Stackelberg Game-Based Pricing and Offloading for the DVFS-Enabled MEC Systems

Jing Mei, Cuibin Zeng, Zhao Tong, *Senior Member, IEEE*, Zhibang Yang, and Keqin Li, *Fellow, IEEE*

***Abstract*—Due to the limited computing resources of both mobile devices (MDs) and the mobile edge computing (MEC) server, devising reasonable strategies for MD task offloading, MEC server resource pricing, and resource allocation is crucial. In this paper, a scenario is considered, comprising multiple MDs and a single MEC server. Each MD has a divisible task in each time slot, allowing for partial offloading and the option to discard parts of the task. The MEC server contains multiple computing units with the same computing power, and its computing resources can be dynamically adjusted through dynamic voltage and frequency scaling (DVFS) according to the size of tasks offloaded by MDs. At any given time slice, a Stackelberg game is formulated based on the strategies of the MDs and the strategy of the MEC server. An iterative evolution algorithm is employed to explore the optimal strategies for MDs and the MEC server. Simulation results demonstrate that both parties can reach an equilibrium state through the game, and these experiments confirm that the algorithm effectively enhances system efficiency.**

***Index Terms*—Dynamic voltage and frequency scaling, mobile edge computing, partial offloading, resource allocation, stackelberg game.**

I. Introduction

THE AMOUNT of data stored at the network edge has been rapidly increasing in recent years [1]. According to [2], it is projected that the global total data size will reach 175ZB by the year 2025. With the rapid growth of data, MEC has become one of the effective technologies for handling this enormous data. In this context, pushing computational capabilities towards the data source, namely the network edge, helps reduce latency, improve efficiency, and better meet the demands of the future digital society.

This paper explores a system model with two roles: multiple MDs, sometimes referred to as users, and a MEC server containing multiple computing units. Each user can choose from three task processing options: complete local processing, complete offloading processing, or partial local and partial offloading processing. The MEC server is located near the base station, receiving wireless task offloading requests from the base station. After processing the tasks, the MEC server returns the results to the base station, which then transmits them to the user. Users can reduce energy consumption and alleviate execution latency pressures by offloading tasks. Nevertheless, MEC server resources are limited, requiring users to obtain suitable resources based on their processing capabilities and task sizes. To better reflect real-world scenarios, MEC server evidently cannot unconditionally provide services. Instead, it will implement pricing for users to generate revenue. Currently, there is an observed phenomenon wherein the user's decision to offload tasks is significantly influenced by pricing, and this pricing varies according to the scale of offloaded tasks. Specifically, as the scale of offloaded tasks increases, users are confronted with corresponding price fluctuations. The challenge at hand revolves around enabling users to make more effective decisions regarding task offloading, while concurrently establishing a rational pricing mechanism for the MEC server. Addressing this issue is crucial in achieving an optimal balance between user decision-making and the formulation of reasonable price.

To tackle this concern, we utilize Stackelberg game theory featuring a solitary leader and numerous followers, aiming to find equilibrium between pricing strategies and offloading decisions. In this game, the MEC server assumes the role of the leader, while each user acts as a follower. The leader is responsible for formulating offloading price for followers, and the followers make offloading decisions based on the price. The aim of this paper is to attain a Stackelberg equilibrium between the MEC server and users via multiple game iterations.

In recent years, some studies have begun to explore the application of Stackelberg games. For example, Liu and Liu [3] modeled the interaction between edge cloud and users as a Stackelberg game. In this game, the edge cloud chooses to allocate limited computing resources equally to each user, and sets the price per unit cycle of data to optimize its income, while users reduce costs by making offloading decisions. Noreen et al. [4] proposed a novel framework

Received 31 January 2024; revised 3 August 2024 and 2 January 2025; accepted 25 February 2025. Date of publication 3 March 2025; date of current version 9 June 2025. This work was supported by the Program of National Natural Science Foundation of China (Grant No. 62372172), Distinguished Youth Science Foundation of Hunan Province, China (Grant No. 2023JJ10030), Research Foundation of Education Bureau of Hunan Province, China (Grant No. 23B0100). The associate editor coordinating the review of this article and approving it for publication was J. J. Yang. *(Corresponding author: Zhao Tong.)*

Jing Mei, Cuibin Zeng, and Zhao Tong are with the College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China (e-mail: jingmei1988@163.com; zeng1941183190@gmail.com; tongzhao@hunnu.edu.cn).

Zhibang Yang is with the Hunan Province Key Laboratory of Industrial Internet Technology and Security, Changsha University, Changsha 410022, Hunan, China (e-mail: yangzb@ccsu.edu.cn).

Keqin Li is with the College of Information Science and Engineering, Hunan University, Changsha 410012, China, also with the Ministry of Science and Technology, PRC, National Supercomputing Center, Changsha 410082, Hunan, China, and also with the Department of Computer Science, State University of New York, New York, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TNSM.2025.3547568

for device-to-device (D2D) communication power control and employed a Stackelberg game to jointly optimize the utility of D2D users and the cellular base station (CBS). Chen et al. [5] introduced a stratified structure and devised a discrete Stackelberg game with multiple leaders and followers to minimize energy consumption. The aforementioned studies employed game theory to rationally adjust resource allocation between MEC servers and users, yet they did not take into account the frequency adjustment of MEC server. Whether it is the user or the MEC server, when dealing with a smaller number of tasks, although utilizing the full computational resources can shorten the task completion time, it also incurs significant energy consumption. In this study, our primary concern is to ensure that tasks are completed within the specified deadline. Since the benefits of both users and MEC server in this paper are associated with energy consumption, optimizing energy consumption while meeting deadlines will enhance the mutual benefits. DVFS is adopted in our study. By integrating DVFS, we are able to adjust the voltage and frequency of the processor based on actual load conditions at runtime to strike a balance between energy consumption and performance. This enables MDs and MEC server to manage energy more intelligently to achieve better performance and energy efficiency under different load conditions. Although the introduction of DVFS can effectively optimize energy consumption, the simultaneous consideration of pricing and frequency adjustments adds complexity to the problem. Due to the limited resources of the MEC server, it must strike a balance between pricing and frequency to ensure that all tasks can be executed effectively within the available resource limits. This presents a new challenge for the optimization of pricing strategy and resource allocation. Therefore, our algorithm aims to address this challenge by formulating reasonable pricing and resource allocation to optimize the utility of the MEC server and users.

The primary contributions of this research entail:

- Fixed computation frequencies can lead to inefficient resource allocation and higher energy consumption. Implementing a dynamic frequency adjustment mechanism based on user task offloading decisions and deadlines allows the MEC system to better optimize resource utilization and reduce energy consumption.
- By employing a Stackelberg game, we jointly optimize task offloading decisions, resource allocation, and resource pricing, allowing both users and the MEC server to dynamically adjust their strategies.
- We utilize the spy-adjust dynamic iterative search algorithm to iteratively update the price, computation frequencies, and offloading decisions during the gaming process. This approach aims to dynamically adjust these parameters to achieve suboptimal solutions that closely approximate the global optimum, thereby optimizing the utility of the MEC server and users.

Following this introduction, the subsequent sections of this paper are structured as follows. Section II offers an elaborate discussion on the pertinent research explored within this study. Section III defines the system model. Section IV presents the studied optimization problem and models the relationship between users and MEC server as a Stackelberg game. Section V analyzes the decision-making process between users and MEC server. Section VI introduces the spy-adjust dynamic iterative search algorithm. Section VII experimentally validates the algorithm's performance. Section VIII provides a conclusion of the paper's contents. The final section shows the acknowledgments.

II. The Related Work

Numerous studies have delved into the task offloading conundrum, primarily from the user's standpoint. Various algorithms are proposed to assist users in devising optimal strategies for both task offloading and resource allocation.

Chen et al. [6] conducted a study on the energy-efficient dynamic offloading challenge within the framework of MEC in the Internet of things (IoT). Starting from the users' perspective, they proposed the objective of minimizing mean transmission energy expenditure while ensuring devices' efficiency. Yang et al. [7] developed a distributed task node-to-helper node pairing and offloading mechanism that only leveraged local information, with the objective of minimizing the delay for each task. Fang et al. [8] investigated the problem of computation task migration among numerous users with intensive computational needs. They introduced an enhanced response-based distributed multi-user computation task offloading algorithm, denoted as BR-DMCTO, to tackle this optimization challenge.

