Journal of Systems Architecture 160 (2025) 103360

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Collaborative optimization of offloading and pricing strategies in dynamic MEC system via Stackelberg game

Jing Mei [a], Cuibin Zeng [a], Zhao Tong [a,1,*], Longbao Dai [a], Keqin Li [b,2]

[a] College of Information Science and Engineering, Hunan Normal University, Changsha, Hunan, 410081, China
[b] Department of Computer Science, State University of New York, New Paltz, NY, 12561, USA

ARTICLE INFO

Keywords:
Mobile edge computing
Energy harvesting
Lyapunov optimization
Stackelberg game

ABSTRACT

The rapid advancement of 5G technology has indirectly propelled the growth of connected devices within the Internet of Things (IoT). Within the IoT domain, mobile edge computing (MEC) has demonstrated potential in task processing. However, as computational services expand, the reliable determination of user offloading strategies and the rational establishment of service prices offered by servers to users continue to present challenging research directions. The primary focus of this paper revolves around task offloading in the MEC system, encompassing numerous user terminal devices that support energy harvesting (EH), a MEC server and a central cloud server. The optimization goals are to maximize the utilities for both users and the MEC server by adjusting offloading and pricing strategies. To guarantee the task queue's stability within the system and achieve a reasonable allocation of system resources, we propose a dynamic task offloading approach rooted in Lyapunov optimization theory and Stackelberg game theory. In this algorithm, the MEC server takes on the role of the leader, while each user terminal device acts as the follower. Aiming at the game equilibrium existence of the algorithm, a series of mathematical analysis is carried out. Additionally, we conduct extensive simulation experiments to validate the proposed algorithm's effectiveness. The proposed algorithm achieves improvements in user utility, with a 6.43% increase compared to the average time-constrained task offloading (ATCTO) scheme, a 61.80% improvement over the local-only processing (LOP) scheme, and a 23.97% enhancement over the genetic algorithm (GA) scheme. Meanwhile, it achieves a task queue backlog reduction of 50.00% compared to ATCTO, 70.00% compared to LOP and 15.28% compared to GA.

1. Introduction

With the proliferation of 5G technology, its high speed and low latency enable real-time connectivity, and the IoT stands as one of the beneficiaries in the era of 5G. According to [1], it is forecasted that the number of IoT devices will exceed 75 billions by 2025. This explosive growth will significantly amplify the scale of mobile data traffic. Effectively managing the surge in mobile data traffic can be achieved through mobile data offloading [2]. However, relying solely on cloud computing is difficult to improve user experience quality. It is the increasing user demand for quality of experience that drives the development of MEC. MEC moves computing power and resources closer to users, usually on edge devices of the network. This aids in reducing data transmission latency and supporting applications with strong real-time requirements. Moreover, edge data processing reduces the need to transmit sensitive data to remote servers, thereby enhancing data privacy and security.

Energy consumption for local computing and wireless transmission tasks on user terminal devices (e.g., wearable devices, tablets, and smartphones) is sourced from their internal batteries. For convenience, these user terminal devices will be referred to as "users". Due to limited size of users, long lifetime battery might not be appropriate, while smaller-capacity battery may require frequent replacements, causing significant inconvenience. Fortunately, in recent years, EH technologies have garnered significant attention. These technologies enable users to harvest energy from the nature (e.g., solar), and store it in batteries. By integrating rechargeable batteries with energy harvesting technologies, the frequency of battery replacements can be notably reduced. In the future, the IoT could potentially integrate various energy harvesting methods [3]. Therefore, the integration of energy harvesting technologies into MEC holds practical significance.

* Corresponding author.
E-mail addresses: jingmei1988@163.com (J. Mei), zeng1941183190@gmail.com (C. Zeng), tongzhao@hunnu.edu.cn (Z. Tong), awaken6758@gmail.com (L. Dai), lik@newpaltz.edu (K. Li).
[1] Member, IEEE.
[2] Fellow, IEEE.

https://doi.org/10.1016/j.sysarc.2025.103360
Received 3 September 2024; Received in revised form 7 January 2025; Accepted 31 January 2025
Available online 12 February 2025

In order to flexibly respond to the changing demands between user devices and MEC server, this study considers a dynamic MEC system. In this system, factors such as the arrival of tasks and energy collection will change with the change of time. The system can dynamically adjust the task offloading strategy and the service pricing strategy of the MEC server according to real-time conditions, thereby improving user utility and MEC server utility. The authors considered the dynamic MEC system in [4], but did not set the services provided by the MEC server as paid services. In contrast, [5] explored the pricing issue in a dynamic system, but the study implemented a unified service pricing for all user devices and failed to consider the differences between user devices. Given the heterogeneity of user devices, we introduced a differentiated pricing strategy.

In this paper, we explore a dynamic MEC system enabled with EH, which is composed of multiple user terminal devices, a MEC server, and a central cloud. Each user possesses the capability to harness energy from the surrounding natural environment and store it in their built-in rechargeable batteries. Users can offload tasks to the MEC server when needed, and the MEC server can further offload tasks to the central cloud as required. In this study, the MEC server's computational capacity is limited, necessitating revenue generation through user charges. Generally, the energy consumed by users for wireless transmission tasks is lower than that required for performing equivalent-sized local computing tasks. In order to decrease energy consumption, users are more likely to opt for the method of remote task offloading. Nevertheless, due to the fact that the MEC server does not offer services free of charge, each user needs to strike a balance between task offloading and service pricing. The main purpose of our research is to determine the optimal distribution of tasks between local computing and remote offloading for users, while also determining suitable service pricing for individual users for the MEC server.

Given that the optimization objective of this paper involves a dynamic long-term MEC scenario, i.e., the parameters and conditions (e.g., energy harvesting, task arrival) may change over time, making the solution of the problem extremely challenging. To simplify the problem, the authors in [5–7] focused their target optimization on a short time slot, defined as a brief period during which system conditions are assumed to be relatively stable. However, focusing only on a single time slot may cause system instability problems, since it is difficult to cope with rapid changes. To overcome this difficulty, we employ Lyapunov optimization theory to transform the long-term optimization problem into a sequence of short-term optimization problems, enabling the achievement of long-term optimization objectives through the resolution of multiple short-term optimization tasks. To enhance the efficiency and rationality of resource allocation, we introduce Stackelberg game theory. In this framework, the MEC server acts as a leader, formulating differentiated pricing strategies to guide the offloading decisions of multiple user devices. By combining Lyapunov optimization theory with Stackelberg game theory, we can effectively address the complexities in dynamic environments and achieve long-term optimization objectives.

This research work makes the following main contributions:

- We explore task offloading and pricing in a multi-user environment with a single MEC server, incorporating energy harvesting (EH) for users to utilize renewable energy for task computation.
- We utilize Lyapunov optimization theory to adapt to external changes by breaking down long-term goals into short-term objectives, stabilizing the task queue while optimizing performance.
- We propose a Stackelberg game model for task offloading and pricing, where users consider energy use, queue length, and pricing, while the MEC server applies differentiated pricing based on offloaded tasks.
- We account for the practical limitation of the MEC server's computing capacity within each time slot, determined by its CPU frequency. If the server cannot process all accumulated tasks from users, it requests processing from the central cloud server at a cost.

Next, we will introduce the remaining structure of this paper. Section 2 delves into the related work. Section 3 defines the system model and presents the optimization problems. Section 4 describes the analysis and solution of the objective optimization problems. Subsequently, we introduce a multi-device task offloading and pricing mechanism algorithm (MDTOPMA) in Section 5. Section 6 evaluates the algorithm's performance through experiments. The final section concludes by summarizing the contributions of this paper.

2. Related work

Numerous investigations have delved into the domain of computation offloading within the context of MEC. Ning et al. [8] presented an energy-efficient scheduling framework for vehicle networking with support for Multi-Access Edge Computing, aimed at minimizing the energy consumption of roadside units (RSUs) while meeting task latency requirements. The authors proposed a heuristic algorithm to address this issue. Mao et al. [9] focused on the joint optimization of execution delay and device energy consumption, proposing a low-complexity suboptimal algorithm based on an alternating minimization strategy. The algorithm minimized the weighted sum of execution delay and device energy consumption by adjusting task offloading scheduling and transmission power allocation. Zhao et al. [10] proposed an energy-efficient offloading algorithm to save mobile device energy while meeting application response time requirements. Chen et al. [11] investigated energy-efficient task offloading in MEC and proposed a dynamic offloading algorithm that guarantees the average queue length. Li et al. [12] proposed a computational offloading mechanism based on a two-stage Stackelberg game and used two dynamic iterative algorithms to solve the utility optimization problem in the game. Although the above studies have their own merits, they do not consider the computing power of the device itself.

To overcome this deficiency, in recent years, many studies have begun to consider introducing local computing resources. Based on the size of the offloaded data, Hu et al. [13] determined the tasks that necessitate local handling. They proposed a MEC system energy consumption optimization problem and solved it in two stages. Wang et al. [14] divided the computing task into two parts, one of which would be used for local computing. Their research intended to minimize the AP's overall energy consumption, however they did not consider the energy consumption resulting from the MEC server's computational tasks. They omitted for the energy consumption consumed by the MEC server during task processing.

