead to unexpected latency as well.

Mobile edge computing has been proposed to overcome the weaknesses [4]. Figure 1 shows the architecture of edge computing systems. Mobile edge computing can provide computing power through the resources deployed in the "edge", such as smart gateways, base stations and so on. Devices usually access the edge cloud via a single-hop network, so the execution latency and transmission energy consumption can be reduced. By mobile edge computation offloading (MECO), tasks can be executed by local devices and the edge in parallel [5]. However, overmuch offloading devices may cause a fierce competition for resources, making it the mainstream to obtain an efficient offloading strategy.

This paper focuses on designing an energy-efficient computation offloading algorithm for edge computing

✉ Jingqi Fu
jqfu@shu.edu.cn

Yue Zhang
sodalife@shu.edu.cn

[1] School of Mechanical Engineering and Automation, Shanghai University, Shanghai, China

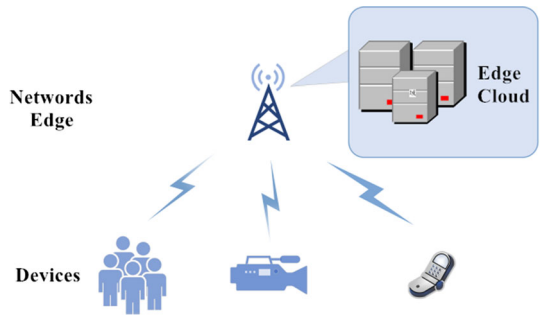

Fig. 1 Mobile edge computing systems

systems. Assume that computation tasks of multiple devices belong to the same service which should be completed within a certain duration, i.e. the latency tolerance [6–8]. Tasks can be executed either by devices (local computing) or by the edge server (edge computing) [9], and the option is binary. The algorithm aims at minimizing the total energy consumption while satisfying the latency tolerance.

The novel contributions of this paper are summarized as follows:

- Aiming at the inadequacies of edge execution model, which may lead to resource conflict and resource waste, we present the scheduling model for edge server. Then, the optimization problem of minimizing the total energy consumption is formulated, the solution of which is more practically feasible and the resource allocation is more efficient.
- Due to the NP-hardness of the optimization problem, we transform the problem into an optimization problem with saved energy maximization as the objective function and transmission time as the decision variable. The transformed problem is an analogous packet knapsack problem, which can be solved by dynamic programming.
- We propose a Dynamic Programming-based Energy Saving Offloading algorithm to solve the saved energy maximization problem. The proposed algorithm makes decision on offloading option, transmission power and offloading sequence to achieve rationality of resource allocation and better energy efficiency.

The remainder of this paper is organized as follows. Sect. 2 reviews related works. Sect. 3 introduces the system model, followed by the problem formulation in Sect. 4. Algorithm design is presented in Sect. 5, and simulation results and analyses are given in Sect. 6. Finally, Sect. 7 concludes the paper.

2 Related works

Numbers of fruitful researches have been carried out on MECO [4], such as system architectures [5, 10], enabling technologies [2, 11] and offloading optimization [12]. The design of offloading control strategies is a key challenge to exploit advantages of edge computing.

Currently, MECO can be divided into single-user offloading [13–16] and multi-user offloading [17–19] according to application scenarios. In single-user offloading, Mao et al. [13] adopted execution latency and task success rate as performance metrics to evaluate offloading strategy, then proposed a low-complexity online algorithm for dynamic offloading decision. You et al. [14] first optimized local computing and computation offloading based on the known channel state, then chose the mode with more energy saving from the above two modes, and finally extended it to the dynamic channel to realize asymptotic optimal allocation of computing tasks. Aiming at the problems of link selection and transmission scheduling in dynamic environment, Xiang et al. [15] designed an approximate dynamic Programming algorithm which can solve the "curse of dimensionality" effectively to jointly optimize system throughput and energy consumption. Zhang et al. [16] studied the energy-efficient computation offloading under stochastic wireless channel. Based on the relationship between the data size and latency tolerance, the authors identified the optimal operational region for local execution and edge execution.

In multi-user offlaoding, to guarantee the fairness among devices, Du et al. [17] formulated the multi-user computation offloading problem as an optimization problem to minimize the maximal weighted cost of devices under the latency tolerance constraint. Then, the offloading decision is obtained by semi-definite relaxation and random extraction. Guo et al. [18] explored the contradictory relationship between energy consumption and task completion time, and proposed a distributed algorithm to make decisions on offloading option, CPU frequency and transmission power respectively for each device. On the basis of the single-user partitioning, Yang et al. [19] used a reward function to associate task partitioning with task offloading, and then proposed an offline heuristic algorithm to minimize average completion time for all the users.

Besides, MECO can be divided into binary offloading [6, 20–22] and partial offloading [7, 23–25] according to offloading mode. In binary offloading model, the computation task can be either executed locally or at the edge server as an integral whole. Liu et al. [6] formulated an energy consumption minimization problem as a mixed integer programming problem considering the task dependency. An energy-efficient collaborative task computation

offloading (ECTCO) algorithm was designed to solve the problem. In [20], Chen et al. formulated the computation offloading decision making problem a multi-user game. The efficient offloading strategy was obtained in a distributed manner based on game theory. Zhang et al. [21] introduced the residual energy of devices' battery as a weighting factor into the offloading problem. The proposed iterative search algorithm combining interior penalty function with D.C. programming achieved low energy consumption and long lifetime of devices. Dinh et al. [22] researched an edge computing system consists of a mobile device and multiple access points. They proofed that when the mobile device can offload computation tasks to different access points, the CPU frequency can affect the task allocation decision.

In partial offloading model, the computation task can be divided into several sub-tasks, some of which are executed locally while the rest are executed at the edge server. In [7], You et al. utilized the thought of partial offloading for obtaining a binary offloading solution of time-division multiple access (TDMA) and OFDMA edge computing system. However, few researches consider the scheduling of offloading tasks at the edge server. Hao et al. [23] introduced the concept of task caching and established an optimization problem under the constraints of computational and storage resources. The alternating iterative algorithm-based strategy could achieve lower energy consumption. In [24], Ren et al. derived the optimal task segmentation strategy. Aiming at the piecewise convex optimization problem, the authors proposed a sub-gradient algorithm to obtain the optimal resource allocation. Wang et al. [25] considered devices with flexible CPU frequency when formulated the joint optimization problem of energy consumption and latency. The results are extended to the scenario of multiple edge servers.