The research in literature [6], [7], [8] all focused on optimizing the offloading problem on the user side, they did not fully consider the benefits of the MEC server side. Many studies in recent years have included the benefits of MEC servers into the consideration of system performance. Chen et al. [9] introduced a framework utilizing Stackelberg games, where users and MEC servers assume the roles of followers and leaders, respectively. The objective of this framework is to derive a solution at Stackelberg equilibrium. Yao et al. [10] explored the resource management and pricing between cloud provider and miners, framing their interaction as a Stackelberg game. Zeng et al. [11] proposed a volunteer task assignment algorithm aimed at maximizing the reward for volunteer vehicles. Most of the aforementioned literature incorporates the benefits of both users and MEC servers into the consideration of system performance. Liu et al. [12] addressed the partial offloading challenge from vehicles tasked with computing-intensive and delay-sensitive operations to the vehicular edge computing (VEC) server, enhancing vehicle services. They introduced a distributed algorithm based on a multi-leader and multi-follower Stackelberg game to optimize vehicle and VEC server utilities while adhering to latency constraints. Li et al. [13] introduced a computation offloading mechanism utilizing a two-stage Stackelberg game, which determines the appropriate price for the computation resource of the edge clouds to maximize their profit, and the computational needs of IIoT devices to enhance their utility through the incorporation of social interaction information from prospective IIoT devices. Zhou et al. [14] explored the collaborative dynamics between cloud servers (CS) and edge

servers (ESs), where CS offloads computation tasks to ECs possessing unused computational resources, thereby mitigating its own cost and pressure. In order to maximize the utility of both sides, the interaction between the CS and ESs is modeled as a Stackelberg game. Zhou et al. [15] investigated the computation offloading problem of a UAV-aided MEC network, including one UAV-MEC server, one BS-MEC server, and several MUs. The two servers are managed by the edge service provider (ESP) and provide the idle resources to MUs to make a profit. A Stackelberg game is employed to represent the interaction between the ESP and MUs with the aim of maximizing their utility. Tao [16] investigated the resources pricing strategy for single MEC server and task offloading strategy for multi-users. A Stackelberg game is established to result in the win-win situation.

Above studies apply the Stackelberg game to address the task offloading and resource pricing strategies for different application scenarios, however, these studies presume that MEC server ability is fixed, and the resource pricing is the only factor to be optimized for the MEC server. Hence, these studies are predominantly single-objective optimizations for MEC servers, primarily focusing on price optimization, while neglecting the optimization on the computation resources. Considering that adjustable frequency can impact the energy consumption of MEC servers, the CPU frequency can be dynamically adjusted according to the computation requirement of users to enhance the resource utilization efficiency of MEC servers and reduce the energy consumption effectively, so its inclusion in the consideration is of paramount significance. Hence, in this research, we consider the joint optimization of task offloading for users, and CPU frequency scaling and resource pricing for the MEC server to improve the benefit of both parties on the premise of win-win.

III. The Models

In the section, some models related to this problem are introduced firstly. Based on the models, the pricing and offloading problem can be established and analyzed strictly.

A. System Model

This paper investigates an MEC system featuring a single antenna and accommodating multiple users, alongside a single base station (BS), as illustrated in Fig. 1. An MEC server, enabled with DVFS, deployed at the BS, offers services to the users. The MEC server can adjust the frequency dynamically according to the system load for energy saving. Let $\mathcal{N} = \{1, 2, \ldots, N\}$ denote the N users within the coverage of the BS, which compete for the communication resources and the MEC computation resources. Because of the system's dynamic nature, we segment time into a series of equal-length time slices τ. The system state is assumed to be pseudo-static during each time slice. Each user generates a new computational task during each time slice, and the generated tasks would be processed at the following time slice. Due to the size of user devices is limited, each user device is equipped with a small capacity battery and an energy harvesting model which can harvest energy from the outside world.

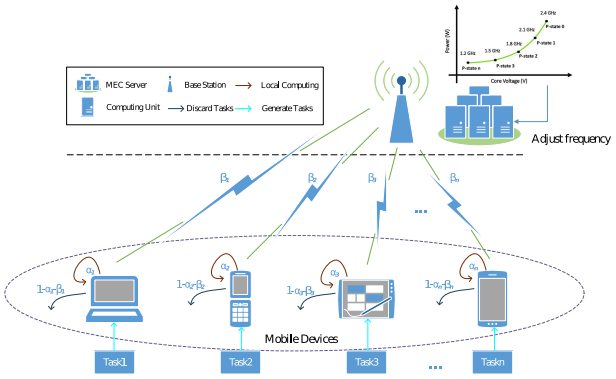


Fig. 1. MEC system model.

Let $\{C_n, D_n\}$ denote the task generated by user n where C_n (in bits) presents the task amount and D_n (in cycles/bit) is a pre-defined constant representing the number of CPU cycles required to process one bit of task for user n. The task can be split into two components, allowing for concurrent execution locally and offloading to MEC.

B. Local Computation Model

For the n-th user, the local task size is represented by $\alpha_n C_n$, where $\alpha_n \in [0, 1]$. The CPU frequency of user n is denoted as f_n, which can be adjusted during the range of $[0, f_{\max}]$. Then, the time expenditure of user n for local computation can be expressed as

$$T_n = \frac{\alpha_n C_n D_n}{f_n}.$$

Since the task should be completed before the end of a time slot, it should satisfy

$$\frac{\alpha_n C_n D_n}{f_n} \leq \tau.$$

Let P_n be the power consumed for computation (in Watts) of user n. Generally, P_n depends on the chip architecture of the user, and it is proportional to CPU frequency, which is $P_n = \xi_n f_n^3$ where ξ_n is a coefficient that relies on chip architecture [17], [18], [19]. The energy expenditure of user n for local computation is

$$E_n^{\text{loc}} = P_n T_n = \xi_n f_n^2 \alpha_n C_n D_n.$$

C. Remote Offloading Model

Let $\beta_n C_n$ be the size of the task offloaded to MEC, where $\beta_n \in [0, 1]$. The task offloading consists of two phases: task transmission and remote computation.

Task Transmission. Let P_n^{tra} be the transmit power (in Watts) of user n. The achievable data rate (in bits/s) of user n is

$$R_n = B_n \log_2 \left(1 + \frac{P_n^{\text{tra}} h_n}{\sigma^2}\right),$$

where B_n denotes the bandwidth that the n-th user occupies, h_n is the channel power gain [20], and σ^2 is the noise power spectral density [21], [22], [23]. This paper assumes

equal bandwidth allocation among all users, so $B_n = B/N$, where B is the total bandwidth of the wireless communication resources.

For the n-th user, the transmit delay is

$$T_n^{\mathrm{tra}} = \frac{\beta_n C_n}{R_n},$$

and the transmit energy consumption is

$$E_n^{\mathrm{tra}} = P_n^{\mathrm{tra}} T_n^{\mathrm{tra}} = \frac{P_n^{\mathrm{tra}} \beta_n C_n}{R_n}.$$

Due to the limited battery capacity, each user's energy expenditure during a time slice must not exceed its residual energy, that is,

$$E_n^{\mathrm{loc}} + E_n^{\mathrm{tra}} \leq E_n^{\mathrm{res}},$$

where E_n^{res} denotes the rest energy of user n.

Remote Computation. After the users delegate the tasks to the MEC server, MEC should allocate enough computation resource to each user to guarantee that all tasks are completed in current time slice. Let f_n^{mec} be the CPU frequency that MEC allocates to the n-th user. Then, the remote computation delay of user n is

$$T_n^{\mathrm{mec}} = \frac{\beta_n C_n D_n}{f_n^{\mathrm{mec}}}.$$

This study examines the MEC server equipped with multiple computing units of equal processing power. The number of units is specified as m and the frequency of each unit is specified as f_{unit} which can be adjusted during $[0, F_{\max}]$. Hence, the total computation capacity of the MEC server is mf_{unit}, and the sum computation capacity of MEC allocated to all users should satisfy

$$\sum_{n=1}^{N} f_n^{\mathrm{mec}} = mf_{\mathrm{unit}}.$$

The energy consumption E_{mec} generated by the MEC server in processing all user offload tasks is as follows:

$$E_{\mathrm{mec}} = mP_{\mathrm{mec}}\tau,$$

where $P_{\mathrm{mec}} = \xi f_{\mathrm{unit}}^3$ and ξ is a coefficient depending on chip architecture of the MEC server.

According to the delay constraint, the sum delay of the two phases should satisfy

$$T_n^{\mathrm{tra}} + T_n^{\mathrm{mec}} \leq \tau, \forall n \in \mathcal{N}.$$

IV. Problem Formulation as a Stackelberg Game

In this problem, the users and the MEC server make their own decisions, respectively. The users make the task offloading decisions and the corresponding frequencies from their own interests based on the resource pricing given by the MEC server. For the MEC server, it provides computation resources to serve the users to earn income. To maximize the profit, the resource pricing and frequency should be adjusted based on the task offloading decisions of all users. Since the decision making of the users and the MEC server are interdependent, we formulate the problem as the Stackelberg's leadership model [24], in which the MEC server and the users are regarded as *leader* and *follower*, respectively.

A. Utility of Users

The utility function of the users in a Stackelberg game reflects the satisfaction level of them with the current game. The satisfaction level is always related to the processed task amount, the energy consumption, and the charge paid to the MEC server.

As mentioned before, the local executed task amount of user n is $\alpha_n C_n$, the offloaded task amount is $\beta_n C_n$, and it is satisfied that $\alpha_n + \beta_n \leq 1$. In this paper, not all tasks have to be handled, that is, the users can discard part of the computation tasks to achieve the maximum benefit. The revenue gain of user n in terms of the computation task amount is measured using a logarithmic function [21] as

$$S_n = \omega_n \ln(1 + (\alpha_n + \beta_n) C_n),$$

where ω_n denotes as the satisfaction factor for the user.