To more comprehensively address issues such as task offloading and energy management, and to achieve optimal performance with limited resources, game theory is an appropriate approach. There is currently a lot of work taking game theory into MEC. Li et al. [6] described the interaction between mobile device (MD) and edge cloud server (ECS) in the process of computing load as a Stackelberg game and confirmed the equilibrium of this game. The authors in [5] proposed an optimal resource purchase strategy with a set price, and proposed the optimal pricing for edge cloud computing resources utilizing the Stackelberg game model. Liu et al. [7] proposed the problem of transmission power offloading optimization and edge cloud pricing in mobile edge computing systems, and adopted the offloading strategy and price control (OSPC) algorithm based on Stackelberg game to solve it. Bishoyi et al. [15] presented a distributed algorithm utilizing the alternating direction multiplier method (ADMM) to solve the Stackelberg game they proposed. Although Bishoyi et al. considered the Stackelberg game, they only considered the optimization of the problem within one time slot. Zeng et al. [16] introduced a reward system to incentivize

Table 1
Difference between our scheme and the main related schemes.

Scheme	Local computation	MEC energy consumption	Multiple time slots	Differentiated pricing
[5] S.-H.Kim et al.	✓	×	×	×
[6] M.Li et al.	✓	×	×	✓
[7] X.Liu et al.	✓	×	×	✓
[8] Z.Ning et al.	×	✓	×	×
[9] Y.Mao et al.	×	×	×	×
[10] X.Zhao et al.	×	✓	×	×
[11] Y.Chen et al.	×	×	✓	×
[12] F.Li et al.	×	✓	×	✓
[13] X.Hu et al.	✓	×	×	×
[14] F.Wang et al.	✓	×	✓	×
[15] P.K.Bishoyi et al.	✓	✓	×	✓
[16] F.Zeng et al.	✓	×	×	✓
Our Scheme	✓	✓	✓	✓

volunteer vehicles participating in Vehicular Edge Computing (VEC) offloading and devised an enhanced genetic algorithm to explore the optimal strategy for the VEC server. However, they also did not account for changes in the number of tasks over time.

Among most of the studies mentioned before, some studies failed to fully leverage local computing resources, some did not account for the energy consumption associated with the MEC server's computations, and some only focused on the optimization of single slot goals (see Table 1). To cope with these situations, each user in this paper supports local computing through EH technology, while the MEC server considers computing energy consumption as its own overhead. In order to achieve a reasonable allocation of resources among users and the MEC server and tackle the long-term optimization problem, we combine the Stackelberg game with Lyapunov optimization.

3. System model and problem formulation

In this section, we will provide an overview of the MEC system architecture and various computation models. Based on these computational models, the optimization problems for both users and the MEC server will be deduced.

3.1. Mobile edge computing system architecture

We investigate a system architecture consisting of three layers. The first layer, called the user terminal device layer, consists of n user devices (e.g., wearables, tablets, smartphones), indexed by $N = \{1, 2, \dots, n\}$. Each user is equipped with an energy harvesting device that enables them to collect energy from the surrounding environment to power their own operations. The second layer is known as the MEC service layer, consisting of a single MEC server co-located with a base station. Lastly, the third layer is denoted as the central cloud service layer, consisting of a cloud server. The three-layer system architecture is illustrated in Fig. 1.

In this architecture, users can transmit their computational tasks to the base station via a wireless network using the time division multiple access (TDMA) protocol. The base station forwards the tasks to the MEC server for computation. Once tasks are completed, the MEC server sends the results back to the base station, which then delivers them back to the users. If the cumulative tasks offloaded by users exceed the MEC server's computational capacity, the excess tasks are offloaded to the central cloud via a wired network, and the results are subsequently returned by the central cloud. Excess tasks can be generated by adjusting the relevant parameters, such as the task arrival rate, and available computational resources. By adjusting these parameters, the total offloading demand can be controlled, simulating the scenario where the

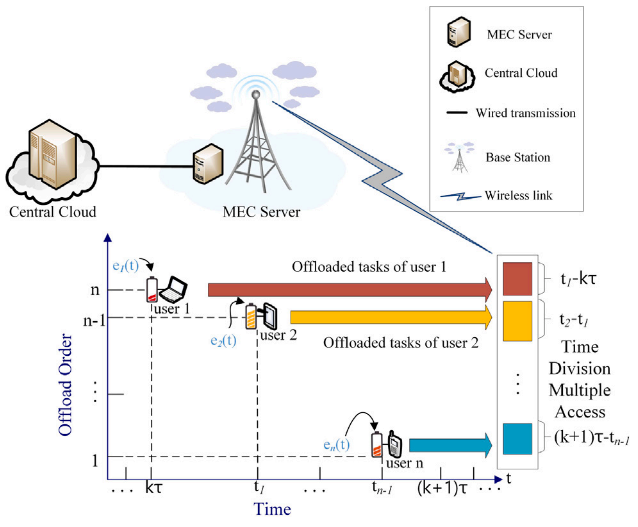


Fig. 1. A mobile edge computing system architecture.

MEC server's computational capacity is exceeded. We consider the central cloud to possess formidable computational capabilities, enabling it to handle a significant amount of tasks independently.

Various uncertainties and interferences exist in practical application environments. To enhance system stability, we consider a time-slot-based system and partition time into equidistant time slots. The system is indexed by $t \in \mathcal{T} = \{0, 1, 2, \dots, T\}$ with slot length τ.

3.2. Computing task and task queue model

For each user i, at the start of the tth time slot, the user's application requests a set of tasks for computation. The size of the tasks is the task arrival rate $a_i(t)$. The tasks received during the current time slot can only be handled in future time slots. In order to achieve more flexible task offloading, we assume that the computational tasks of the users follow a data partitioning model [4,17], where the data bits of the computational tasks are independent and can be arbitrarily partitioned into multiple independent subtasks. Each user's tasks are stored in the task queue $\boldsymbol{Q} = \{Q_1, Q_2, \dots, Q_n\}$.

Let $Q_i(t)$ represent the tasks remaining incomplete for the ith user in the preceding t time slots. The amount of tasks in the queue can be adjusted using the following formula.

$$Q_i(t+1) = \max\{0, Q_i(t) - q_i(t)\} + a_i(t), \tag{1}$$

where $q_i(t)$ denotes the total tasks processed by user i during the tth time slot. $q_i(t)$ can be modeled as $q_i(t) = q_{i0}(t) + q_{i1}(t)$, where $q_{i0}(t)$ denotes the quantity of tasks processed locally by user i, and $q_{i1}(t)$ signifies the quantity of tasks offloaded remotely by user i. $q_i(t)$ satisfies

$$0 \le q_i(t) \le Q_i(t), t \in \mathcal{T}. \tag{2}$$

According to the definition of queue stability in [18], the constraint for task queue stability is given as

$$\lim_{t\to+\infty} \frac{\mathbb{E}\{Q_i(t)\}}{t} = 0, t \in \mathcal{T}. \tag{3}$$

3.3. Local execution and communication model

3.3.1. Local computing energy consumption model

Local task execution involves users using their available processing resources to handle tasks, leading to local energy consumption. This local energy consumption is dynamic, as it varies based on offloading strategy. Same as [19], we model it as

$$H_{i0}^u(t) = k_i^u [f_i^u]^2 q_{i0}(t) d_i, \tag{4}$$

where k_i^u is the capacitance switching coefficient, which is dependent on the chip architecture [19], and the superscript u is employed to denote user-specific parameters, distinguishing them from the parameters denoted by the same symbol for the MEC server; f_i^u represents the CPU computing frequency of user i, while d_i represents the computing density of user i.

Given the computation frequency f_i^u and the time slot length τ, the local computing tasks are constrained as

$$0 \le q_{i0}(t) \le \frac{f_i^u \tau}{d_i}, t \in \mathcal{T}. \tag{5}$$

3.3.2. Transmission energy consumption model

Let p_i represent the transmission power of user i. When users offload tasks to the MEC server, they incur transmission energy consumption [19], which is modeled as

$$H_{i1}^u(t) = T_i^{tr}(t) p_i, \tag{6}$$

where $T_i^{tr}(t)$ represents the duration of task transmission for user i.

The task transmission time does not exceed one time slot and satisfies

$$0 \le T_i^{tr}(t) \le \tau, t \in \mathcal{T}. \tag{7}$$

Based on Eqs. (4) and (6), the overall energy consumption $H_i^u(t)$ is derived as $H_i^u(t) = H_{i0}^u(t) + H_{i1}^u(t)$.