In the existing researches on the computation offloading decision problem, only the execution time is taken into account in the phase of edge execution, but the impact of task scheduling is less considered. It may result in an infeasible optimal solution in practice or a waste of system resources. Different from the previous studies, taking the simultaneous availability of different resources and interaction among offloading tasks into consideration, we present the scheduling model for edge server and formulate the optimization problem of minimizing the total energy consumption. The solution of the optimization problem is more practically feasible and the resource allocation is more efficient.

3 System model

In this section, the system model including local computing model, communication model and edge computing model is introduced. Then, the energy consumption minimization problem is formulated as a mixed integer programming problem.

3.1 Scenario description

In this paper, an edge computing system based on TDMA[1] is considered. The system consists of an edge server and n devices with a task, denoted by a set $N = \{1, 2, \ldots, n\}$. Devices can access the edge server by wireless channel (e.g., 5G and WiFi). In TDMA systems, time is divided into several time slots and each user can only transmit data in their own specified time slot. It allows multiple users to share bandwidth and transmission media at different time slots. The mechanism of TDMA mode determines that the data transmission process of devices is sequential.

For convenience of description, device i refers to the device that owns computation task i. The service cannot be provided until all tasks are completed. To guarantee the quality of service (QoS), tasks should be completed within the latency tolerance, denoted as T.

Similar to existing studies [6, 7, 21, 26, 27], the problem is studied in a quasi-static scenario. The continuous time is divided into periods of time. The system state remains unchanged in each period (called a decision cycle). In a specific decision cycle, the edge server is envisioned to have completed the task parameter collection before making offloading decision. The devices with a task in the system will upload the task parameters, while the devices without task will keep silent and enter the sleep mechanism, making their energy consumption is ignored. Devices with computation tasks may vary between different decision cycles, however, they can all be referred to as "n devices with a computation task".

To obtain an energy efficient offloading strategy, the edge server needs to collect task parameters. This process is completed by device profiler [28] and edge server's system manager module [29] (such as Rover Toolkit [30] and Profiler [31]). Device profilers monitor the running state of mobile devices and the system management module is responsible for node awareness. When devices have tasks, the manager module will notify devices to report the task parameters to the edge server. Therefore, we assume that the edge server has a good knowledge of task parameters such as data size and computation complexity.

[1] Although we assume a TDMA scenario, our analysis can be extended into other access schemes with a minor modification on the framework.

The edge server will determine the offloading strategy based on these information, including the offloading option, offloading sequence and transmission power.

3.2 System model description

3.2.1 Local computing model

For local computing, computation capability, data size and computation complexity are assumed to vary from devices to simulate a heterogeneous system. Let d_i and c_i denote the data size (in bits) and the computation complexity (in CPU cycles per bit) of device i, respectively. In addition, computation capability of device i is represented by CPU frequency f_i. Hence, the local execution time of task i can be expressed as

$$t_i^l = \frac{d_i \cdot c_i}{f_i}. \tag{1}$$

According to [32], the energy consumption per CPU cycle of processor is $e = \kappa \cdot f^2$, where κ is a constant related to chip architecture. Thus, the local execution energy consumption of task i is

$$E_i^l = \kappa \cdot f_i^2 \cdot d_i \cdot c_i. \tag{2}$$

3.2.2 Communication model

When a device is chosen to offload its task, the communication between the device and the edge server comprises two phases, i.e. computation offloading and results downloading. For computation offloading phase, given the transmission power p_i and the channel gain g_i, the data transmission rate of device i can be calculated according to Shannon theory [33] as

$$r_i = B \cdot \log_2\left(1 + \frac{p_i \cdot g_i}{\omega_0}\right). \tag{3}$$

Here B means the channel bandwidth and ω_0 the variance of complex white Gaussian channel noise. Based on r_i, the transmission time and energy consumption of device i can be computed as

$$t_i^o = \frac{d_i}{r_i} = \frac{d_i}{B \cdot \log_2\left(1 + \frac{p_i \cdot g_i}{\omega_0}\right)}, \tag{4}$$

and

$$E_i^o = p_i \cdot t_i^o = \frac{p_i \cdot d_i}{r_i}. \tag{5}$$

The delay and energy consumption of results downloading are ignored [7, 21]. First, because the data size of results is much smaller than that of input, and second, because the energy consumption is mainly afforded by the edge server, which is powered by wired supply. Therefore, computation offloading is considered equivalent to the communication process in this paper.

3.2.3 Edge computing model

Similar to the local computing model, let f_e be the CPU frequency of edge cloud, then the time of edge computing for task i is given by

$$t_i^e = \frac{d_i \cdot c_i}{f_e}. \tag{6}$$

As all the offloading tasks are executed by an edge server and the limited edge cloud capacity, the edge server needs to allocate the computation resources among all offloading devices efficiently. The execution at the server is treated as sequential execution in this paper. In fact, serialization or parallelization of task execution has no effect on the overall completion time, which is the metric of interest in the paper. In a real edge computing system, both serialization and parallelization of task execution at the server are feasible. Evidently, the allocation satisfies the following constraints: (1) The completion time of each task should be shorter than T; (2) The execution cannot start until the transmission ends for an offloading task; (3) The execution starts only when the edge server is idle.

To describe the order of task execution at the edge cloud, the definitions of transmission finishing time of device i and idle time of edge cloud are given.

Definition 1 The transmission finishing time of device i is the time when the input data of task has been fully transmitted from device i to the edge cloud, denoted by t_i^f.

Definition 2 The idle time of edge cloud is the time when the edge cloud completes executing the current offloading tasks, denoted by t_e^i.

Then, the first constraint can be expressed as

$$t_i^f + t_i^e \le T, i \in O. \tag{7}$$

The second and third constraints can be expressed as

$$t_i^b \ge \max\left\{t_i^f, t_e^i\right\}, i \in O. \tag{8}$$

Here, O is the set of offloading devices and t_i^b is the execution beginning time of device i at the edge cloud.