For user n, it offloads $\beta_n C_n$ bits of task to the MEC server, and pays it the corresponding fee. Let p be the resource pricing (in cent/cycle) given by MEC, the total fee that user n should pay to MEC is $p\beta_n C_n D_n$ [1]. Hence, the utility function of user n is specified as

$$U_n(f_n, \alpha_n, \beta_n) = \gamma_n(\omega_n \ln(1 + (\alpha_n + \beta_n) C_n)) - \overline{\gamma}_n p \beta_n C_n D_n,$$

where γ_n is the trade-off factor between the revenue and the cost, and $\overline{\gamma}_n = 1 - \gamma_n$. In this paper, γ_n is set as 0.5.

B. Utility of MEC

For the MEC server, it provides computation resources to the users to obtain profit. The revenue of the MEC server is calculated as

$$R_{\mathrm{mec}} = p \sum_{n=1}^{N} \beta_n C_n D_n,$$

where p is the unit price that MEC charges each user.

As mentioned before, the MEC server generates energy consumption to maintain its operation, which is its main cost. Let η be the electricity price (in cents/J), the electricity cost per time slice is calculated as

$$C_{\mathrm{mec}} = \eta E_{\mathrm{mec}} = m\xi f_{\mathrm{unit}}^3 \tau \eta.$$

Hence, the utility function of MEC is formulated as

$$U_{\mathrm{mec}}(p, f_{\mathrm{unit}}) = p \sum_{n=1}^{N} \beta_n C_n D_n - m\xi f_{\mathrm{unit}}^3 \tau \eta. \quad (1)$$

C. Utility Optimization Problem Formulation

1) Follower Level:

$$(\mathbf{P1}) \quad \max_{f_n, \alpha_n, \beta_n} \ \gamma_n(\omega_n \ln(1 + (\alpha_n + \beta_n) C_n)) - \overline{\gamma}_n p \beta_n C_n D_n,$$

$$\text{s.t.} \quad \frac{\alpha_n C_n D_n}{f_n} \leq \tau, \quad (2a)$$

$$\frac{\beta_n C_n}{R_n} + \frac{\beta_n C_n D_n}{f_n^{\mathrm{mec}}} \leq \tau, \quad (2b)$$

$$\alpha_n + \beta_n \le 1, \tag{2c}$$
$$0 \le f_n \le f_{\max}, \tag{2d}$$
$$\xi_n f_n^2 \alpha_n C_n D_n + \frac{P_n^{\mathrm{tra}} \beta_n C_n}{R_n} \le E_n^{\mathrm{res}}, \tag{2e}$$
$$\alpha_n \ge 0, \beta_n \ge 0. \tag{2f}$$

The objective of the follower level problem is to maximize the utility of the users which reflects the satisfaction level of task execution and the service charge. The optimization variables are α_n, β_n and f_n which denote the offloading strategy and the resource management strategy of user n. In constraint (2a) and (2b), the time consumption for processing tasks locally and executing tasks remotely are respectively constrained. Constraint (2c) limits the total amount of task to be processed. Constraint (2d) is the value range of the frequency that the users can be adjusted. In Eq. (2e), the combined energy usage for local computation and task transmission must not surpass the remaining energy.

2) Leader Level:

$$(\mathbf{P2}) \max_{p, f_{\mathrm{unit}}} \quad U_{\mathrm{mec}} = p \sum_{n=1}^{N} \beta_n C_n D_n - m \xi f_{\mathrm{unit}}^3 \tau \eta,$$
$$\text{s.t.} \quad \sum_{n=1}^{N} f_n^{\mathrm{mec}} = m f_{\mathrm{unit}}, \tag{3a}$$
$$0 \le f_{\mathrm{unit}} \le F_{\max}. \tag{3b}$$

In the leader level problem, the objective is to maximize the profit of the MEC server through reasonable pricing and frequency setting. Constraint (3a) denotes the resources allocated to all users cannot surpass the computation ability of MEC. Constraint (3b) shows that the MEC server should adjust the computing unit frequency during the reasonable range.

V. Strategies Analysis

In the Stackelberg game, the anticipants know the fully information about each others. The leader first announces its initial strategies in terms of resource pricing and unit frequency setting. The followers then observe the leader's strategy and make optimal decisions regarding task offloading and frequency setting. By predicting the response of the followers, the leader can adjust its strategies, correspondingly. After several rounds of games, the formulated problem can be solved by finding a Stackelberg equilibrium [11], which is defined as follows:

Definition 1: Let $(f_n^*, \alpha_n^*, \beta_n^*)$ be the optimal strategy of the n-th follower, and $(p^*, f_{\mathrm{unit}}^*)$ be the optimal strategy of the leader. The optimal solution $(f_n^*, \alpha_n^*, \beta_n^*, p^*, f_{\mathrm{unit}}^*)$ is the Stackelberg equilibrium, if the following conditions are satisfied:

$$U_n(f_n^*, \alpha_n^*, \beta_n^*, p^*, f_{\mathrm{unit}}^*) \ge U_n(f_n, \alpha_n, \beta_n, p^*, f_{\mathrm{unit}}^*), \forall n,$$
$$U_{mec}(f_n^*, \alpha_n^*, \beta_n^*, p^*, f_{\mathrm{unit}}^*) \ge U_{mec}(f_n^*, \alpha_n^*, \beta_n^*, p, f_{\mathrm{unit}}).$$

A. Strategies at the Follower Level

To solve the follower level problem, the strategy of the leader should be known. Let p be the resource pricing given by MEC. In order to solve problem **(P1)**, we first simplify the problem by some deformation as follows:

- According to constraint (2d), the user frequency can be adjusted between 0 and $f_{\max}$, hence, the task executed locally cannot exceed the maximal computation capacity of the users, that is,
$$0 \le \alpha_n \le \frac{\tau f_{\max}}{C_i D_i}.$$
- Once α_i is determined, the optimal frequency of the n-th user f_n can be calculated as $\alpha_n C_n D_n / \tau$. Substituting $f_n = \alpha_n C_n D_n / \tau$ into constraint (2e), it can be rewritten as
$$\alpha_n^3 \frac{\xi_n C_n^3 D_n^3}{\tau^2} + \beta_n \frac{P_n^{tra} C_n}{R_n} \le E_n^{\mathrm{res}}.$$
- Analyzing constraint (2b), to make the remote offloading delay satisfy the time constraint, it must satisfy that $\beta_n C_n / R_n \le \tau$. Combining with constraint (2f), we have
$$0 \le \beta_n \le \frac{\tau R_n}{C_n}.$$
In this premise, the MEC server should allocate sufficient computation resources to complete the task before the deadline τ. This is ensured by the server's price dynamic adjustment strategy (in Alg. 3 and Alg. 4). If the MEC server's resources are insufficient to complete the tasks by the deadline, the server will increase price to reduce the task offloading, thereby ensuring that tasks are completed within the deadline. The required computation resources are given by:
$$f_n^{\mathrm{mec}} = \frac{D_n}{\frac{\tau}{\beta_n C_n} - \frac{1}{R_n}}. \tag{4}$$

According to the preceding analysis, the original problem **(P1)** can be rewritten as

$$(\mathbf{P3}) \max_{\alpha_n, \beta_n} \quad \gamma_n(\omega_n \ln(1 + (\alpha_n + \beta_n) C_n)) - \overline{\gamma}_n p \beta_n C_n D_n,$$
$$\text{s.t.} \quad 0 \le \alpha_n \le \frac{\tau f_{\max}}{C_i D_i}, \tag{5a}$$
$$0 \le \beta_n \le \frac{\tau R_n}{C_n}, \tag{5b}$$
$$\alpha_n^3 \frac{\xi_n C_n^3 D_n^3}{\tau^2} + \beta_n \frac{P_n^{tra} C_n}{R_n} \le E_n^{\mathrm{res}}, \tag{5c}$$
$$\alpha_n + \beta_n \le 1. \tag{5d}$$

Theorem 1: Problem **(P3)** is a convex optimization problem.

Proof: The first-order partial derivative of the utility function U_n with respect to α_n is

$$\frac{\partial U_n(\alpha_n, \beta_n)}{\partial \alpha_n} = \frac{\gamma_n \omega_n C_n}{1 + (\alpha_n + \beta_n) C_n},$$

and the second-order partial derivative of U_n with respect to α_n is

$$\frac{\partial^2 U_n(\alpha_n, \beta_n)}{\partial \alpha_n^2} = -\frac{\gamma_n \omega_n C_n^2}{(1 + (\alpha_n + \beta_n) C_n)^2}.$$

Similarly, taking the first-order and second-order partial derivatives of U_n with respect to β_n, we have

$$\frac{\partial U_n(\alpha_n,\beta_n)}{\partial \beta_n} = \frac{\gamma_n \omega_n C_n}{1+(\alpha_n+\beta_n)C_n} - \overline{\gamma}_n p C_n D_n,$$

and

$$\frac{\partial^2 U_n(\alpha_n,\beta_n)}{\partial \beta_n^2} = -\frac{\gamma_n \omega_n C_n^2}{(1+(\alpha_n+\beta_n)C_n)^2}.$$

Next, we calculate the mixed partial derivatives of α_n and β_n.

$$\frac{\partial^2 U_n(\alpha_n,\beta_n)}{\partial \alpha_n \beta_n} = -\frac{\gamma_n \omega_n C_n^2}{((1+(\alpha_n+\beta_n)C_n)^2},$$
$$\frac{\partial^2 U_n(\alpha_n,\beta_n)}{\partial \beta_n \alpha_n} = -\frac{\gamma_n \omega_n C_n^2}{((1+(\alpha_n+\beta_n)C_n)^2}.$$

By combining the second-order partial derivatives into the Hessian matrix and calculating the eigenvalues, it can be easily verified that the matrix is negative semi-definite (due to space limitations, the specific process of solving the Hessian matrix is not presented in this paper). Since the objective function is jointly concave, maximizing the concave function is equivalent to minimizing its negative value, which is a convex function. Therefore, problem **(P3)** is a convex optimization problem. ■

According to **Theorem 1**, problem **(P3)** is a convex optimization problem. We use CVX, the convex optimization tool in MATLAB, to solve it, and the optimal task offloading scheme (α_n, β_n) under the given price p can be obtained for each user.