3.3.3. Energy harvesting model

We consider the user's energy harvesting to follow the HUS strategy [20], where the energy collected during the current time slot is only available for use in subsequent time slots. Let $e_i(t)$ represent the energy harvested by user i, and $B_i(t)$ represent the remaining energy in the battery. The battery energy update is modeled as:

$$B_i(t+1) = \min\{\max\{B_i(t) - H_i^u(t), 0\} + e_i(t), B_i^{\max}\}, \tag{8}$$

where $B_i^{\max}$ represents the maximum capacity of the battery.

The overall energy consumption generated by user i cannot surpass the remaining battery energy, i.e.,

$$0 \le H_i^u(t) \le B_i(t), t \in \mathcal{T}. \tag{9}$$

3.3.4. Transmission rate model

In the system architecture, data exchange between the users and the MEC server occurs through wireless networks. During this data exchange process, we consider Shannon's formula as the calculation formula for the channel's data transfer rate. Similar to [11], the transmission rate can be modeled as

$$r_i^{tr} = w \log_2 \left(1 + \frac{p_i h_i}{w n_0}\right), \tag{10}$$

where w signifies the base station's bandwidth; n_0 stands for the noise power density; and h_i indicates the channel gain.

According to the transmission time and transmission rate, the offloading task size $q_{i1}(t) = T_i^{tr}(t) r_i^{tr}$ can be deduced.

3.4. Mobile edge server processing model

When users offload tasks to the MEC server, the server receives and processes these tasks, thereby resulting in energy consumption. The energy consumption of the MEC server can be modeled as

$$H^e(t) = k^e (f^e)^2 \sum_{i=1}^{n} q_{i1}^c(t), \tag{11}$$

where k^e is the capacitance switching coefficient of the MEC server; f^e represents the CPU computing frequency of the MEC server; $q_{i1}^c(t)$ denotes the cycle count of the offloaded task for user i and can be described as $q_{i1}^c(t) = q_{i1}(t) d_i$.

Considering the limited resources of the MEC server, it may not be able to fully process all the tasks offloaded by users. Consequently, the MEC server is required to upload the excess tasks to the central cloud for synchronous processing. Let q^e represent the processing capacity of the MEC server within a time slot, i.e., the maximum number of task cycles it can handle. When the amount of task cycles offloaded by users exceeds the MEC server's computational capacity, the excess portion is transferred to the central cloud. Consequently, the MEC server's energy consumption model is expressed as

$$H^e(t) = k^e (f^e)^2 \min\{q^e, q^c(t)\}, \tag{12}$$

where $q^e = f^e \tau$ and $q^c(t) = \sum_{i=1}^n q_{i1}^c(t)$.

3.5. The utility optimization problem

3.5.1. MEC server utility optimization problem

The MEC server generates costs while processing tasks. It is assumed that the MEC server obtains revenue by pricing data per cycle. We adopt a differential pricing approach. Let $R_i^e(t)$ represent the fee that user i needs to pay to the MEC server for offloading data per cycle and define $\boldsymbol{R}(t) = \{R_1^e(t), R_2^e(t), \dots, R_n^e(t)\}$. Let $\pi^e(t)$ represent the revenue obtained from processing all user tasks by the MEC server, which is denoted as $\pi^e(t) = \sum_{i=1}^n R_i^e(t) q_{i1}^c(t)$. Let c^e represent the cost incurred by the MEC server for each unit of energy consumption. When the amount of tasks offloaded by all users exceeds the processing capacity of the MEC server, the MEC server needs to pay a fee to the central cloud for handling. Let $q^r(t) = \max\{0, q^c(t) - q^e\}$ represent the tasks redirected to the central cloud and R^c represent the cost incurred by the MEC server for offloading data to the central cloud per cycle. The optimization problem $\boldsymbol{E1}$ of the MEC server can be modeled as

$$\boldsymbol{E1}: \max_{\boldsymbol{R}(t)} s = \lim_{T\to+\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{\pi^e(t) - \epsilon^e(t)\} \tag{13}$$

$$\text{s.t.} 0 \le R_i^e(t) \le R_i^{e\_\max}(t), i \in N, t \in \mathcal{T}, \tag{14}$$

where $\epsilon^e(t) = c^e H^e(t) + R^c q^r(t)$; $R_i^{e\_\max}(t)$ represents the maximum price per cycle of data charged by the MEC server to user i.

3.5.2. User utility optimization problem

A higher total task quantity processed by users in a time slot implies a reduced task queue, thereby increasing user satisfaction. We adopt the logarithmic utility function [21], modeling it as

$$\omega_i(t) = \chi \sum_{j=0}^{1} \log_2(1 + q_{ij}(t)), \tag{15}$$

where χ is a weight parameter.

In general, for a same task, the energy consumption resulting from local execution exceeds that of transmitting the same quantity of offloaded tasks [22]. Therefore, we consider that offloading an appropriate amount of tasks remotely is advantageous for reducing energy consumption. Consequently, we incorporate the energy saved by offloading tasks compared to local processing into the user utility model. The saved energy is modeled as

$$\psi_i(t) = k_i^u [f_i^u]^2 q_{i1}(t) d_i - \frac{q_{i1}(t) p_i}{r_i^{tr}}. \tag{16}$$

Considering that the MEC server does not provide services to users for free, user i is required to pay a certain cost to the MEC server, determined by the offloaded task cycle count. The offloading cost is modeled as

$$c_i(t) = R_i^e(t) q_{i1}^c(t). \tag{17}$$

The user's utility is related to the total quantity of tasks handled by the user and the energy savings achieved. The user's costs are related to the total energy consumption and the offloading costs. Therefore, the user's utility optimization problem ***P*1** can be described as

$$\boldsymbol{P1}: \max_{T_i^{tr}(t), q_{i0}(t)} \quad u_i = \lim_{T \to +\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{u_i(t)\} \tag{18}$$
$$\text{s.t.}(2), (3), (5), (7) \text{ and } (9),$$

where $u_i(t) = \pi_i^u(t) - \epsilon_i^u(t)$, $\pi_i^u(t) = \omega_i(t) + \psi_i(t)$, $\epsilon_i^u(t) = H_i^u(t) + \lambda c_i(t)$, and λ is a weight parameter.

4. Problem analysis and solution

In this section, the game relationship and related processes between the users and the MEC server will be introduced in detail. To transform long-term optimization problem into multiple short-term optimization problem and ensure the task queue's stability, the Lyapunov optimization theory will be adopted. The optimal strategies of both users and MEC will be taken into consideration.

4.1. The game relationship between users and MEC server

Without adequate incentive measures, the MEC server may be less willing to participate in computation offloading [23]. To incentivize the MEC server, we employ a strategy rooted in Stackelberg game theory to enable multiple users and the MEC server to both achieve their respective benefits. In this game process, the MEC server plays the role of the leader, while users act as followers. In the tth time slot, firstly, the MEC server will provide each user with an initial service quotation $R_i^e(t)$. Secondly, each user is required to make task offloading strategy (i.e., $q_{i0}(t)$ and $T_i^{tr}(t)$) based on the price. Subsequently, the MEC server will update the corresponding prices based on the user's remote offloading task $q_{i1}(t)$ and its own computational capacity. Following this, users will update their task offloading decisions. As this process continues, through multiple iterative steps, until a balance is reached between users' task offloading decisions and the MEC server prices, the iteration for the current time slot concludes. At this point, a Stackelberg equilibrium is achieved between users and the MEC server.

4.2. Problem transformation based on Lyapunov optimization

Since the initial optimization problem involves the situation in the long-term range, and the parameters such as energy acquisition and task arrival will change over time, this makes it complicated to solve the initial problem directly. However, Lyapunov optimization can transform this long-term problem into multiple tractable short-term problems, so that only these short-term problems need to be dealt with, and the original difficult problem can be solved more efficiently. Moreover, through Lyapunov optimization, the user's task queue can remain stable (i.e., satisfying Eq. (3)). Therefore Lyapunov optimization theory is employed.

The Lyapunov function for user i can be defined as

$$L_i(t) \triangleq \frac{1}{2}[Q_i(t)]^2. \tag{19}$$

Based on [24], the Lyapunov drift for user i can be defined as

$$\Delta_i(t) \triangleq \mathbb{E}\left\{ L_i(t+1) - L_i(t) | Q_i(t) \right\}. \tag{20}$$

To balance queue stability and user utility optimization, we introduce the concept of drift-plus-penalty function, which makes a trade-off between task queue stability and user utility. By incorporating an additional penalty term into the Lyapunov function, we can optimize the user utility while satisfying the queue stability condition.

To ensure queue stability, it is necessary to minimize Lyapunov drift. However, the objective is to maximize user utility, we transform the maximization of the user utility function into the minimization of the user loss function (i.e., the negative of the user utility function). The drift-plus-penalty function for user i is represented as $\Delta_i(t) - V\mathbb{E}\left\{u_i(t)|Q_i(t)\right\}$, where V is a non-negative weight parameter.

When considering Lyapunov drift-plus-penalty function, the lack of information about the following time slot (i.e., $L_i(t+1)$) makes direct solving challenging. To remove the reliance on future information, we use scaling to get an upper bound on this function.