According to the optimization theory, the order of task execution at the edge cloud adopts queue mode, i.e., "first offload, first executed". A directed acyclic graph $G = (V, E)$ shown in Fig. 2 is adopted to model the order of task execution at the edge cloud, where $V = \{DT_i, EC_i\}$ denotes the node set for data transmission (DT_i) and edge

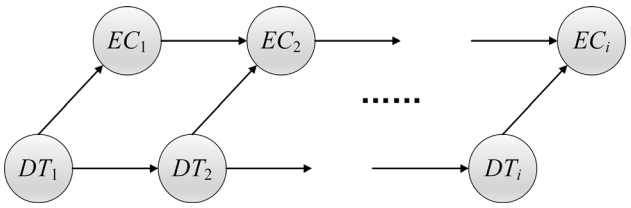

Fig. 2 Order of task execution at the edge cloud

computing (EC_i) process, and the dependency relationship among the nodes is represented by the directed arc set in E. For example, task 2 cannot begin edge computing until the input data has been transmitted to edge cloud and task 1 has been executed.

Let $P(i)$ denote the precursor node set of task i, Then, the completion time of task i at the edge cloud can be derived as

$$t_i^c = \begin{cases} t_i^o + t_i^e, & P(i) = \emptyset \\ \max\left\{t_j^c, \sum\limits_{j \in P(i)} t_j^o + t_i^o\right\} + t_i^e, & P(i) \neq \emptyset, \quad i \in O. \end{cases} \tag{9}$$

4 Problem formulation

In this section, the offloading decision problem is formulated, with the objective for the minimum total energy consumption of devices. A binary variable, α_i is defined to represent the offloading option of device i. $\alpha_i = 1$ if device i is selected to offload, otherwise $\alpha_i = 0$. Thus, the energy consumption of device i can be integrated as

$$E_i = (1 - \alpha_i) \cdot E_i^l + \alpha_i \cdot E_i^o. \tag{10}$$

The offloading decision problem can be formulated as

$$\begin{aligned} \text{P1}: \min_{\alpha_i} & \sum_{i=1}^{n} (1 - \alpha_i) \cdot \kappa \cdot f_i^2 \cdot d_i \cdot c_i + \alpha_i \cdot \frac{p_i \cdot d_i}{r_i} \\ s.t. \quad & \text{C1}: t_i^l \leq T, i \in N \text{ and } i \notin O, \\ & \text{C2}: t_i^c \leq T, i \in O, \\ & \text{C3}: t_i^b \geq \max\left\{t_i^f, t_e^i\right\}, i \in O. \\ & \text{C4}: \alpha_i \in \{0, 1\}, i \in N. \end{aligned} \tag{11}$$

In P1, C1 and C2 are the latency tolerance constraint for local computing devices and offloading devices respectively; C3 represents the resources competition constraint; C4 is the value constraint of each device's offloading option. Problem P1 is NP-hard to solve [34]. To address the issue, we adopt a two-stage solution approach. The problem P2 is obtained by relaxing the constraint of the edge cloud capacity shown as follows:

$$\begin{aligned} \text{P2}: \min_{\alpha_i} & \sum_{i=1}^{n} (1 - \alpha_i) \cdot \kappa \cdot f_i^2 \cdot d_i \cdot c_i + \alpha_i \cdot \frac{p_i \cdot d_i}{r_i} \\ s.t. \quad & \text{C1}: t_i^l \leq T, i \in N \text{ and } i \notin O, \\ & \text{C2}: t_i^c \leq T, i \in O, \\ & \text{C4}: \alpha_i \in \{0, 1\}, i \in N. \end{aligned} \tag{12}$$

Due to the edge cloud capacity is assumed be infinite, the execution time of task i at the edge cloud is 0, i.e., C3 are ignored.

5 Algorithm design

In this section, the computation offloading decision problem with infinite edge cloud capacity and finite edge cloud capacity are solved in turn. The original optimization problem is transformed into an equivalent form with saved energy maximization as the objective function and transmission time as the decision variable. Furthermore, the function between the objective function and decision variables and the monotonicity of the function are given. Based on dynamic programming theory, the DPESO algorithm is proposed to solve the computation offloading decision problem.

5.1 MECO with infinite edge cloud capacity

5.1.1 Device classification

To guarantee the tasks are completed in T, devices that are not qualified should be selected to offload at first. The definition of the offloading necessary device is given in terms of the relationship of the local execution time and latency tolerancy as follows.

Definition 3 For device i, if $t_i^l > T$, then device i belongs to offloading necessary device, denoted as O_n. Otherwise, device i can be selected to local computing as well as computation offloading.

Accordig to (1), the offloading necessary devices are the devices which have the poor computation capacity, large data size and high computation complexity. As we study the binary offloading model, for the offloading necessary devices, their offloading option $\alpha_i = 1$ and we need to determine their transmission power.

5.1.2 Transmission power decision

According to (2), the energy consumption of local computing is a constant for each device. Moreover, according to (3)–(5), if device i is selected to offload, the energy consumption can be calculated as

$$E_i^o = \frac{\omega_0}{g_i} \cdot \left(2^{\frac{d_i}{B \cdot t_i^o}} - 1\right) \cdot t_i^o. \tag{13}$$

It can be observed that E_i^o is a injective function of t_i^o. Thus, once the transmission time t_i^o is determined, the transmission power can be calculated according to (4).

To achieve energy-efficient computation offloading, edge computing is expected to save energy compared to local computing. For device i, the energy saved by edge computing can be expressed as the difference between E_i^l and E_i^o, i.e.,

$$\begin{aligned} E_i^s &= E_i^l - E_i^o \\ &= \kappa \cdot f_i^2 \cdot d_i \cdot c_i - \frac{\omega_0}{g_i} \cdot \left(2^{\frac{d_i}{B \cdot t_i^o}} - 1\right) \cdot t_i^o. \end{aligned} \tag{14}$$

Theorem 1 *For mobile edge computing systems based on TDMA, the energy saved by edge computing increases as the extension of transmission time.*

Proof Theorem 1 can be proved by derivation. The derivative of E_i^s with respect to t_i^o is

$$E_i^{s\prime}\left(t_i^o\right) = \frac{\omega_0}{g_i} \cdot \left(\frac{d_i}{B \cdot t_i^o} \cdot 2^{\frac{d_i}{B \cdot t_i^o}} \cdot \ln 2 - 2^{\frac{d_i}{B \cdot t_i^o}} + 1\right). \tag{15}$$

Define a variable $x = d_i / \left(B \cdot t_i^o\right)$, $x \geq 0$, and a function $f(x) = x \cdot 2^x \cdot \ln 2 - 2^x + 1$, we just need to determine the positivity of $f(x)$ because $\omega_0 / g_i \geq 0$. The derivative of $f(x)$ is

$$f'(x) = x \cdot 2^x \cdot (\ln 2)^2. \tag{16}$$

It is evident that for $x \geq 0$, we have $f'(x) \geq 0$. Besides, we have $f(0) = 0$. Thus, it can be derived that $f(x) \geq 0$ for $x \geq 0$, i.e., the derivative of E_i^s is non-negative. In other words, the energy saved by edge computing increases as the extension of transmission time. □

For the sake of discussion, we first give the definition of minimum transmission time.