B. Strategies at the Leader Level

When the users determine their offloading schemes, the MEC server should adjust its decision on the resource pricing and the frequency scaling to maximize its utility.

Since all offloaded tasks should be completed before the given deadline, the computation resources that the MEC server should allocate to each user can be calculated according to Eq. (4). Hence, to satisfy the delay requirements of all users, the computing unit frequency of the MEC server should be adjusted to

$$f_{\text{unit}} = \frac{1}{m}\sum_{n=1}^{N} f_n^{\text{mec}} = \frac{1}{m}\sum_{n=1}^{N} \frac{\beta_n C_n D_n R_n}{\tau R_n - \beta_n C_n}, \tag{6}$$

where m is the number of computing units.

However, the maximal frequency that the unit can be adjusted to is $F_{\max}$. If the objective frequency f_{unit} is higher than the maximal frequency $F_{\max}$, i.e., $f_{\text{unit}} > F_{\max}$, the system load exceeds the maximal computation capacity. In the case, the only choice for the MEC server is to raise pricing to reduce the offloading requirements of users. The price is updated as

$$p = p + \rho\Delta, \tag{7}$$

where ρ denotes the price update step, and Δ denotes the price update direction (1 and -1), and $\Delta = 1$ here.

If $f_{\text{unit}} \le F_{\max}$, substituting Eq. (6) into Eq. (1), the utility function of MEC can be updated as

$$U_{\text{mec}}(p) = p\sum_{n=1}^{N}\beta_n C_n D_n - \frac{\xi\tau\eta}{m^2}\left(\sum_{n=1}^{N}\frac{\beta_n C_n D_n R_n}{\tau R_n - \beta_n C_n}\right)^3. \tag{8}$$

The problem **(P2)** can be rewritten as

$$\textbf{(P4)} \quad \max_{p} \quad p\sum_{n=1}^{N}\beta_n C_n D_n - \frac{\xi\tau\eta}{m^2}\left(\sum_{n=1}^{N}\frac{\beta_n C_n D_n R_n}{\tau R_n - \beta_n C_n}\right)^3,$$
$$\text{s.t.} \quad 0 \le \frac{1}{m}\sum_{n=1}^{N}\frac{\beta_n C_n D_n R_n}{\tau R_n - \beta_n C_n} \le F_{\max}. \tag{9a}$$

After problem deformation, the utility of MEC is related with the offloading decisions of users $\{\beta_1,\ldots,\beta_N\}$, and the offloading decisions of users are further relying on the resource pricing p, so the utility of MEC only depends on the resource pricing p indirectly.

To find the suboptimal pricing, we adopt a spy-adjust strategy. Firstly, the MEC server gives an initial pricing, the users determine their optimal offloading decisions under the pricing. And then, the MEC server makes the corresponding adjustment on the pricing and the frequency to improve its utility based on the decisions of users. After that, the users update their decisions again based on the updated pricing. This process is repeated until the decisions of both parties are no longer changing, that is, the Stackelberg equilibrium is found.

C. Stackelberg Equilibrium Analysis

In order to verify whether a joint optimal solution exists for the utility optimization problems of users and MEC server, this section employs Stackelberg equilibrium for the proof. Since Section V-A shows that the distribution of f_n and f_n^{mec} depends on α_n and β_n respectively, we only need to consider whether there is a Stackelberg equilibrium between (α_n, β_n) and p.

Theorem 2: If the resource price p of MEC server is fixed, then the user's utility function $U_n(\alpha_n, \beta_n)$ reaches the maximum value at (α_n^*, β_n^*).

Proof: According to **Theorem 1**, the optimal solution (α_n^*, β_n^*) can be solved through the convex optimization toolkit to maximize $U_n(\alpha_n, \beta_n)$. ■

Theorem 3: Let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n, \ldots, \alpha_N)$, $\beta = (\beta_1, \beta_2, \ldots, \beta_n, \ldots, \beta_N)$. If users' offloading decisions (α, β) are determined, then the utility function $U_{mec}(p)$ of the MEC server is maximized at resource pricing $p*$.

Proof: According to Alg. 1 and Alg. 3, the range of unit price of MEC server resources (p_{low}, p_{up}) is determined. Let s represent the traversal step size and $i \in \{0, 1, 2, \ldots, (p_{up} - p_{low})/s\}$. By traversing $(p_{low} + s \times i)$ within the price range, the MEC server calculates utilities based on user offloading decisions. As shown in Fig. 2, the MEC server can determine the optimal resource price to maximize the $U_{mec}(p)$. ■

According to **Theorem 2** and **Theorem 3**, we know that when the MEC server prices resources as p^*, users will make

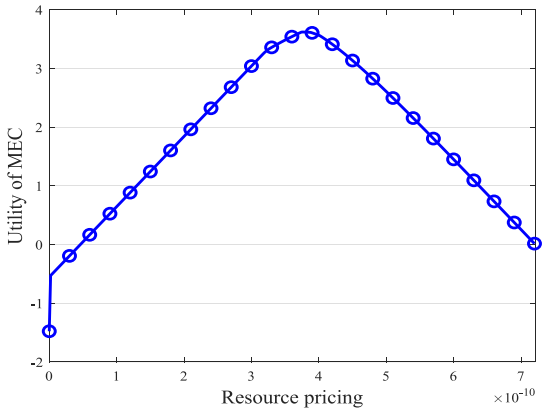


Fig. 2. Variation in MEC server utility with changing resource prices.

Algorithm 1: Price Upper Bound Solution

Input: ϵ;
Output: the price up bound p_{up};
1 Initialize the price range $[p_{\text{left}}, p_{\text{right}}]$ where $U_{\text{mec}}(p_{\text{left}}) \neq 0$ and $U_{\text{mec}}(p_{\text{right}}) = 0$ by Alg. 2;
2 **while** $p_{right} - p_{left} > \epsilon$ **do**
3 $\quad p_{\text{mid}} \leftarrow (p_{\text{left}} + p_{\text{right}})/2$;
4 $\quad$ **if** $U_{mec}(p_{mid}) = 0$ **then**
5 $\quad\quad p_{\text{right}} \leftarrow p_{\text{mid}}$;
6 $\quad$ **else**
7 $\quad\quad p_{\text{left}} \leftarrow p_{\text{mid}}$;
8 $\quad$ **end**
9 **end**
10 $p_{\text{up}} \leftarrow p_{\text{left}}$;

Algorithm 2: MEC Frequency and Benefit Solution

Input: the resource pricing p;
Output: the computing unit frequency f_{unit}, the utility of MEC U_{mec};
1 $\{(\alpha_n, \beta_n) | n \in (1, \ldots, N)\} \leftarrow$ solve the follower level problem for all users;
2 $f_{\text{unit}} \leftarrow$ calculate the computing unit frequency by Eq. (6);
3 $U_{\text{mec}} \leftarrow$ calculate the MEC utility by Eq. (8);

Algorithm 3: Price Lower Bound Solution

Input: ϵ;
Output: the price low bound p_{low};
1 Initialize the price range $[p_{\text{left}}, p_{\text{right}}]$ as $[0, p_{\text{up}}]$;
2 $f_{\text{unit}} \leftarrow$ calculate the objective computing unit frequency under the price p_{left} by Alg. 2;
3 **if** $f_{unit} < F_{max}$ **then**
4 $\quad p_{\text{low}} \leftarrow p_{\text{left}}$;
5 **else**
6 $\quad$ **while** $p_{right} - p_{left} > \epsilon$ **do**
7 $\quad\quad p_{\text{mid}} \leftarrow (p_{\text{left}} + p_{\text{right}})/2$;
8 $\quad\quad$ **if** $f_{unit}(p_{mid}) > F_{max}$ **then**
9 $\quad\quad\quad p_{\text{left}} \leftarrow p_{\text{mid}}$;
10 $\quad\quad$ **else**
11 $\quad\quad\quad p_{\text{right}} \leftarrow p_{\text{mid}}$;
12 $\quad\quad$ **end**
13 $\quad$ **end**
14 $\quad p_{\text{low}} \leftarrow p_{\text{right}}$;
15 **end**

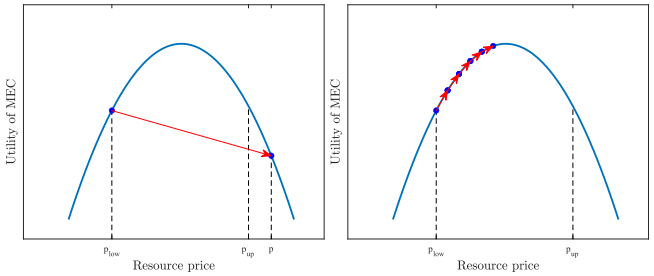
offloading decisions (α^*, β^*) to achieve optimal utilities. In this case, neither the MEC server nor users can obtain better benefit by adjusting the strategies, which is consistent with **Definition 1**. Therefore, there is a Stackelberg equilibrium between the MEC server and users.