Theorem 1. *When a control parameter $V > 0$ is chosen, and considering that both $q_i(t) \in \{0, 1, \dots, Q_i(t)\}$ and $a_i(t) \in \{0, 1, \dots, a^{\max}\}$, we obtain*

$$\begin{aligned} &\Delta_i(t) - V\mathbb{E}\left\{u_i(t)|Q_i(t)\right\} \\ &\quad \le z - \mathbb{E}\left\{q_i(t)[Q_i(t) + a_i(t)]|Q_i(t)\right\} \\ &\qquad - V\mathbb{E}\left\{u_i(t)|Q_i(t)\right\}, \end{aligned} \tag{21}$$

where $z = \frac{1}{2}\{[Q_i(t)]^2 + (a^{\max})^2\} + Q_i(t)a_i(t)$.

Proof. Taking the square of both sides of Eq. (1) and we find that $(max[x, 0])^2 < x^2$ for any $x \in \mathbb{R}$. Therefore, the inequality can be calculated as

$$\begin{aligned} [Q_i(t+1)]^2 &= \left\{\max\{0, Q_i(t) - q_i(t)\} + a_i(t)\right\}^2 \\ &\le [Q_i(t) - q_i(t)]^2 + [a_i(t)]^2 \\ &+ 2a_i(t)[Q_i(t) - q_i(t)] \\ &= [Q_i(t)]^2 + [q_i(t)]^2 + [a_i(t)]^2 \\ &- 2Q_i(t)q_i(t) + 2a_i(t)[Q_i(t) - q_i(t)] \\ &\le 2[Q_i(t)]^2 + (a^{\max})^2 - 2Q_i(t)q_i(t) \\ &+ 2a_i(t)[Q_i(t) - q_i(t)]. \end{aligned} \tag{22}$$

Based on Definition (19) and the aforementioned inequality, the Lagrangian function for the time slot $t+1$ is computed as

$$\begin{aligned} L_i(t+1) &= \frac{1}{2}[Q_i(t+1)]^2 \\ &\le [Q_i(t)]^2 + \frac{(a^{\max})^2}{2} - Q_i(t)q_i(t) \\ &+ a_i(t)[Q_i(t) - q_i(t)]. \end{aligned} \tag{23}$$

Based on Definitions (19) and (20), as well as Eq. (23), the inequality can be derived as

$$\Delta_i(t) \le \frac{1}{2}\{[Q_i(t)]^2 + (a^{\max})^2\} + Q_i(t)a_i^{max}(t) - \mathbb{E}\left\{q_i(t)[Q_i(t)+a_i(t)]|Q_i(t)\right\}. \quad (24)$$

By adding the expression $-V\mathbb{E}\left\{u_i(t)|Q_i(t)\right\}$ to both sides of Eq. (24), we can deduce Eq. (21).

Taking into account the impact of task queue length on user experience, we incorporate the task queue length into the consideration of user utility through Lyapunov optimization. According to Theorem 1, the original problem $\boldsymbol{P1}$ is transformed into $\boldsymbol{P2}$, described as

$$\boldsymbol{P2}: \min_{T_i^{tr}(t), q_{i0}(t)} \mathcal{U}_i(t) = -\left\{q_i(t)[Q_i(t)+a_i(t)]\right\} - Vu_i(t) \quad (25)$$

$$\text{s.t.}(2), (5), (7) \text{ and } (9).$$

4.3. Optimal strategy for users

Each user in the system can collect a certain amount of energy during each time slot and store it in a battery. Users can use this battery energy for local task processing or task offloading to the MEC server. Each user must address two essential inquiries: (i) How many tasks should be processed locally in each time slot; (ii) How much task transfer time is required in each time slot.

4.3.1. Optimal local task strategy

Referring to (25), we calculate the first-order partial derivative of $\mathcal{U}_i(t)$ with respect to $q_{i0}(t)$ as

$$\frac{\partial \mathcal{U}_i(t)}{\partial q_{i0}(t)} = -[Q_i(t)+a_i(t)] - V\left[\frac{\chi}{(1+q_{i0}(t))\ln 2} - k_i^u[f_i^u]^2 d_i\right], \quad (26)$$

and the second-order partial derivative of $\mathcal{U}_i(t)$ with respect to $q_{i0}(t)$ is computed as

$$\frac{\partial^2 \mathcal{U}_i(t)}{\partial q_{i0}(t)^2} = \frac{V\chi}{[1+q_{i0}(t)]^2 \ln 2}. \quad (27)$$

Since Eq. (27) is non-negative, $\mathcal{U}_i(t)$ exhibits convexity concerning $q_{i0}(t)$.

Following Eq. (25), we calculate the first-order partial derivative of $\mathcal{U}_i(t)$ with respect to $T_i^{tr}(t)$ as

$$\frac{\partial \mathcal{U}_i(t)}{\partial T_i^{tr}(t)} = -[Q_i(t)+a_i(t)]r_i^{tr} - V\left[\frac{\chi r_i^{tr}(t)}{(1+q_{i1}(t))\ln 2} + k_i^u[f_i^u]^2 r_i^{tr}(t)d_i + C\right], \quad (28)$$

where $C = -2p_i - \lambda_i R_i^e(t) r_i^{tr} d_i$; and the second-order partial derivative of $\mathcal{U}_i(t)$ with respect to $T_i^{tr}(t)$ is calculated as

$$\frac{\partial^2 \mathcal{U}_i(t)}{\partial T_i^{tr}(t)^2} = \frac{V\chi(r_i^{tr}(t))^2}{(1+q_{i1}(t))^2 \ln 2}. \quad (29)$$

Since Eq. (29) is non-negative, $\mathcal{U}_i(t)$ is also convex with respect to $T_i^{tr}(t)$. In conclusion, $\mathcal{U}_i(t)$ is a convex function with respect to $q_{i0}(t)$ and is also convex with respect to $T_i^{tr}(t)$, and the constraints of problem $\boldsymbol{P2}$ are affine. Therefore, $\boldsymbol{P2}$ can be solved using the method of Lagrange multipliers. Let $\boldsymbol{\mu} = \{\mu_1, \mu_2, \dots, \mu_6\}$ denote the Lagrange multipliers. Similar to [25], the Lagrangian function of Eq. (25) is expressed as

$$\mathcal{L}_i(q_{i0}(t), T_i^{tr}(t), \boldsymbol{\mu}) = \mathcal{U}_i(t) + \mu_1[q_i(t) - Q_i(t)] + \mu_2[H_i^u(t) - B_i(t)] + \mu_3[T_i^{tr}(t) - \tau] - \mu_4 T_i^{tr} + \mu_5[q_{i0}(t) - \frac{f_i^u \tau}{d_i}] - \mu_6 q_{i0}(t). \quad (30)$$

The first-order partial derivative of $\mathcal{L}_i$ with respect to $q_{i0}(t)$ is given by the following expression

$$\frac{\partial \mathcal{L}_i}{\partial q_{i0}(t)} = \frac{\partial \mathcal{U}_i(t)}{\partial q_{i0}(t)} + \mu_1 + \mu_2 k_i^u (f_i^u)^2 d_i + \mu_5 - \mu_6. \quad (31)$$

Solving for $\partial\mathcal{L}_i/\partial q_{i0}(t) = 0$, the optimal local task workload $q_{i0}^*(t)$ can be calculated as

$$q_{i0}^*(t) = \frac{V\chi}{g\ln 2} - 1, \quad (32)$$

where $g = [\mu_1 + \mu_5 - \mu_6 - Q_i(t) - a_i(t)] + k_i^u(f_i^u)^2 d_i[\mu_2 + V]$.

4.3.2. Optimal offloading task strategy

Compute the first-order partial derivative of $\mathcal{L}_i$ with respect to $T_i^{tr}(t)$,

$$\frac{\partial \mathcal{L}_i}{\partial T_i^{tr}(t)} = \frac{\partial \mathcal{U}_i(t)}{\partial T_i^{tr}(t)} + \mu_1 r_i^{tr} + \mu_2 p_i + \mu_3 - \mu_4. \quad (33)$$

By solving for $\partial\mathcal{L}_i/\partial T_i^{tr}(t) = 0$, the optimal task transmission time $T_i^{tr}(t)$ can be obtained as

$$T_i^*(t) = \frac{V\chi}{y\ln 2} - \frac{1}{r_i^{tr}}, \quad (34)$$

where $y = [V\lambda R_i^e(t)d_i - Q_i(t) - a_i(t) + \mu_1 - Vk_i^u(f_i^u)^2 d_i]r_i^{tr} + [2V + \mu_2]p_i + \mu_3 - \mu_4$.