Definition 4 For device i, the transmission time that satisfies $E_i^s = 0$, namely $E_i^l = E_i^o$ is defined as the minimum transmission time, denoted by t_i^m.

According to Theorem 1 and Definition 4, the following conclusion can be drawn: only when the transmission time of device i, $t_i^o > t_i^m$, device i tend to offload its task. Otherwise, it is more efficient to choose local computing. Futhermore, the objective of P2 can be replaced by $\max \sum_{i=1}^n E_i^s$. We just need to determine the transmission time for each device. For device i, if $t_i^o = 0$ then $\alpha_i = 0$, otherwise, $\alpha_i = 1$.

When the edge cloud has infinite capacity, the time of edge computing $t_i^e = 0$ according to (4). The offloading sequence schematic diagram is shown as Fig. 3. Then, (9) is reduced to the following form.

$$t_i^c = \sum_{j \in P(i)} t_j^o + t_i^o, \quad i \in O. \tag{17}$$

Then, the constraint C2 can be expressed as $\sum_{i \in O} t_i^o \leq T$. Through the transformation of the objective and constraint, we found that P2 is very similar to the packet knapsack problem, the difference is that t_i^o is a continuous variable rather than a discrete variable.

To determine the transmission time of each user, a time infinitesimal $\Delta t (\Delta t << T)$ is adopted to discretize t_i^o. If device i is offloading necessary device, to achieve energy efficiency, the least transmission time should be t_i^m. For device i, the possible values of t_i^o are

$$t_i^o = \begin{cases} t_i^m, t_i^m + \Delta t, \ldots\ldots, t_i^m + k_1 \cdot \Delta t, & t_i^l > T \\ 0, \Delta t, \ldots\ldots, k_2 \cdot \Delta t, & otherwise \end{cases} \tag{18}$$

So, we first allocate the respective t_i^m to the offloading necessary devices. The remaining time to allocate is $T' = T - \sum_{i \in O_n} t_i^m$. For offloading necessary devices, we just need to determine additional redistribution of transmission time over t_i^m.

The computation offloading decision problem is transformed into a packet knapsack problem which can be solved by dynamic programming. Each device is treated as a group and for device i, the possible values of transmission time are the items in this group and only one certain value of t_i^o can be selected. The key to solve dynamic programming problem is to find the state transfer equation. For any possible value of t_i^o, there are two options, i.e., selecting the value or not selecting the value.

Let $E^*[i][t]$ represent the maximum energy consumption that can be saved when the former i devices using time t to offload. For device i, if t_i^o is selected to be a certain value in (18), the saved energy by device can be calculated

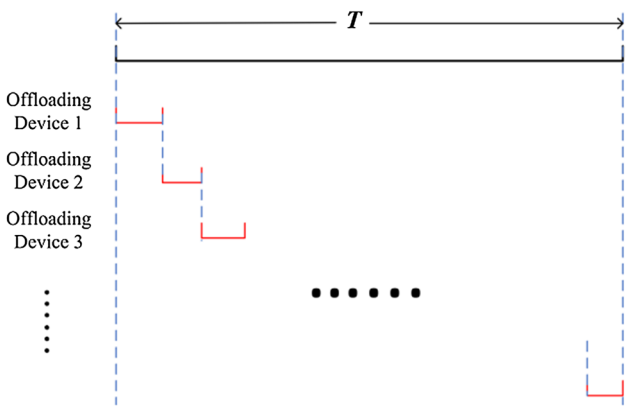

Fig. 3 Offloading sequence with infinite edge capacity

according to (14). Then, the remaining allocatable time is $T' - t_i^o$. In this case, we can obtain that

$$E^*[i][t] = E^*[i-1]\left[t - t_i^o\right] + E_i^s\left(t_i^o\right). \tag{19}$$

Otherwise, if t_i^o is selected to be 0, The remaining allocatable time is still t. In this case, we can obtain that

$$E^*[i][t] = E^*[i-1][t]. \tag{20}$$

Thus, the state transfer equation is as follows

$$E^*[i][t] = \max\left\{E^*[i-1][t], E^*[i-1]\left[t - t_i^o\right] + E_i^s\left(t_i^o\right)\right\}. \tag{21}$$

Algorithm 1

Input: $B, T, C_e, \omega_0, \kappa, d_i, c_i, f_i, g_i, \Delta t$;
Output: t_i^o, p_i, α_i;
1: **for** $i = 1$ to n **do**
2: Calculate t_i^l of each user according to (1);
3: **if** $t_i^l > T$ **then**
4: Device i belongs to offloading necessary devices;
5: **end if**
6: **end for**
7: Calculate the remaining allocatable time $T' = T - \sum_{i \in O_n} t_i^m$;
8: **for** $i = 1$ to n **do**
9: **for** $t = T'$ to 0 **do**
10: **if** $i \in O_n$ **then**
11: **for** $t_i^o = t_i^m$ to $\lfloor t/\Delta t \rfloor \cdot \Delta t$ **do**
12: $E^*[i][t] = \max\{E^*[i-1][t], E^*[i-1][t - t_i^o] + E_i^s(t_i^o)\}$;
13: **end for**
14: **else**
15: **for** $t_i^o = 0$ to $\lfloor t/\Delta t \rfloor \cdot \Delta t$ **do**
16: $E^*[i][t] = \max\{E^*[i-1][t], E^*[i-1][t - t_i^o] + E_i^s(t_i^o)\}$;
17: **end for**
18: **end if**
19: **end for**
20: **end for**
21: Calculate t_i^o of each user by backtracking;
22: Calculate p_i and α_i according to (4);
23: **return** t_i^o, p_i, α_i.