VI. Spy-Adjust Dynamic Iterative Search Algorithm

In this section, a Dynamic Iterative Search Algorithm based on a Spy-Adjust Strategy is proposed to get the unique Stackelberg equilibrium. Before introducing the algorithm, we firstly introduce two sub-algorithms Alg. 1 and Alg. 3. In the two algorithms, the binary search strategy is adopted to find the price up bound and price low bound, respectively.

Alg. 1 introduces the binary search method to find the up bound of the resource pricing. The price up bound means, once the resource pricing exceeds the up bound, the users no longer offload any tasks to MEC, hence, the utility of the MEC server declines to 0 and remains unchanged. In the algorithm, the utility of MEC is calculated by calling Alg. 2 (in lines 1 and 3).

Similarly, the price low bound can be found adopting the binary search method as well, shown as Alg. 3. If the resource pricing is lower than the price low bound, the users tend to offload a great amount of tasks to the MEC server, which leads to the MEC server overload and the performance of the tasks cannot be guaranteed.

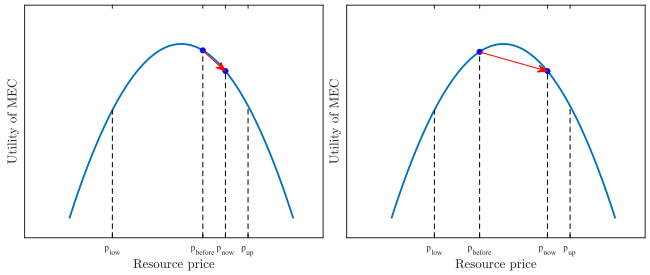


(a) Out of range (b) Normal update of price

Fig. 3. Price update indication.

Next, we will give a detailed introduction on the proposed algorithm.

The price is initialized as 0 and progressively approximates the optimal value by adjustment. In the algorithm, the price update step ρ is randomly set, and the price update direction is initialized to 1, that is, the MEC server raises the price by default. Because ρ is randomly set, the updated price might be out of the effective price range $[p_{\text{low}}, p_{\text{up}}]$ (shown in Fig. 3(a)). Under the case, the ρ value is halved iteratively until the updated price falls within the effective range (shown as lines 5-8).



(a) Opposite update direction (b) Large update step

Fig. 4. Utility decline reason analysis.

Each time after the price is updated, the MEC utility is calculated under the updated price p'_n, denoted as U'_{mec}. If U'_{mec} is greater than U_{mec} (see Fig. 3(b)), the price is further updated in the original direction (lines 23-30). If U'_{mec} is less than U_{mec}, it should be discussed in two cases.

Fig. 4(a) gives the first case, that is, the price update goes in the opposite direction. Fig. 4(b) gives the second case, that is, the price update step is too large. To confirm the reason for the decline in utility, a spying tactic is employed. In lines 13-14, we update the price in the opposite direction and calculate the utility U''_{mec}. If U''_{mec} is greater than U_{mec}, adjust the price update direction (in line 16). Otherwise, halve the price update step (in lines 18-20).

Since we focus on the computation offloading problem of the dynamic system where many parameters change over time, our algorithm should make snap decisions on the offloading decision and resource allocation scheme per time slot. Alg. 4 represents our final proposed algorithm. The spy-adjust algorithm iteratively adjusts the step size and update direction multiple times to optimize the pricing strategy of the MEC server, and users can formulate reasonable offloading strategies based on the price. Let K denote the number of game iterations between MEC server and users, and V represent the time complexity of solving a convex optimization problem using the toolbox. In summary, the time complexity of our proposed Alg. 4 is $O(KNV)$.

Algorithm 4: Spy-Adjust Dynamic Iterative Search Algorithm (SADISA)

Input: device parameters such as N, γ_n, ω_n, C_n, D_n, P_n^{tra}, ξ_n, E_n^{res},$f_{\max}$; channel parameters such as B, h_n, σ^2; MEC parameters such as m, $F_{\max}$; price update step ρ;

Output: the suboptimal decision (p^*, f^*_{unit}) for MEC, and the optimal decision $(\alpha^*_n, \beta^*_n, f^*_n)$ for each user;

1 $p_{\text{up}} \leftarrow$ find the up bound of the pricing by Alg. 1;
2 $p_{\text{low}} \leftarrow$ find the low bound of the pricing by Alg. 3;
3 $p \leftarrow 0, \Delta \leftarrow 1, \theta \leftarrow 10^{-10}$;
4 $U_{\text{mec}} \leftarrow$ calculate the MEC utility under p by Alg. 2;
5 $p' \leftarrow p + \rho\Delta$;
6 **while** $p' > p_{up}$ **do**
7 $\quad \rho \leftarrow \rho/2$;
8 $\quad p' \leftarrow p + \rho\Delta$;
9 **end**
10 $U'_{\text{mec}} \leftarrow$ calculate the MEC utility under p' by Alg. 2;
11 **while** $|U'_{mec} - U_{mec}| > \epsilon$ **do**
12 $\quad$ **if** $U'_{mec} < U_{mec}$ **then**
13 $\quad\quad p'' \leftarrow p - \theta\Delta$;
14 $\quad\quad U''_{\text{mec}} \leftarrow$ calculate the MEC utility under p'' by Alg. 2;
15 $\quad\quad$ **if** $U''_{mec} > U_{mec}$ **then**
16 $\quad\quad\quad \Delta \leftarrow -\Delta$;
17 $\quad\quad$ **else**
18 $\quad\quad\quad \rho \leftarrow \rho/2$;
19 $\quad\quad\quad p' \leftarrow p + \rho\Delta$;
20 $\quad\quad\quad U'_{\text{mec}} \leftarrow$ calculate the MEC utility under p' by Alg. 2;
21 $\quad\quad$ **end**
22 $\quad$ **else**
23 $\quad\quad \mathcal{D} = (U'_{\text{mec}} - U_{\text{mec}})/(p' - p)$;
24 $\quad\quad \Delta \leftarrow (\mathcal{D} > 0)?1: -1$;
25 $\quad\quad p' \leftarrow p + \rho\Delta$;
26 $\quad\quad$ **while** $p' > p_{up} || p' < p_{low}$ **do**
27 $\quad\quad\quad \rho \leftarrow \rho/2$;
28 $\quad\quad\quad p' \leftarrow p + \rho\Delta$;
29 $\quad\quad$ **end**
30 $\quad\quad U'_{\text{mec}} \leftarrow$ calculate the MEC utility under p' by Alg. 2;
31 $\quad$ **end**
32 **end**

VII. Simulation Results

In the section, we give several groups of simulation experiments to verify the performance of the proposed algorithms and analyze the influence factors that affect the algorithm performance.

A. Parameter Setting

In the simulations, several users are located within the coverage of the MEC server. The number of the users is set in the range of [2, 16]. The MEC server is equipped with multiple computing units, and the number of units is set to 4. Each unit is DVFS-enabled and its maximal adjustable frequency is set in the range of $[2, 6]$ GHz. The users also can adjust each frequency during the range of $[0, 2]$ GHz. The length of each time slice τ is set as 1 s [25]. The size of the task generated by the n-th user is set between 600 and 800 KB, and the computation density is set in the range of $[600, 800]$ cycles/bit. It is assumed that all users are equipped with the same type of battery and the maximum battery capacity is 1337 J (e.g., 3.7V, 1000mAh). The rest energy in each user is randomly generated in $[0, 133.7]$ J. The transmission power of user n is randomly generated in the range of $[150, 250]$ mW.

Besides, the wireless bandwidth B is set in the range of $[20, 150]$ MHz. The noise power σ^2 is set as 10^{-6} W/Hz [6] and the channel power gain h_n follows an exponential distribution with mean of 1. The energy coefficient factor is set as 10^{-26} [26] for the MEC server and 5×10^{-27} for each user. The electricity price is set as 0.5×10^{-6} per joule. The satisfaction factor ω_n is set as 1, and the trade-off factor γ_n is

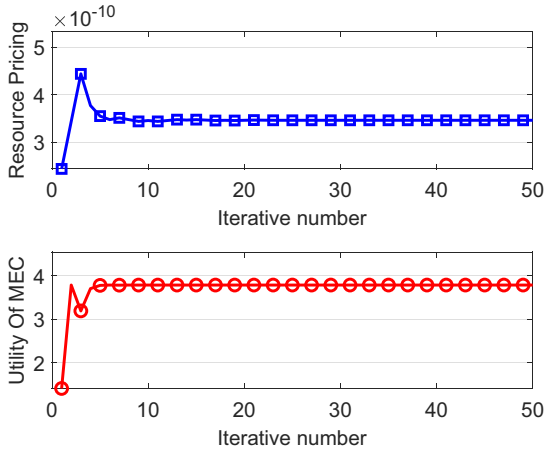


Fig. 5. The resource pricing and utility of the MEC server change versus the number of iterations.

set as 0.5. For each setting, we run the experiments multiple times and average the results to improve the reliability.