To ensure that the user i maintains an transmission time $T_i^*(t) \ge 0$, we solve for $T_i^*(t) = 0$ to obtain the maximum price the user i can accept, denoted as $R_i^{e\_\max}(t)$.

$$R_i^{e\_\max}(t) = \frac{\chi}{\lambda d_i \ln 2} + \frac{\mu_4 - \mu_3 - p_i(2V + \mu_2)}{V\lambda d_i r_i^{tr}} + \frac{Q_i(t) + a_i(t) - \mu_1 + Vk_i^u(f_i^u)^2 d_i}{V\lambda d_i}. \quad (35)$$

By referring to Eq. (34), we can obtain the optimal amount of remote offloading tasks (i.e., $q_{i1}^*(t)$) for user i within the tth time slot.

$$q_{i1}^*(t) = T_i^*(t) r_i^{tr}. \quad (36)$$

4.4. Optimal strategy for MEC server

According to the dynamic programming theory, it breaks down the problem into a sequence of sub-problems, and uses the properties of overlapping sub-problems to reduce the amount of computation. In the same way, in long-term optimization problem, each time slot can be considered a stage, and by making optimal strategies in each stage, optimal result can be achieved in the long run. According to [17], the optimization problem $\boldsymbol{E1}$ can be transformed into $\boldsymbol{E2}$ as

$$\boldsymbol{E2}: \max_{\boldsymbol{R}(t)} \mathcal{S}(t) = \pi^e(t) - \epsilon^e(t) \quad (37)$$

$$\text{s.t.}(14).$$

For the MEC server, the optimal price (i.e., $R_i^e(t)$) for each user needs to be determined. By combining Eqs. (36) and (37), the first-order partial derivative of $\mathcal{S}(t)$ with respect to $R_i^e(t)$ is calculated as

$$\frac{\partial \mathcal{S}(t)}{\partial R_i^e(t)} = \begin{cases} q_{i1}^*(t)d_i & \text{if} q^e < q^c(t); \\ -\frac{V^2\lambda d_i^2[r_i^{tr}]^2\chi}{y^2\ln 2}[R_i^e(t) - R^c], & \\ q_{i1}^*(t)d_i & \text{if} q^e \ge q^c(t); \\ -\frac{V^2\lambda d_i^2[r_i^{tr}]^2\chi}{y^2\ln 2}[R_i^e(t) - c^e k^e (f^e)^2], & \end{cases} \quad (38)$$

and the second-order partial derivative of $\mathcal{S}(t)$ with respect to $R_i^e(t)$ is calculated as

$$\frac{\partial^2 \mathcal{S}(t)}{\partial R_i^e(t)^2} = \begin{cases} \frac{2V^2\lambda d_i^2[r_i^{tr}]^2\chi z_1}{y^3\ln 2}, & \text{if} q^e < q^c(t); \\ \frac{2V^2\lambda d_i^2[r_i^{tr}]^2\chi z_2}{y^3\ln 2}, & \text{if} q^e \ge q^c(t); \end{cases} \quad (39)$$

where $z_1 = [Q_i(t) + a_i(t) - \mu_1 + V k_i^u (f_i^u)^2 d_i - V\lambda R_i^c(t) d_i] r_i^{tr} - [2V + \mu_2] p_i - \mu_3 + \mu_4$ and $z_2 = [Q_i(t) + a_i(t) - \mu_1 + V k_i^u (f_i^u)^2 d_i - V\lambda c^e k^e (f^e)^2 d_i] r_i^{tr} - [2V + \mu_2] p_i - \mu_3 + \mu_4$.

Due to $y > 0$ and the numerator of the second derivative of $\mathcal{S}(t)$ with respect to $R_i^e(t)$ being constant when the MEC server makes game decisions, the sign of the second derivative depends on the sign of the numerator. Consequently, in any iteration, $\mathcal{S}(t)$ is either a concave or convex function relative to $R_i^e(t)$. If $\mathcal{S}(t)$ exhibits convexity concerning $R_i^e(t)$, then the optimal solution of $R_i^e(t)$ is at one of the endpoints of the constraint range. Therefore, we can deduce that the optimal solution for $R_i^e(t)$, denoted as $R_i^*(t)$, is calculated as

$$R_i^*(t) = R_i^{e\_\max}(t). \tag{40}$$

If $\mathcal{S}(t)$ exhibits concavity concerning $R_i^e(t)$, and it is straightforward to ascertain that the constraints of optimization problem $\boldsymbol{E2}$ are affine, then we can employ the Lagrange multiplier method to obtain the solution for $R_i^*(t)$. The Lagrangian function for Eq. (37) is expressed as

$$\mathcal{L}_e(\boldsymbol{R(t)}, \boldsymbol{\eta}) = \mathcal{S}(t) - \eta_1 R_i^e(t) + \eta_2 [R_i^e(t) - R_i^{e\_\max}(t)], \tag{41}$$

where $\boldsymbol{\eta} = \{\eta_1, \eta_2\}$ denotes the Lagrange multipliers, with each multiplier being non-negative.

By solving for $\partial \mathcal{L}_e / \partial R_i^e(t) = 0$, the optimal processing price $R_i^*(t)$ is calculated as

$$R_i^*(t) = \begin{cases} [q_{i1}^*(t) d_i - \eta_1 + \eta_2] \frac{y^2 \ln 2}{V^2 \lambda d_i^2 (r_i^{tr})^2 \chi} & \text{if} q^e < q^c(t); \\ +R^c, & \\ [q_{i1}^*(t) d_i - \eta_1 + \eta_2] \frac{y^2 \ln 2}{V^2 \lambda d_i^2 (r_i^{tr})^2 \chi} & \text{if} q^e \geq q^c(t); \\ +c^e k^e (f^e)^2, & \end{cases} \tag{42}$$

4.5. Stackelberg equilibrium analysis

In this section, we demonstrate that the optimal strategy $\left(T_i^*(t), R_i^*(t)\right)$, $i \in N$ between the users and the MEC server is the Stakelberg equilibrium solution. We consider the MEC server as a leader in a Stackelberg game, and users as followers. For simplicity, the game equilibrium within a time slot will be analyzed. The Stackelberg equilibrium is defined in the following.

Definition 1. $\left(T_i^{SE}(t), R_i^{SE}(t)\right)$ is a Stackelberg equilibrium solution when the price $R_i^*(t)$ of leader is determined, and $T_i^{SE}(t)$ satisfies

$$\mathcal{U}_i\left(T_i^{SE}(t)\right) = \inf_{T_i^{\min}(t) \leq T_i^{tr}(t) \leq T_i^{\max}(t)} \left\{\mathcal{U}_i\left(T_i^{tr}(t)\right)\right\}, \forall t \in \mathcal{T}, \tag{43}$$

and when the task transmission time $T_i^*(t)$ is determined, and $R_i^{SE}(t)$ satisfies

$$\mathcal{S}\left(R_i^{SE}(t)\right) = \sup_{R_i^{\min}(t) \leq R_i^e(t) \leq R_i^{\max}(t)} \left\{\mathcal{S}\left(R_i^e(t)\right)\right\}, \forall t \in \mathcal{T}. \tag{44}$$

Next, it will be verified whether the optimal solution $\left(T_i^*(t), R_i^*(t)\right)$ is the Stackelberg equilibrium solution $\left(T_i^{SE}(t), R_i^{SE}(t)\right)$.

Lemma 2. *If the price $R_i^*(t)$ of the leader is fixed, the function $\mathcal{U}_i\left(T_i^{tr}(t)\right)$ of user i takes the minimum value at T_i^*.*

Proof. Based on Eq. (29), $\mathcal{U}_i(t)$ exhibits convexity concerning $T_i^{tr}(t)$, indicating that the function $\mathcal{U}_i(t)$ attains its minimum value at $T_i^*(t)$. According to Definition 1, $T_i^*(t)$ is the $T_i^{SE}(t)$.

Lemma 3. *For users, the optimal task transmission time $T_i^*(t)$ decreases with the increased price $R_i^e(t)$.*

Proof. By taking the first-order partial derivative of Eq. (34) with respect to $R_i^e(t)$, $\partial T_i^*(t) / \partial R_i^e(t) = -V^2 \chi \lambda d_i r_i^{tr} / (y \ln 2)^2$ is obtained. It is evident that $\partial T_i^*(t) / \partial R_i^e(t) < 0$, indicating that $T_i^{tr}(t)$ decreases as $R_i^e(t)$ increases. This implies that as the price of the MEC server increases, users are less willing to offload tasks.

Lemma 4. *If the optimal task transmission time $T_i^*(t)$ of user i is fixed, the $\mathcal{S}\left(R_i^e(t)\right)$ of the MEC server takes the maximum value at $R_i^*(t)$.*

Proof. Based on the optimal strategy for the MEC server discussed in this section of the paper, it can be concluded that the optimization function (i.e., $\mathcal{S}(t)$) of the MEC server is either concave or convex with respect to the variable $R_i^e(t)$ in each iteration. When the optimization function $\mathcal{S}(t)$ is convex, the efficiency of the MEC server reaches its maximum at $R_i^e(t) = R_i^*(t)$, as shown in Eq. (40). Similarly, when $\mathcal{S}(t)$ is concave, the efficiency of the MEC server reaches its maximum at $R_i^e(t) = R_i^*(t)$, as indicated by Eq. (42). According to Definition 1, $R_i^{SE}(t) = R_i^*(t)$.