Then, the transmission time of each device can be obtained by backtracking. But it is worth notice that for offloading necessary devices, the actual transmission time has to be plus its t_i^m. The transmission power can be calculated according to (4). The detailed process is described in Algorithm 1.

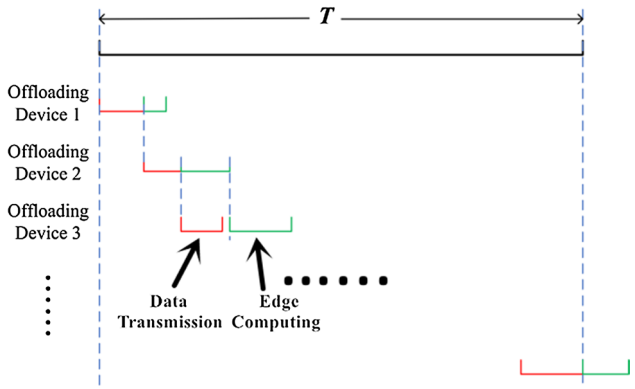

Fig. 4 Offloading sequence with finite edge capacity

5.2 MECO with finite edge cloud capacity

When the edge cloud has finite capacity, the time of edge computing cannot be ignored any more. The offloading sequence schematic diagram describing the finite edge capacity sees Fig. 4. It can be known from (9) that the completion time is related to the precursor node set $P(i)$, in other words, the completion time is related to the sequence of data transmission. So we can shorten the overall completion time by adjusting the offloading sequence [35]. If the edge computing time of the previous offloading device is longer than own transmission time, the overall completion time will be prolonged and the overall completion time is related to the offloading order. In order to make the overall completion time as short as possible, we need to manage the order among offloading devices.

For mobile edge computing systems based on TDMA, there is a channel for data transmission and an edge cloud for edge computing. In fact, data transmission and edge computing can be seen as two working procedure, and the offloading sorting is a two-stage production schedule. To achieve the shortest overall completion time, Jackson Algorithm is applied to solve this problem.

Algorithm 2 Dynamic Programming-based Energy Saving Offloading Algorithm.

Input: $B, T, C_e, \omega_0, \kappa, d_i, c_i, f_i, g_i, \Delta t$;
Output: t_i^o, p_i, α_i;
1: **Initialize:** Prepare a list l_1 whose length is the size of set O, and calculate the initial t_i^o, p_i and α_i of each device according to Algorithm 1;
2: **for** $i \in O$ **do**
3: Calculate the time of edge computing t_i^e according to (6);
4: **end for**
5: Sort the t_i^o and t_i^e of all offloading devices in ascending order, and the sorted list is denoted as l_2;
6: **for** t in l **do**
7: **if** t is t_i^o **then**
8: Device i is placed in the first free position, and remove the corresponding t_i^e from l_2;
9: **else**
10: Device i is placed in the last free position, and remove the corresponding t_i^o from l_2;
11: **end if**
12: **end for**
13: Calculate the completion time t_i^c of each offloading device according to (9);
14: **for** $i \in O$ **do**
15: **if** $t_i^c > T$ **then**
16: $t_i^o = p_i = \alpha_i = 0$ and update l_1;
17: **end if**
18: **end for**
19: **return** $t_i^o, p_i, \alpha_i, l_1$.

The sorted offloading strategy violates the constraint of latency tolerance, so we need to adjust the above solution. The completion time of each offloading device can be obtained according to (9). The devices whose t_i^c exceeds T will be shifted to local computing. The detailed process to obtain offloading decision with finite edge cloud capacity is described in Algorithm 2.

5.3 Complexity of DPESO

Firstly, the complexity of Algorithm 1 to solve the computation offloading decision with infinite edge cloud capacity is analyzed. The time complexity of the loop to determine whether device i is offloading necessary device (line 1-6, Algorithm 1) is $O(n)$. As to the step to determine the offloading options and transmission power (line 8-20, Algorithm 1), the number of cycles in the first loop is n. The second and third loops are the traversal from 0 to T with a step size of Δt, each of which has $T/\Delta t$ operations. Therefore, the complexity of this step is $O(n(T/\Delta t)^2)$. After the above step, we can obtain the value of $E^*[i][t](i = 1, \ldots, n; t = 0, \Delta t, \ldots, T)$ for each i and t, making up the state space $\mathbb{E}_{n\times(T/\Delta t)}$. We need to traverse the state space $\mathbb{E}_{n\times(T/\Delta t)}$ to determine the transmission time for each device, and it has the complexity of $O(n \cdot T/\Delta t)$. In summary, the complexity of Algorithm 1 is

$$O\left(n + n\left(\frac{T}{\Delta t}\right)^2 + n \cdot \frac{T}{\Delta t}\right) = O\left(n\left(\frac{T}{\Delta t}\right)^2\right). \tag{22}$$

Then, the complexity of the proposed DPESO algorithm to solve the computation offloading decision with finite edge cloud capacity is analyzed. Because Algorithm 2 is based on the results of Algorithm 1, the complexity of Initialize in Algorithm 2 is $O(n(T/\Delta t)^2)$. The time complexity of the loop to calculate the time of edge computing (line 2–4, Algorithm 2) is $O(n)$. The time complexity of the offloading sorting (line 5–12, Algorithm 2) is $O(n\log_2 n)$. Adjusting the feasibility of the solution is the process of comparing the completion time of each device with the latency tolerance, whose time complexity is $O(n)$. In summary, the complexity of the proposed DPESO algorithm is

$$O\left(n\left(\frac{T}{\Delta t}\right)^2 + 2n + n\log_2 n\right) = O\left(n \cdot \max\left\{\left(\frac{T}{\Delta t}\right)^2, \log_2 n\right\}\right). \tag{23}$$

Table 1 Simulation parameters setting

Parameters	Value
B	2 MHz
ω_0	10^{-9}
T	100 ms
κ	10^{-27}
f_e	10 GHz
d_i	40–200 kb uniformly
c_i	100–500 cycles per bit uniformly
f_i	0.5–1 GHz uniformly
d_i	5–30 m uniformly