B. The Convergence Performance

Conducting convergence experiments aims to verify whether the system, with a single MEC server and multiple users, can achieve an equilibrium point, providing empirical support for the system's gaming strategy. Fig. 5 depicts the variations in the utility and resource pricing of the MEC server. With an increasing number of iterations, both the utility and pricing of the MEC server stabilize after 5 iterations. It can be discerned from this that the user's offloading decisions also manifest a tendency towards stability. Consequently, it can be inferred that a Stackelberg equilibrium exists between the strategies of the users and the MEC server.

C. The Impact of Maximum Adjustable Frequency

In this group of experiments, we present the impact of the MEC maximal adjustable frequency on the performance of both parties. The number of users is set as 6, the bandwidth is set as 100 MHz, and the MEC maximal frequency $F_{\max}$ is varying from 1.6 GHz to 4.8 GHz in step of 0.4.

Firstly, the impact of $F_{\max}$ on the task offloading decision and utility of MDs is shown in Fig. 6. To better show the results, only three MDs are selected randomly. In Fig. 6(a), the changing trends of the task offloading decision, i.e., the local execution task ratio α_i and the remote offloading task ratio β_i, with the increasing $F_{\max}$ value are shown. From the figure, we can observe that the task throughput of MD1, MD2, and MD3 initially increases with the rise of $F_{\max}$ and levels off when the $F_{\max}$ value reaches 3.2 GHz. That is because the MEC server can provide more computation capacity under a greater $F_{\max}$, and can process more offloaded tasks. The main reason for the increase in task throughput is the increase in the amount of offloading tasks. However, when the $F_{\max}$ value increases further, the actual optimal frequency of the MEC server is not increasing correspondingly. That is because the energy consumption is raising rapidly with the increase of CPU frequency. If the frequency of MEC is set very high, the resource pricing should be put up correspondingly to offset energy costs. Under the case, the MDs would reduce the amount of offloaded tasks to safeguard their own interests. That is to say, further increasing the computation capacity does not bring more benefit for the MEC server. Considering the benefits of both sides, even the adjustable frequency gets higher, the optimal frequency of MEC remains unchanged.

In Fig. 6(b), the changing trends of the utility are plotted for the three MDs. It is obvious that the utility value of the three MDs shows the same trend, that is, increasing firstly and keeping unchanged lastly. That is because the utility of MDs is proportional to the amount of processed tasks, and inversely proportional to resource prices. For MD1, MD2, and MD3, the trend in utility change is similar to the task processing volume, both stabilizing after $F_{\max}$ reaches 3.2 GHz. On the other hand, the price of the MEC server gradually decreases with the increase in $F_{\max}$ until stabilizing after reaching 3.2 GHz. For detailed results, please refer to Table I.

Table I gives the experimental results of MEC under different $F_{\max}$ value, including the suboptimal resource pricing, the optimal frequency, and the maximal utility. The results show that when there are fewer computing resources available, MEC would set a higher price to avoid overloading. Along with the increment of the $F_{\max}$ value, MEC could set a lower price to attract more computing requirements and generate more profit. That is also the reason that the utility of MEC raises when $F_{\max}$ is increasing from 1.6 GHz to 3.2 GHz. When $F_{\max}$ increases further, the utility of MEC stops growing, that is because the global optimal combination of resource pricing and frequency setting has reached at 3.2 GHz, and it does not change as the $F_{\max}$ value changes.

D. The Impact of Bandwidth

Tables II and III show the results with different bandwidth. In this group of experiments, the number of users is set as 6, the maximum main frequency of each computing unit of the MEC server is set to 2 GHz, the bandwidth B is set as $\{20, 50, 80, 100, 120, 150\}$ MHz, respectively.

According to Table III, it can be observed that with the increase in bandwidth, the average offloading rate β_i of users gradually rises. This is attributed to the fact that higher bandwidth helps reduce the latency of task transmission, enhancing user experience and making users more willing to opt for task offloading. Furthermore, as indicated in Table II, the pricing of resources from the MEC server exhibits a downward trend with increasing bandwidth. This trend further encourages users to lean towards task offloading. However, the user's offloading rate increases slowly, mainly due to the limited computing resources of the MEC server. It can be observed from Table II that the optimal usage frequency of a single computing unit of the MEC server is almost equal to all frequencies owned by the computing unit. The benefit of the MEC server gradually increase, attributed to the rise in the number of offloaded tasks by users. In Table III, the local task processing rate for users remains relatively constant with the increase in bandwidth. This is because users have sufficient computing resources locally to handle a portion of

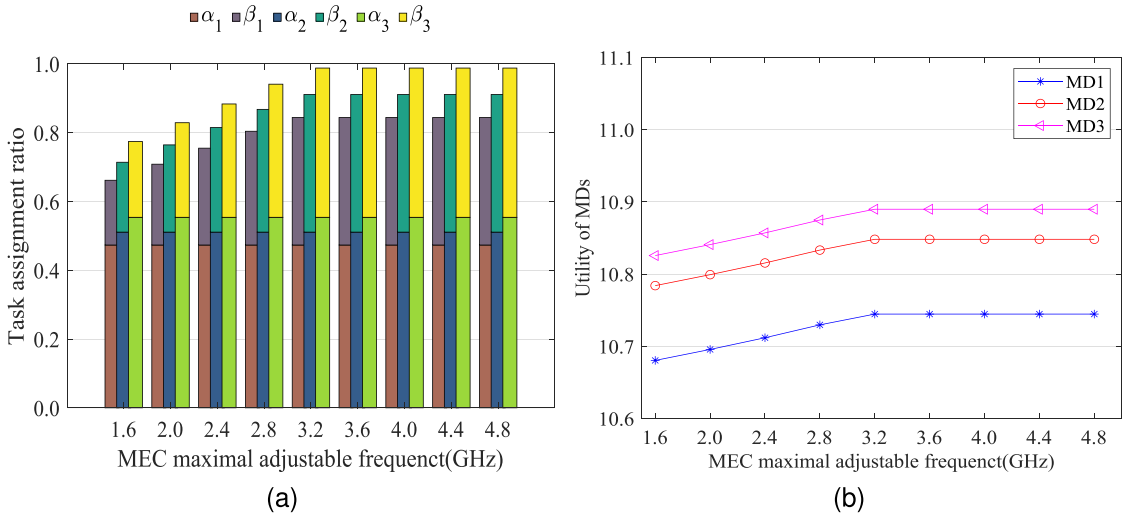


Fig. 6. Task assignment ratio and utility of three MDs vs. MEC maximal adjustable frequency.

TABLE I
DIFFERENT PERFORMANCE INDICATORS OF MEC VS. MAXIMAL ADJUSTABLE FREQUENCY

$F_{\max}$	1.60	2.00	2.40	2.80	3.20	3.60	4.00	4.40	4.80
Suboptimal Price	5.16E-10	4.82E-10	4.52E-10	4.25E-10	4.04E-10	4.04E-10	4.04E-10	4.04E-10	4.04E-10
Optimal Frequency(GHz)	1.60	2.00	2.40	2.80	3.07	3.07	3.07	3.07	3.07
Maximal Utility	3.2023	3.6690	4.0286	4.2699	4.3343	4.3343	4.3343	4.3343	4.3343

TABLE II
DIFFERENT PERFORMANCE INDICATORS OF MEC VS. BANDWIDTH

Bandwidth	0.20	0.50	0.80	1.00	1.20	1.50
Suboptimal Price	4.86E-10	4.83E-10	4.82E-10	4.82E-10	4.82E-10	4.82E-10
Optimal Utility	3.6031	3.6530	3.6658	3.6699	3.6728	3.6756
Optimal Frequency(GHz)	2.00	2.00	2.00	2.00	2.00	2.00

TABLE III
DIFFERENT PERFORMANCE INDICATORS OF MDS VS. BANDWIDTH

Bandwidth	0.20	0.50	0.80	1.00	1.20	1.50
Average α	0.5432	0.5432	0.5432	0.5432	0.5432	0.5432
Average β	0.2628	0.2680	0.2694	0.2698	0.2701	0.2704
Average Utility	10.7712	10.7727	10.7731	10.7733	10.7733	10.7734

the tasks, and the increased bandwidth does not affect the efficiency of local task processing. The utilities for users exhibit a rising trend, propelled by an increase in the number of tasks processed by users and a decrease in the resource pricing of the MEC server.

E. The Impact of User Numbers at Different Frequencies

Fig. 7 plots the results for different numbers of users at different frequencies. In this group of experiments, the bandwidth is set as 100 MHz, the maximum computing frequency of a single computing unit of the MEC server is set as $\{1.6, 2.4, 3.2, 4.0\}$ GHz, the number of users N is set from 2 to 16 in step of 2.

From Fig. 7(a) we can observe the changes in the average local processing task rate α and remote offloading rate β of users as the number of users changes under the four frequencies. Under the same frequency, as the number of users increases, the average remote offloading task rate shows a downward trend. This is because more users lead to a reduction in the bandwidth obtained by each user, thereby reducing the transmission rate and increasing latency. Under different frequency conditions, as the number of users increases, the average offloading task ratio of users at higher frequencies will be higher than the average offloading task ratio at lower frequencies. This is because, with an increasing number of users, the low-frequency MEC server cannot handle more tasks within the required latency, whereas the high-frequency MEC server has sufficient computing resources.