In conclusion, $\left(T_i^*(t), R_i^*(t)\right)$ represents the optimal decision for task transmission time and price, and it is also the Stackelberg equilibrium solution $\left(T_i^{SE}(t), R_i^{SE}(t)\right)$.

5. Multi-device task offloading and pricing mechanism algorithm

We will delve into the updates of Lagrange multipliers and MEC server prices in this section. At the end of this section, pseudocode for the algorithm that describes the game process between users and the MEC server is provided.

5.1. Lagrange multiplier update strategy

According to Eqs. (32) and (34), the optimal local task workload and task transmission time can be determined, respectively. As the results are obtained using the Lagrange multiplier method, it is necessary to update these multipliers to ensure the satisfaction of constraints during the optimization process. A standard subgradient method is employed to update the Lagrange multipliers (i.e., $\boldsymbol{\mu}$). The update method is shown as

$$\begin{aligned} \mu_1 &= \left\{\mu_1 + \alpha\left[q_i(t) - Q_i(t)\right]\right\}^+, \\ \mu_2 &= \left\{\mu_2 + \alpha\left[H_i^u(t) - B_i(t)\right]\right\}^+, \\ \mu_3 &= \left\{\mu_3 + \alpha\left[T_i^*(t) - \tau\right]\right\}^+, \\ \mu_4 &= \left[\mu_4 - \alpha T_i^*(t)\right]^+, \\ \mu_5 &= \left\{\mu_5 + \alpha\left[q_{i0}^*(t) - \frac{f_i^u \tau}{d_i}\right]\right\}^+, \\ \mu_6 &= \left[\mu_6 - \alpha q_{i0}^*(t)\right]^+, \end{aligned} \tag{45}$$

where α represents the iteration step size and $[x]^+ = \max\{0, x\}$.

The local task workload and task transmission time for each user are computed by updating the Lagrange multipliers and the price in each iteration.

5.2. Price update strategy

In one iteration, when the function $\mathcal{S}(t)$ exhibits convexity concerning the price $R_i^e(t)$, the optimal price $R_i^*(t) = R_i^{e\_\max}(t)$. However, when the function $\mathcal{S}(t)$ is concave with respect to the price $R_i^e(t)$, it is difficult to compute the optimal price $R_i^*(t)$ based on $\partial \mathcal{S}(t) / \partial R_i^e(t) = 0$. To address this, the gradient ascent method is employed to update the price $R_i^e(t)$, using the first-order partial derivative of the MEC server utility function with respect to the price as the Marginal Utility [26] to update the price. The update expression can be derived as

$$R_i^{e\_\kappa+1}(t) = R_i^{e\_\kappa}(t) + \beta \frac{\partial \mathcal{S}(t)}{\partial R_i^{e\_\kappa}(t)}, \tag{46}$$

where β represents the iteration step size and κ represents the number of iterations within the current time slot.

Due to the constraint (14), after one iteration, the service price of the MEC server is calculated as

$$R_i^{e\_\kappa+1}(t) = \min\left\{R_i^{e\_\max}(t), \max\left\{0, R_i^{e\_\kappa}(t)\right\}\right\}. \tag{47}$$

Finally, the price of the $(\kappa+1)$-th iteration within the tth time slot can be expressed as

$$R_i^{e\_\kappa+1}(t) = \begin{cases} R_i^{e\_\kappa}(t), & \text{if} \frac{\partial^2 S(t)}{\partial R_i^{e\_\kappa}(t)^2} \le 0; \\ R_i^{e\_\max}(t), & \text{if} \frac{\partial^2 S(t)}{\partial R_i^{e\_\kappa}(t)^2} > 0; \end{cases} \tag{48}$$

To provide a clearer description of the price updating process, the pseudocode for price updating is presented in the Alg. 1.

Algorithm 1 Price Update Algorithm

```
1: Input:$k_i^u$, $f_i^u$, $d_i$, $\tau$, $p_i$, $\boldsymbol{\mu}$, $r_i^{tr}$, $k^e$, $V$, $\lambda$, $\chi$, $f^e$, $c^e$, $R^c$, $q^c$, $i \in N$;
2: Output:$R_i^e(t), i \in N$;
3: for all $i \in N$ do
4:     Calculate $\partial S(t)/\partial R_i^e(t)$ based on Eq. (38);
5:     Calculate $\partial^2 S(t)/\partial R_i^e(t)^2$ based on Eq. (39);
6:     Calculate $R_i^{e\text{-}\max}(t)$ based on Eq. (40);
7:     Calculate $R_i^{e\_it}(t)$ based on Eq. (47);
8:     Calculate $R_i^e(t)$ based on Eq. (48);
9: end for
```

5.3. Multi-device task offloading and pricing mechanism algorithm

By integrating the content in Section 4 and Section 5 of this paper, the optimization of $q_{i0}(t)$, $T_i^{tr}(t)$, and $R_i^e(t)$ for each time slot is formulated. The implementation process of the proposed Multi-Device Task Offloading and Pricing Mechanism Algorithm is outlined in the Alg. 2. The core of this algorithm lies in lines 8 to 22 of the pseudocode. In pseudocode, lines 8 to 13 describe how each user calculates local processing tasks and transfer time based on price. Lines 14 to 18 demonstrate the redistribution of transmission time for each user using time slot constraints when the total transmission time of all users exceeds one time slot. Lines 19 to 22 illustrate how the MEC server updates the prices based on each user's offloading tasks. The time complexity of the algorithm primarily arises from the iterative computation of task offloading and pricing for each user device. In each time slot, the time complexity is $O(D \cdot N)$, where D is the number of iterations and N is the number of users.

6. Performance evaluation

In order to assess the validity of MDTOPMA, we will organize three distinct sets of simulations. First of all, we demonstrate the convergence of the game through iterative experiments within a time slot. And the stability of the queue is validated through experiments spanning multiple time slots. Second, under the premise of confirming the existence of game equilibrium, we will focus on parameter tuning experiments to evaluate the impact of different parameters on performance. Finally, an empirical study is conducted to compare it with the benchmark schemes.

6.1. Simulation setting

For each user, the CPU computing frequency f_i^u is uniformly distributed within $[0.9, 1.2]$ GHz [15] and the capacitance switching coefficient k_i^u is distributed within $[10^{-28}, 10^{-27}]$ [18,19]. The transmission power p_i is uniformly distributed within [100,150] mW [11]. The task arrival quantity $a_i(t)$ follows a uniform distribution within the range

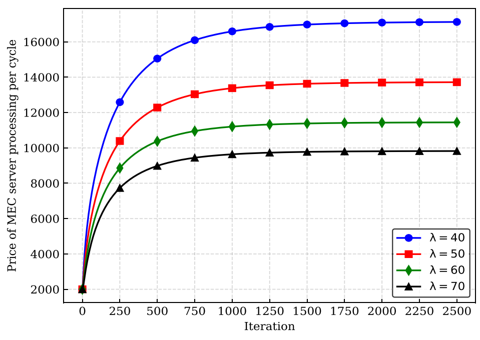
Algorithm 2 Multi-Device Task Offloading and Pricing Mechanism Algorithm (MDTOPMA)

```
1: Input:$k_i^u$, $f_i^u$, $d_i$, $\tau$, $p_i$, $B_i^{\max}$, $w$, $h_i$, $n_0$, $k^e$, $f^e$, $c^e$, $R^c$, $i \in N$;
2: Output:optimal solution $q_{i0}^*(t)$, $T_i^*(t)$, $R_i^*(t)$, $i \in N$, $t \in \mathcal{T}$, $totalIt$;
3: initial $\lambda$, $\alpha$, $\beta$, $Q_i(0)$, $B_i(0)$, $\boldsymbol{\mu} \leftarrow [\mu_1, ..., \mu_6]$, $\boldsymbol{\sigma} \leftarrow [\sigma_1, ..., \sigma_6]$;
4: $\boldsymbol{\mu^{pre}} \leftarrow \boldsymbol{\mu}$;
5: for all $i \in N$ do
6:     Calculate $r_i^{tr}$ by Eq. (10);
7: end for
8: for $t \in \mathcal{T}$ do
9:     $it \leftarrow 0$;
10:    while $it \le totalIt$ or $\boldsymbol{\mu} - \boldsymbol{\mu^{pre}} \ge \boldsymbol{\sigma}$ do
11:        for all $i \in N$ do
12:            Calculate $q_{i0}(t)$ and $T_i^{tr}(t)$ according to
13:            Eq. (32) and Eq. (34) respectively;
14:            $time \leftarrow time + T_i^{tr}(t)$;
15:        end for
16:        if $time > \tau$ then
17:            for all $i \in N$ do
18:                $T_i^{tr}(t) \leftarrow \tau T_i^{tr}(t)/time$;
19:            end for
20:        end if
21:        for $i \in N$ do
22:            $q_{i1}^c(t) \leftarrow T_i^{tr}(t) r_i^{tr}/d_i$;
23:            $q^c \leftarrow q^c + q_{i1}^c(t)$, $i \leftarrow i + 1$;
24:        end for
25:        Call Algorithm 1;
26:        $\boldsymbol{\mu^{pre}} \leftarrow \boldsymbol{\mu}$, calculate $\boldsymbol{\mu}$ by Eq. (45);
27:        $it \leftarrow it + 1$;
28:    end while
29:    $q_{i0}^*(t) \leftarrow q_{i0}(t)$, $T_i^*(t) \leftarrow T_i^{tr}(t)$, $R_i^*(t) \leftarrow R_i^e(t)$;
30: end for
```