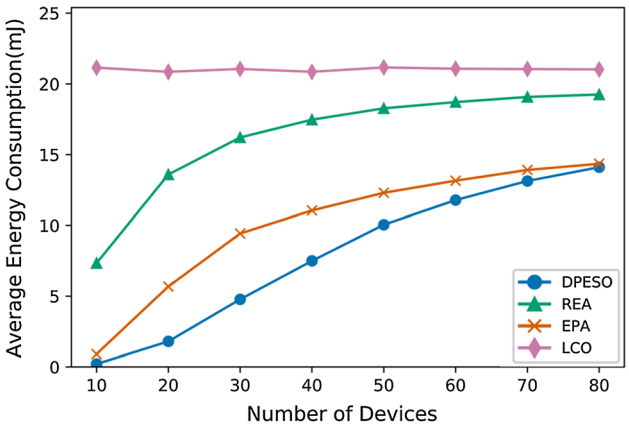

Fig. 5 Average energy consumption versus the number of devices

6 Numerical simulation

In this section, the performance of the proposed DPESO algorithm is evaluated by simulation. The simulation settings are first given and the following simulation results show the efficiency in energy saving.

6.1 Simulation settings

An edge computing system which consists of multiple devices and an edge server is simulated to run the proposed DPESO algorithm. The simulation is implemented randomly by Monte Carlo method. The main simulation parameters are listed in Table 1 unless otherwise specified. According to the model for cellular radio environment [36], the channel gain is set as $g_i = d_i^{-\alpha}$, where d_i is the distance between the edge server and device i and $\alpha = 4$ is the path loss factor.

To visually evaluate the performance of the proposed DPESO algorithm, the following baseline strategies are simulated for comparison.

- Local computing only (LCO): Except for offloading necessary devices, all tasks are executed locally.
- Resource equal allocation (REA): The resources are allocated to offloading devices equally.
- Energy-based priority allocation (EPA): Devices with high local computing energy consumption has a high offloading priority, and the transmission time is allocated based on local computing energy consumption.

6.2 Performance and analysis

The first part of simulation is the performance of the proposed DPESO versus the number of devices. Fig. 5 shows the energy consumption of the LCO, REA, EPA and DPESO strategies under different numbers of devices. Following observations can be made. First, the energy consumption of each offloading strategy increases as the number of devices grows. Second, the energy consumption of the proposed DPESO strategy is much lower than the LCO, REA and EPA strategies. For example, when the number of devices is 50, the DPESO strategy can reduce 52.49%, 45.02% and 18.28% of energy consumption compared with the strategies of LCO, REA and EPA, respectively.

To evaluate the DPESO algorithm more comprehensively, two metrics, i.e., the utilization of edge server (UES) and number of offloading devices (NOD) are introduced. In Fig. 6(a), it can be observed that the UES of the DPESO and EPA strategies increases gradually until a stable value is reached while that of the REA strategy stays low. Fig. 6(b) shows that for the DPESO and EPA strategies, the NOD first increases to about 20 and then decreases with the growing number of devices. However, the DPESO strategy has a slower rate of decline than the EPA. This is because when the UES reaches a high level, further growth in the number of devices will increase competition for edge resources, causing the decline in the NOD. And the DPESO algorithm well considers the competition among offloading devices, so it makes more devices benefit from edge computing. Furthermore, the combination of Figs. 5 and 6

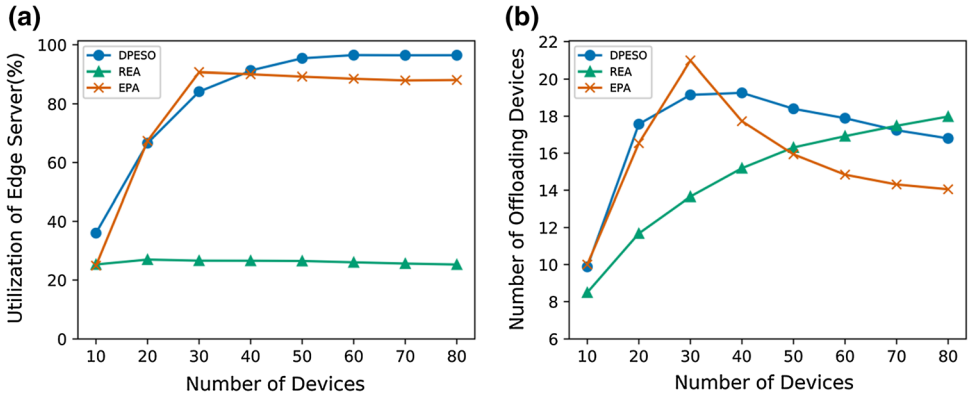

Fig. 6 **a** The UES versus the number of devices; **b** the NOD versus the number of devices

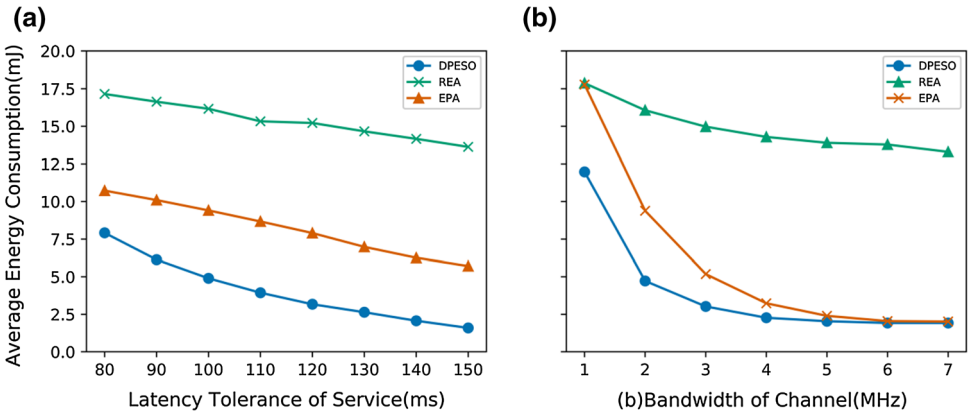

Fig. 7 **a** Impact of the latency tolerance; **b** impact of the bandwidth

Table 2 Average running time versus Δt

Δt (ms)	0.02	0.05	0.1	0.2	0.5	1	2
Average running time (s)	9.4125	1.5314	0.4722	0.1197	0.0401	0.0183	0.0124
Average energy consumption (mJ)	4.770	4.772	4.779	4.795	5.013	6.125	8.434

Table 3 Average running time versus number of devices

Number of devices	10	20	30	40	50	60	70	80
Average running time (s)	0.1847	0.3495	0.4722	0.5319	0.6835	0.8291	0.9745	1.1218

illustrates the DPESO strategy can make full use of the limited resources.