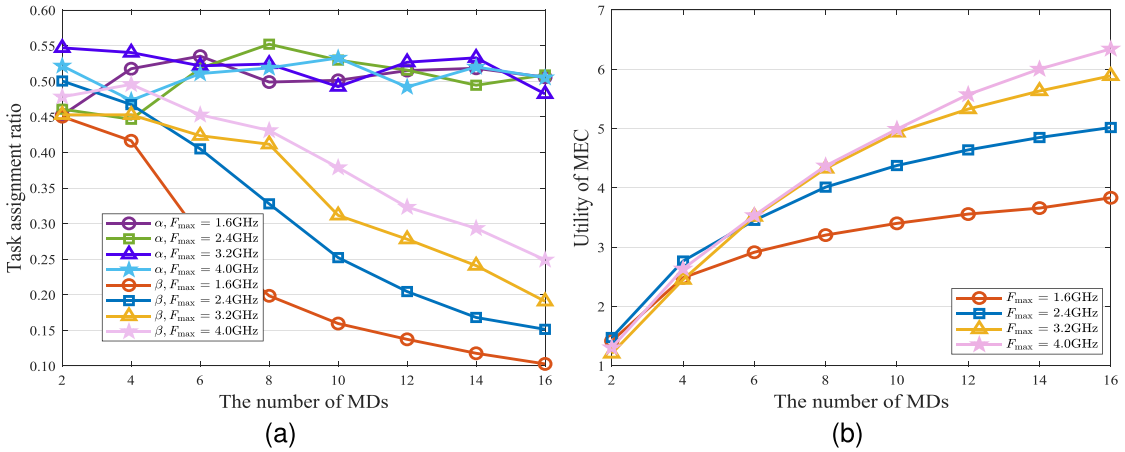


Fig. 7. Task offloading decisions and MEC server utility versus MDs and frequency.

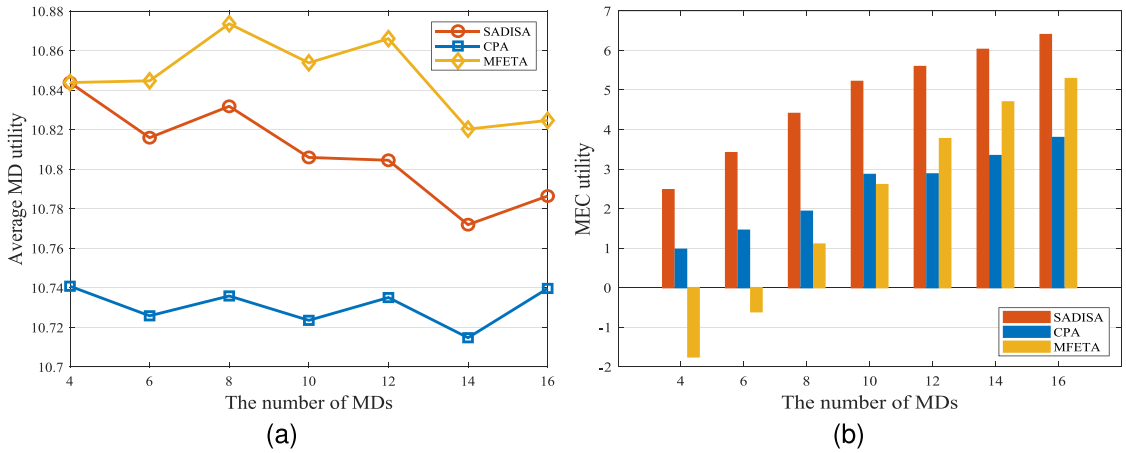


Fig. 8. Performance comparison with different number of MDs.

From Fig. 7(b) we can see that as the number of users increases, the benefits of the MEC server under the four frequencies show an upward trend. This is because the increase in the number of users represents a rise in the total number of tasks, leading to more tasks being offloaded to the MEC server, increasing the efficiency of the MEC server.

F. Performance Comparison

To further evaluate the performance of the SADISA proposed in this paper, we compare SADISA with two baseline algorithms.

- *Constant price algorithm (CPA):* The MEC server sets the resource pricing to a fixed value p^{cpa}. The CPU computation frequency is variable and depends on the users' task offloading decisions, which are influenced by the resource pricing and users' existing task loads.
- *Maximum frequency execution task algorithm (MFETA):* The task execution frequency of each computing unit of the MEC server is set to the maximum value, while the resource pricing is variable and depends on the current resource demand and utility.

In the following numerical calculations, we set the maximum frequency of each computing unit of the MEC server to 6 GHz, while setting the price in the CPA strategy to 6.5E-10. The remaining parameters remain consistent with previous calculations.

Fig. 8 presents the numerical calculation results of the three strategies with respect to the number of users. Fig. 8(a) illustrates the user average benefits generated as the number of users varies, where the benefits of SADISA are lower than MFETA but higher than CPA. This is attributed to MFETA utilizing all computing resources to assist users in computing tasks, while CPA performs the worst in terms of benefits since it does not encourage users to offload more tasks through price adjustments. As the number of users changes, the average user benefits generated by the three strategies fluctuate up and down. This is because each group of different numbers of user tasks is randomly generated with different task sizes. Fig. 8(b) illustrates the impact of user quantity variations on the MEC server benefit for three different strategies. As the number of users increases, the benefit gradually rise, with SADISA yielding the highest MEC server benefit due to its comprehensive consideration of price and frequency adjustments. When the user quantity is less than 12, CPA exhibits higher MEC server benefits than MFETA. This is attributed to the fewer tasks offloaded by users when the user quantity is low, leading to higher energy consumption when the MEC

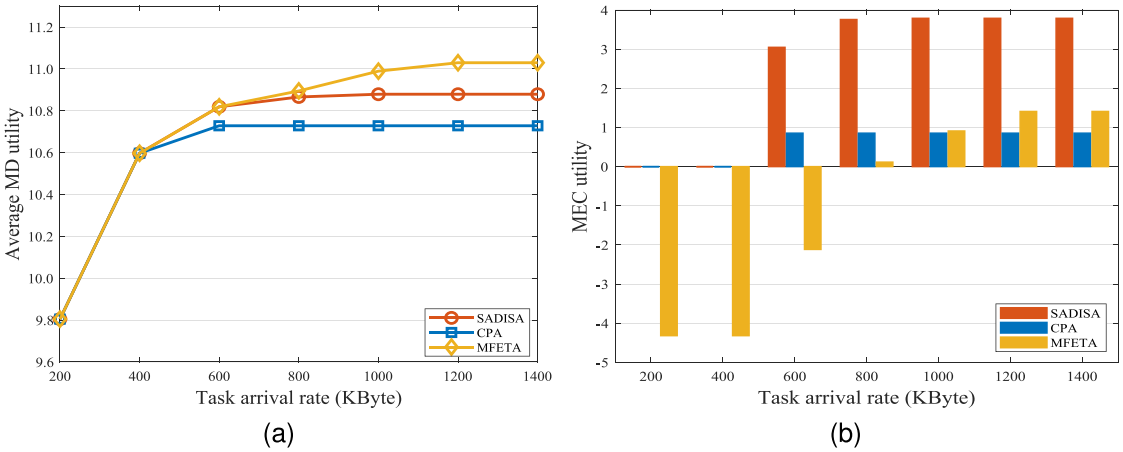


Fig. 9. Performance comparison with different task arrival rate.

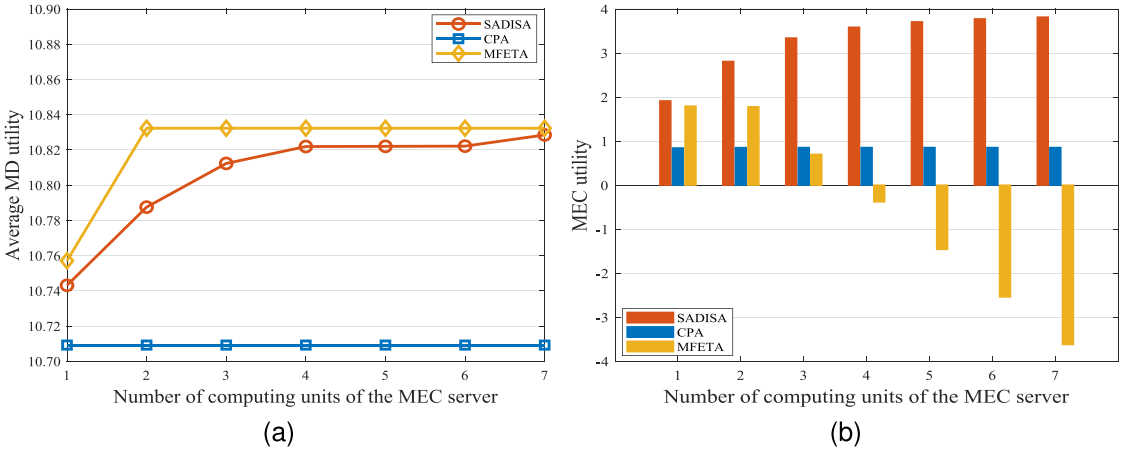


Fig. 10. Performance comparison with different number of computing units.

server utilizes all computing resources for task processing, making the performance of MFETA inferior to CPA. For user quantities of 4 and 6, MFETA generates negative MEC server benefit, resulting from a scarcity of offloaded tasks and high energy consumption. While MFETA can generate more user benefits, SADISA outperforms in generating more MEC server benefit. Overall, SADISA demonstrates optimal system efficiency.