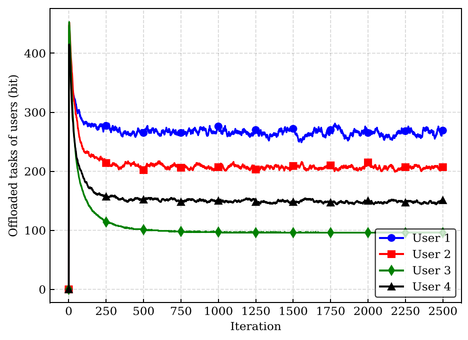
of [2000, 6000] bits [11]. The bandwidth of the base station w is set to 20 MHz [27]. The task computation density d_i is uniformly distributed within $[2,12]\times 10^3$ cycles/bit. The time slot duration τ is configured as 0.1 s [28]. λ is set to 100. The noise power density n_0 is set to 10^{-9} W/Hz. The channel gain adheres to an exponential distribution, denoted as E(1) [29]. The harvested energy $e_i(t)$ is uniformly distributed within [0, 20] mJ/s [30]. For the MEC server, the CPU computing frequency f^e is set to 3 GHz [27] and the capacitance switching coefficient k^e is set to 10^{-28} [19]. All the simulations are performed on a workstation PC with an Intel i5 12600KF processor, 16 GB of memory, and Windows 10 operating system.

6.2. Game convergence simulation experiments

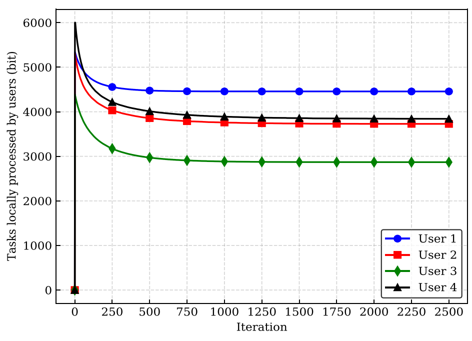
In this simulation, we primarily focus on the convergence of the Stackelberg game and the stability of the system. For ease of observation, there are 4 users considered in this simulation. In Fig. 2, we plot the differentiated prices of the MEC server, tasks offloaded by users, tasks processed locally by users, and user utility as the number of iterations accumulates. Fig. 2(a) shows that the prices of the MEC server tend to stabilize after a certain number of iterations. In the figure, the final price obtained by each user through the game is different. This is attributed to the heterogeneity among users, for instance, a certain user with higher energy consumption when processing tasks locally tends to lean towards offloading tasks. By observing Eq. (42), it can be seen that $\partial^2 R_i^*(t)/\partial q_{i1}^*(t)^2 > 0$, the MEC server will increase the service price of the user accordingly. In addition, since each user's task arrival and energy collection may be different, this will also result in varying task size offloaded by each user, consequently leading to differentiated pricing by the MEC server. Fig. 2(b) and Fig. 2(c) demonstrate that the offloaded tasks and locally processed tasks for each user tend to



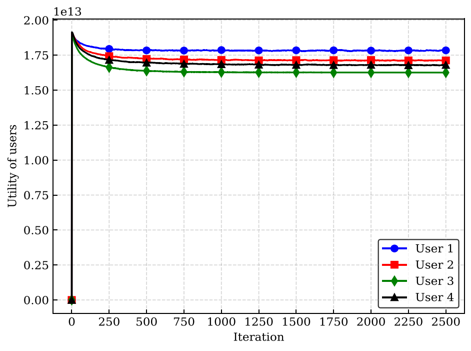
(a) Price of the MEC server processing per cycle



(b) Offloaded task of users

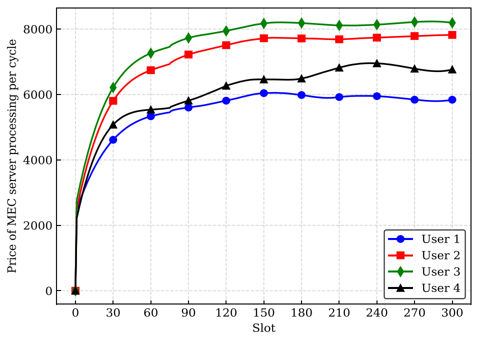


(c) Tasks locally processed by users

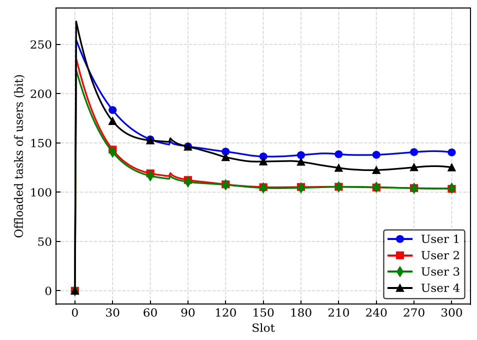


(d) Utility of users

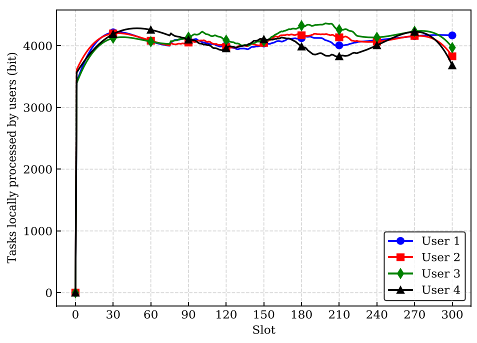
Fig. 2. Price, offloaded tasks, locally processed tasks, and user utility versus iteration.



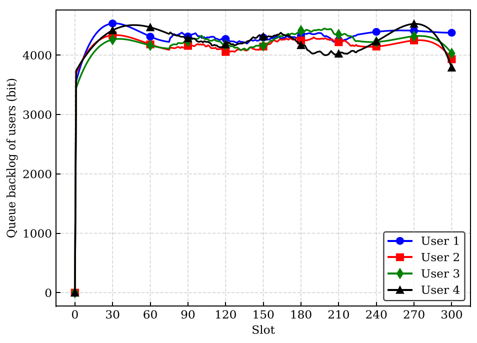
(a) Price of the MEC server processing per cycle



(b) Offloaded tasks of users

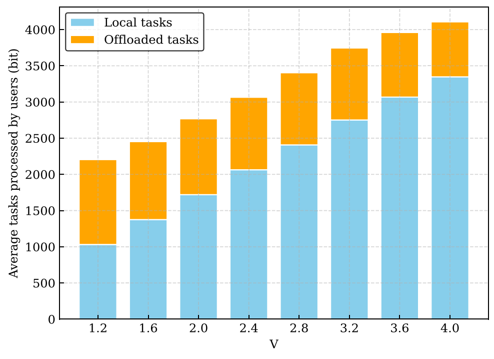


(c) Tasks locally processed by users

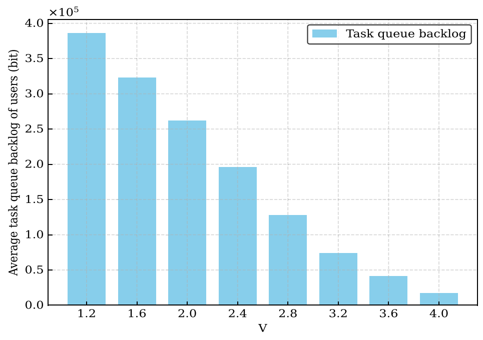


(d) Queue backlog of users

Fig. 3. Price, offloaded tasks, locally processed tasks, and queue backlog versus time.



(a) Average tasks processed by users



(b) Average task queue backlog of users

Fig. 4. Average processed tasks and queue backlog vary with different values of V.