The second part of simulation evaluates the impact of system parameters. Edge computing has the advantage over cloud computing in the better channel quality. So B is chosen for simulation to evaluate the DPESO algorithm. In addition, the impact of the latency tolerance of service T is also evaluated. The number of devices is set to 30 in this part.

Figure 7 shows the impact of T and B on the average energy consumption. Experiments are conducted with different value of $T = 80, 90, \ldots, 150$ ms and $B = 1, 2, \ldots, 7$ MHz, respectively. In Fig. 7(a), It can be seen that the energy consumption under the REA and EPA strategies decline almost linearly with the increasing T. With respect to the DPESO strategy, the rate of decline in energy consumption slows down gradually. Furthermore, the DPESO strategy always consumes the least energy and has the greatest reduction in energy consumption. This is because the relaxation of the latency tolerance constraint reduces the number of offloading necessary devices, weakening the competition among offloading devices and releasing the resources occupied by these devices.

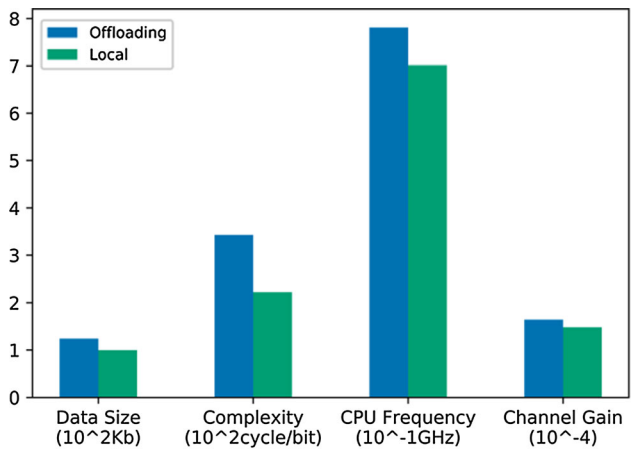

Fig. 8 Features of local devices and offloading devices

In Fig. 7(b), as B increases, the energy consumption performed by every strategy tends to decrease but the rate of decline slows down gradually. It indicates that the enriching of a single type of resource is not a sustainable way to reduce energy consumption. So when deploying the resources of edge computing systems, people should coordinate deployment throughout the system. Moreover, the energy consumption by the DPESO strategy is always the lowest regardless of channel bandwidth.

In the third part of simulation, the average running time of the proposed algorithm is evaluated on a computer equipped with Intel Core i5-7500, 3.40 GHz processor and 8 GB RAM. Table 2 shows the results of average running time versus the different value of Δt. The number of devices is set to 30. It can be observed that the average running time of the algorithm almost decreases exponentially with the increase of Δt, while the average energy consumption increases. When Δt is selected to be an

appropriate value (0.05–0.2ms), the proposed algorithm can obtain the almost optimal offloading decision in a relatively short time.

Table 3 shows the results of average running time versus the different number of devices. The value of Δt is set to 0.1ms. It can be observed that the average running time of the algorithm increases almost linearly with the number of devices, which further illustrates the scalability of the algorithm.

In the last part of simulation, the main parameters of devices are analyzed. Figure 8 shows the means of d_i, c_i, f_i and g_i of offloading devices and local computing devices, respectively.By comparison, it is observed that offloading devices have larger data size, better channel quality, higher computational complexity and CPU frequency. This can be explained by (2) and (13), which illustrate that E_i^l is positively correlated with d_i; c_i; f_i and E_i^o is negatively correlated with g_i. The higher energy consumption of local computing, the more likely it is to save energy by offloading. The better channel quality is, the lower transmitting power of device will be, that is, the more inclined to offload.

7 Conclusions

In this paper, the energy saving problem of computation offloading decision in multi-device edge computing systems is studied. Aiming at the inadequacies of edge execution model, which may lead to resource conflict and resource waste, we present the scheduling model for edge server. Then, the computation offloading decision problem is formulated as an energy consumption optimization problem with the latency tolerance constraint. In order to solve the above NP-hard problem, the original optimization problem is transformed into an analogous packet knapsack problem. Then, the DPESO algorithm is designed to obtain the offloading strategy including the offloading option, offloading sequence and transmission power. Simulation results show that the DPESO offloading strategy can achieve the much better performance than the baseline strategies in terms of energy consumption. Besides, simulations of system parameters demonstrate the good scalability of the DPESO algorithm.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Mung, C., & Zhang, T. (2016). Fog and IoT: An overview of research opportunities. *IEEE Internet of Things Journal*, *3*(6), 854–864.
2. Osanaiye, O., Chen, S., Yan, Z., Lu, R., Choo, K.-K. R., & Dlodlo, M. (2017). From cloud to fog computing: A review and a conceptual live VM migration framework. *IEEE Access*, *5*, 8284–8300. https://doi.org/10.1109/ACCESS.2017.2692960.
3. Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646.
4. Sun, X., & Ansari, N. (2016). EdgeIoT: Mobile edge computing for the Internet of Things. *IEEE Communications Magazine*, *54*(12), 22–29.
5. Mach, P., & Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, *19*(3), 1628–1656.
6. Liu, F., Huang, Z., & Wang, L. (2019). Energy-efficient collaborative task computation offloading in cloud-assisted edge computing for IoT sensors. *Sensors (Basel)*, *19*(5), 1105.
7. You, C., Huang, K., Chae, H., & Kim, B. H. (2017). Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, *16*(3), 1397–1411.
8. Liu, M., & Liu, Y. (2018). Price-based distributed offloading for mobile-edge computing with computation capacity constraints. *IEEE Wireless Communications Letters*, *7*(3), 420–423. https://doi.org/10.1109/lwc.2017.2780128.
9. Kumar, K., Liu, J., Lu, Y.-H., & Bhargava, B. (2012). A survey of computation offloading for mobile systems. *Mobile Networks and Applications*, *18*(1), 129–140.
10. Liu, H., Eldarrat, F., Alqahtani, H., Reznik, A., de Foy, X., & Zhang, Y., (2018). Mobile edge cloud system: Architectures, challenges, and approaches. *IEEE Systems Journal*, *12*(3), 2495–2508.
11. Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., & Wang, W. (2017). A survey on mobile edge networks: convergence of computing, caching and communications. *IEEE Access*, *5*, 6757–6779.
12. Ceselli, A., Premoli, M., & Secci, S. (2017). Mobile edge cloud network design optimization. *IEEE/ACM Transactions on Networking*, *25*(3), 1818–1831. https://doi.org/10.1109/tnet.2017.2652850.
13. Mao, Y., Zhang, J., & Letaief, K. (2016). Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, *34*(12), 3590–3605.
14. You, C., Huang, K., & Chae, H. (2016). Energy efficient mobile cloud computing powered by wireless energy transfer. *IEEE Journal on Selected Areas in Communications*, *34*(5), 1757–1771. https://doi.org/10.1109/jsac.2016.2545382.
15. Xiang, X., Lin, C., & Chen, X. (2014). Energy-efficient link selection and transmission scheduling in mobile cloud computing. *IEEE Wireless Communications Letters*, *3*(2), 153–156.
16. Zhang, W., Wen, Y., Guan, K., Kilper, D., Luo, H., & Wu, D. (2013). Energy-optimal mobile cloud computing under stochastic wireless channel. *IEEE Transactions on Wireless Communications*, *12*(9), 4569–4581.
17. Du, J., Zhao, L., Feng, J., & Chu, X. (2018). Computation offloading and resource allocation in mixed fog/cloud computing systems with min–max fairness guarantee. *IEEE Transactions on Communications*, *66*(4), 1594–1608.
18. Guo, S., Liu, J., Yang, Y., Xiao, B., & Li, Z. (2019). Energy-efficient dynamic computation offloading and cooperative task