Fig. 9 presents the numerical calculation results of the three strategies with respect to the task arrival rate. Fig. 9(a) illustrates the variations in average user benefits for the three strategies under different task arrival rates. As the task arrival rate increases, the benefit of SADISA gradually decreases, falling below that of MFETA but surpassing CPA. This is attributed to MFETA making full use of all computing resources in MEC, prompting users to offload more tasks and consequently increasing user benefits. Fig. 9(b) depicts the impact of the three strategies on the MEC server benefit. SADISA yields the maximum MEC server benefit, as it dynamically adjusts both pricing and frequency to maximize the efficiency of the MEC server. For MFETA, MEC server benefit are negative when the task arrival rate is less than 800KB, due to the strategy utilizing all computing resources, resulting in excessive energy consumption. CPA fails to generate MEC server benefit at task arrival rates of 200KB and 400KB, as the low number of tasks allows users to rely on their own computing resources to maximize their individual benefits.

Fig. 10 presents the experimental results of the three strategies with respect to the number of computing units. Fig. 10(a) illustrates the impact of different numbers of computing units on user benefits for the three strategies. User benefits show an initial increase followed by stabilization in both SADISA and MFETA. This trend is attributed to the constant total number of user tasks. In the case of CPA, user benefits also exhibit an initial increase followed by stabilization, although the growth is relatively less pronounced on the graph. Fig. 10(b) illustrates the impact of the three strategies on the MEC server benefit under varying numbers of computing units. In both SADISA and CPA, the MEC benefit shows an increasing trend, gradually stabilizing. For SADISA, this is attributed to having more resources, enabling it to handle a greater number of tasks. In the case of CPA, as the number of computing units increases, each unit bears a reduced share of computational resources, leading to lower energy consumption and an enhancement in MEC benefit. However, in MFETA, MEC benefit shows a downward trend, which is due to the increase in the number of computing units leading to

an increase in energy consumption, thereby reducing MEC benefit.

VIII. Conclusion

In this paper, we study the computation offloading problem for a multi-device single MEC system within a time slice. Both the MDs and the MEC can perform frequency adjustment to effectively save energy consumption. In order to make reasonable resource pricing, resource allocation and task offloading decisions, Stackelberg game theory is adopted. We propose the SADISA to find optimal policies for users and MEC server. Simulation experiments show that by adjusting parameters, the performance of SADISA can be changed. In comparative experiments, SADISA can better improve system efficiency compared to the other two methods.

References

[1] X. Wang, J. Ye, and J. C. S. Lui, "Mean field graph based D2D collaboration and offloading pricing in mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 491–505, Feb. 2024.

[2] D. Reinsel, J. Gantz, and J. Rydning, *The Digitization of the World From Edge to Core, Data Age 2025*, IDC, Needham, MA, USA, Nov. 2018.

[3] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 420–423, Jun. 2018.

[4] S. Noreen, N. Saxena, and A. Roy, "Discount interference pricing mechanism for data offloading in D2D communications," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1688–1691, Aug. 2018.

[5] J. Chen et al., "A multi-leader multi-follower Stackelberg game for coalition-based UAV MEC networks," *IEEE Wireless Commun. Lett.*, vol. 10, no. 11, pp. 2350–2354, Nov. 2021.

[6] Y. Chen, N. Zhang, Y. Zhang, X. Chen, and X. S. Shen, "Energy efficient dynamic offloading in mobile edge computing for Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1050–1060, Jul.–Sep. 2021.

[7] Y. Yang, Z. Liu, X. Yang, K. Wang, X. Hong, and X. Ge, "POMT: Paired offloading of multiple tasks in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8658–8669, Oct. 2019.

[8] T. Fang, F. Yuan, L. Ao, and J. Chen, "Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: A potential game approach," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3226–3237, Mar. 2022.

[9] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 273–287, Jul. 2020.

[10] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource trading in blockchain-based Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3602–3609, Jun. 2019.

[11] F. Zeng, Q. Chen, L. Meng, and J. Wu, "Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3247–3257, Jun. 2021.

[12] S. Liu, J. Tian, X. Deng, Y. Zhi, and J. Bian, "Stackelberg game-based task offloading in vehicular edge computing networks," *Int. J. Commun. Syst.*, vol. 34, no. 16, 2021, Art. no. e4947.

[13] F. Li, H. Yao, J. Du, C. Jiang, and Y. Qian, "Stackelberg game-based computation offloading in social and cognitive Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5444–5455, Aug. 2020.

[14] H. Zhou, Z. Wang, N. Cheng, D. Zeng, and P. Fan, "Stackelberg-game-based computation offloading method in cloud–edge computing networks," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16510–16520, Sep. 2022.

[15] H. Zhou, Z. Wang, G. Min, and H. Zhang, "UAV-aided computation offloading in mobile-edge computing networks: A Stackelberg game approach," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 6622–6633, Apr. 2023.

[16] M. Tao, K. Ota, M. Dong, and H. Yuan, "Stackelberg game-based pricing and offloading in mobile edge computing," *IEEE Wireless Commun. Lett.*, vol. 11, no. 5, pp. 883–887, May 2022.

[17] M. Guo, W. Wang, X. Huang, Y. Chen, L. Zhang, and L. Chen, "Lyapunov-based partial computation offloading for multiple mobile devices enabled by harvested energy in MEC," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 9025–9035, Jun. 2022.

[18] F. Zhou, H. Sun, Z. Chu, and R. Q. Hu, "Computation efficiency maximization for wireless-powered mobile edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–6.

[19] Y. Deng, Z. Chen, X. Yao, S. Hassan, and A. M. A. Ibrahim, "Parallel offloading in green and sustainable mobile edge computing for delay-constrained IoT system," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12202–12214, Dec. 2019.

[20] J. Mei, L. Dai, Z. Tong, X. Deng, and K. Li, "Throughput-aware dynamic task offloading under resource constant for MEC with energy harvesting devices," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 3460–3473, Sep. 2023.

[21] S. Xia, Z. Yao, Y. Li, and S. Mao, "Online distributed offloading and computing resource management with energy harvesting for heterogeneous MEC-enabled IoT," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6743–6757, Oct. 2021.

[22] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 4858–4873, Aug. 2021.

[23] P. Cai, F. Yang, J. Wang, X. Wu, Y. Yang, and X. Luo, "JOTE: Joint offloading of tasks and energy in fog-enabled IoT networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3067–3082, Apr. 2020.

[24] Z. Zhang and F. Zeng, "Efficient task allocation for computation offloading in vehicular edge computing," *IEEE Internet Things J.*, vol. 10, no. 6, pp. 5595–5606, Mar. 2023.

[25] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[26] P. Zhao, J. Tao, K. Lui, G. Zhang, and F. Gao, "Deep reinforcement learning-based joint optimization of delay and privacy in multiple-user MEC systems," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1487–1499, Apr.–Jun. 2023.

[27] K. Wang, Z. Ding, D. K. C. So, and G. K. Karagiannidis, "Stackelberg game of energy consumption and latency in MEC systems with NOMA," *IEEE Trans. Commun.*, vol. 69, no. 4, pp. 2191–2206, Apr. 2021.

Jing Mei received the Ph.D. degree in computer science from Hunan University, China, in 2015. She is currently an Assistant Professor with the College of Information Science and Engineering, Hunan Normal University. She has published 16 research articles in international conference and journals, such as IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed System, IEEE Transactions on Service Computing, *Cluster Computing*, *Journal of Grid Computing*, and *Journal of Supercomputing*. Her research interests include parallel and distributed computing, cloud computing, and edge computing.

Cuibin Zeng received the B.S. degree in computer science and technology from Jishou University, Jishou, China, in 2022. He is currently pursuing the M.S. degree with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. His research focuses on resource scheduling and price allocation in mobile-edge computing.

Zhao Tong (Senior Member, IEEE) is currently a Professor with Hunan Normal University. He was a Visiting Scholar with Georgia State University from 2017 to 2018. He has author or co-authored more than 50 papers in peer-reviewed international journals and conferences, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. His research interests include AI computing, parallel and distributed computing systems, and resource management. He is a Senior Member of the CCF.

Zhibang Yang received the Ph.D. degree in computer science from Hunan University, China. He is currently a Professor with Changsha University. His major research contains data management, parallel computing, and network security.

Keqin Li (Fellow, IEEE) is a SUNY Distinguished Professor of Computer Science with the State University of New York. He is also a National Distinguished Professor with Hunan University, China. He has authored or co-authored more than 770 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He holds nearly 60 patents announced or authorized by the Chinese National Intellectual Property Administration. His current research interests include cloud computing, fog computing, and mobile-edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has chaired many international conferences. He is currently an Associate Editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.