scheduling in mobile cloud computing. *IEEE Transactions on Mobile Computing*, *18*(2), 319–333.
19. Yang, L., Cao, J., Cheng, H., & Ji, Y. (2015). Multi-user computation partitioning for latency sensitive mobile cloud applications. *IEEE Transactions on Computers*, *64*(8), 2253–2266.
20. Chen, X., Jiao, L., Li, W., & Fu, X. (2016). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, *24*(5), 2795–2808.
21. Zhang, J., Hu, X., Ning, Z., Ngai, E., Zhou, L., Wei, J., et al. (2018). Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks. *IEEE Internet of Things Journal*, *5*(4), 2633–2645.
22. Dinh, T. Q., Tang, J., La, Q. D., & Quek, T. Q. S. (2017). Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Transactions on Communications*, *65*(8), 3571–3584.
23. Hao, Y., Chen, M., Hu, L., Hossain, M. S., & Ghoniem, A. (2018). Energy efficient task caching and offloading for mobile edge computing. *IEEE Access*, *6*, 11365–11373.
24. Ren, J., Yu, G., Ca, Y., & He, Y. (2018). Latency optimization for resource allocation in mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, *17*(8), 5506–5519.
25. Wang, Y., Sheng, M., Wang, X., Wang, L., & Li, J. (2016). Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. *IEEE Transactions on Communications*, *64*(10), 4268–4282.
26. Zhang, K., Mao, Y., Leng, S., Zhao, Q., Li, L., Peng, X., et al. (2016). Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. *IEEE Access*, *4*, 5896–5907.
27. Xiao, M., Lin, C., Han, Z., & Liu, J. (2018). Energy-aware computation offloading of IoT sensors in cloudlet-based mobile edge computing. *Sensors*, *18*(6), 1945.
28. Flores, H., Hui, P., Tarkoma, S., Li, Y., Srirama, S., & Buyya, R. (2015). Mobile code offloading: From concept to practice and beyond. *IEEE Communications Magazine*, *53*(3), 80–88. https://doi.org/10.1109/MCOM.2015.7060486.
29. Sophia, A. (2016). *ETSI mobile edge computing publishes foundation specifications [EB/OL]*. Retrieved from July 18, 2020 from http://www.etsi.org/index.php/news-events/news/1078-2016-04-etsi-mobile-edge-computing-publishes-foundation-specifications.
30. Joseph, A. D., deLespinasse, A. F., Tauber, J. A., Gifford, D. K., & Kaashoek, M. F. (1995). Rover: A toolkit for mobile information access. *ACM SIGOPS Operating Systems Review*, *29*(5), 156–171. https://doi.org/10.1145/224057.224069.
31. Kosta, S., Aucinas, A., Hui, P., Mortier, R., & Zhang, X. (2012). ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Proceedings IEEE Infocom* (pp. 945–953).
32. Burd, T. D., & Brodersen, R. W. (1996). Processor design for portable systems. *Journal of VLSI Signal Processing Systems for Signal Image & Video Technology*, *13*(2–3), 203–221.
33. Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*, 329–423.
34. Labbé, M., Laporte, G., & Martello, S. (2003). Upper bounds and algorithms for the maximum cardinality bin packing problem. *European Journal of Operational Research*, *149*(3), 490–498.
35. Johnson, S. M. (2006). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, *1*(1), 61–68.
36. Sarkar, T. K., Burintramart, S., Yilmazer, N., Hwang, S., Zhang, Y., De, A., et al. (2006). A discussion about some of the principles/practices of wireless communication under a Maxwellian framework. *IEEE Transactions on Antennas and Propagation*, *54*(12), 3727–3745. https://doi.org/10.1109/TAP.2006.886522.

Yue Zhang received the B.Eng. degree in automation from Shanghai University, Shanghai, China, in 2018, where he is currently pursuing the M.Eng. degree in control science and engineering. His research interests include mobile-edge computing, computation offloading algorithms, and game theory.

Jingqi Fu received the B.Sc. degree from the Northeast Heavy Machinery Institute, in 1984, the M.Sc. degree from Yanshan University, in 1989, and the Ph.D. degree from the Nanjing University of Science and Technology, in 1995. He is currently a Professor with Shanghai University. His current research interests include network measurement and control technology.