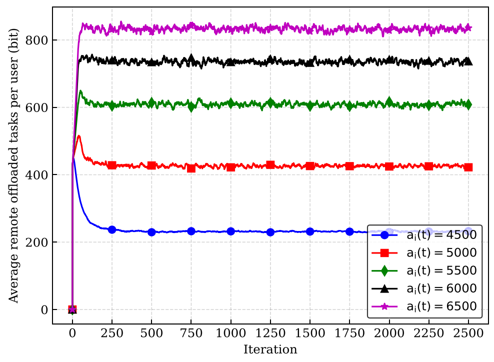


Fig. 5. Average remote offloaded tasks with different values of $a_i(t)$.

stabilize after a certain number of iterations. The trend in Fig. 2(b) exhibits a characteristic of initially increasing and then decreasing. This phenomenon stems from the relatively low initial prices depicted in Fig. 2(a), resulting in a sharp increase in the tasks offloaded by users. However, as prices rise, the offloaded tasks gradually decrease, aligning with the anticipated trend. Additionally, Lemma 3 can be used to substantiate this observation. If the MEC server offers a higher price to a particular user, that user typically reduces the amount of tasks offloaded, such as user 3. Based on the three previous figures in Fig. 2, it can be inferred that user utility also tends to stabilize with the number of iterations, which aligns with the results shown in Fig. 2(d). According to the results in Fig. 2, it can be confirmed that there exists a Stackelberg equilibrium between the users and the MEC server within a time slot.

In Fig. 3, we present the variations of the MEC server prices, user offloaded tasks, user local processing tasks, and the backlog in the user task queue with varying time slots. Fig. 3(a) illustrates the gradual stabilization of differential prices provided by the MEC server to every user during long-term evolution. Due to the heterogeneity of users and the varying task arrival rates and energy harvesting conditions in each time slot, the prices calculated through the game may also differ for each user. Fig. 3(b) depicts the increasing trend of remote offloaded tasks for each user starting from zero and gradually decreasing thereafter. This pattern can be attributed to the initially low prices shown in Fig. 3(a), which encourage users to offload more tasks. However, as the prices subsequently rise, users' inclination for offloading diminishes, leading to a reduction in the amount of offloaded tasks. Fig. 3(c) and Fig. 3(d) illustrate the steady state of user local processing tasks and user task queue during long-term evolution, respectively. The arrival of tasks in each time slot is uncertain, so the queue in Fig. 3(d) shows a state of continuous ups and downs. But overall, the task queue is gradually stabilizing. This is consistent with what we expect to achieve with Lyapunov optimization.

Observing both Fig. 2 and Fig. 3, it is evident that the game converges in both short-term and long-term scenarios, ensuring the long-term stability of the MEC system.

6.3. Parameter tuning simulation experiments

In this simulation, multiple experiments are conducted to demonstrate the dynamic task offloading and update of prices with consideration of 20 users. In Fig. 4, the average processed tasks and queue backlog for users are plotted with varying parameter V. Fig. 4(a) depicts how the average processed tasks correlates with the parameter V. The observed phenomenon shows that as V decreases, the average processing tasks present a downward trend. According to Eq. (25), as the parameter V decreases, the proportion of $-Q_i(t)$ in the minimization Eq. (25) increases, which means that the task queue backlog will show an upward trend. Therefore, it can be deduced that as V decreases, the average processing tasks of users will show a downward trend. This aligns with the findings exhibited in Fig. 4(a). Fig. 4(b) illustrates how the average queue backlog varies with the parameter V. As V increases, we can observe a decreasing trend in the queue backlog. This phenomenon aligns with the trend of increasing processed tasks shown in Fig. 4(a).

Fig. 5, the average remote offloaded tasks for users are plotted with varying $a_i(t)$. It can be observed that as the task arrival rate increases, the average number of tasks offloaded by users also increases. This is because with an increase in the task arrival rate, the backlog in the task queue also grows. To maintain queue stability, users tend to offload more tasks.

In Fig. 6, we plot the offloaded tasks, prices, and the MEC server utility with varying values of the parameter λ. In Fig. 6(a), as λ increases, the convergence price of the MEC server decreases. This is because λ is weighted on the offloading cost function (i.e., $c_i(t)$) of users. With increase of λ, the proportion of user offloading costs increases, leading to a decreased inclination among users for task offloading. Consequently, MEC server stimulate task offloading by reducing prices. In Fig. 6(b), as the value of λ decreases, users tend to offload more tasks. This is because a lower λ results in a smaller proportion of user offloading costs, leading to an increased desire for task offloading. Fig. 6(c) illustrates that the effectiveness of the MEC server increases with lower λ. This experimental result can be calculated based on the experimental data in Fig. 6(a) and Fig. 6(b).

6.4. Comparison with benchmark schemes

To further evaluate the MDTOPMA′s performance, there are three schemes compared with MDTOPMA:

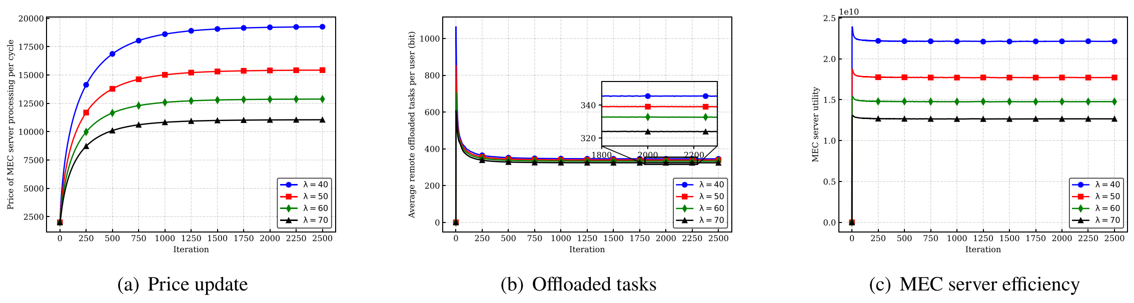


Fig. 6. Price, offloaded tasks, and the MEC server utility with different values of λ.

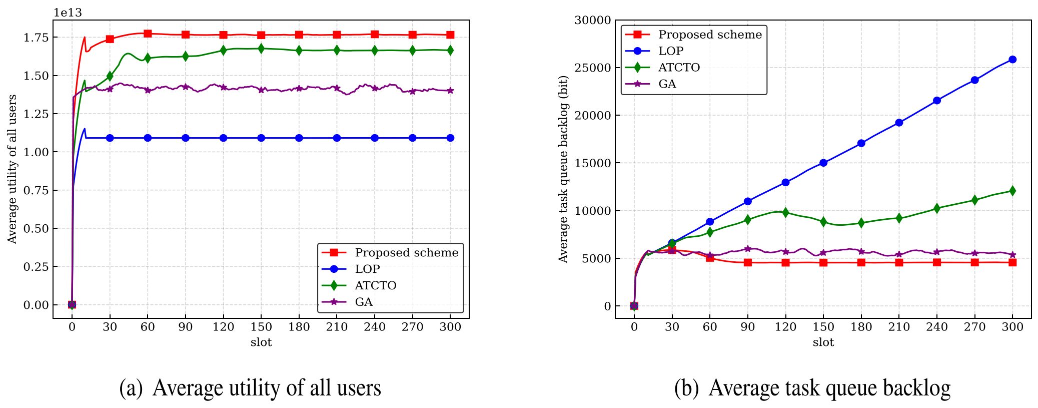


Fig. 7. User utility and task queue backlog with four different schemes.

- **Local-Only Processing (LOP)** scheme [19]: In each time slot, users solely process all tasks locally, i.e., $T_i^*(t) = 0$.
- **Average Time-Constrained Task Offloading (ATCTO)** scheme: In each time slot, the channel transmission time for each user's offloaded tasks must not surpass a predefined threshold, i.e., $T_i^{tr}(t) \leq \tau/n$. The ATCTO scheme is inspired by the Equal Allocation Strategy from [11], which emphasizes fairly distributing the offloading time among users.
- **Genetic Algorithm (GA)** scheme: In this study, we use a genetic algorithm for comparison and initialize a population of 20 individuals, each representing offloading strategies for users and a pricing strategy for the MEC server. After multiple iterations, we select the individual with the highest fitness for comparison.

In Fig. 7, the average user utility and average queue backlog are plotted alongside different algorithm schemes. In the experimental results, our MDTOPMA surpasses the other schemes in user utility and task queue backlog. In Fig. 7(a), it can be observed that the improvement in user utility by LOP is significantly smaller than other schemes. This is because LOP only allows users to process tasks locally, neglecting the computing capacity of the MEC server, and the energy consumption of remote offloading is generally lower than the energy consumption generated by local computing tasks. Fig. 7(b) demonstrates that the task queue backlog of LOP is significantly higher than that of our proposed algorithm. This is mainly due to the fact that LOP does not utilize the MEC server's computational resources. In Fig. 7, the performance of ATCTO is less than our scheme. For instance, if a user calculates the optimal transmission time T, such that $T_i^*(t) > \tau/n$, but is constrained by $T_i^{tr}(t) \leq \tau/n$, it results in the user being unable to achieve the optimal performance. Although ATCTO allows each user to have a transmission time ranging from 0 to τ/n, ensuring that each user has a fair opportunity to utilize network resources, achieving better performance is challenging. In Fig. 7, GA's task queue performance is better than ATCTO, but the user benefit is lower than ATCTO. This is due to the randomness of GA. Even when the price is high, users may still choose to remotely offload many tasks, resulting in reduced utilities.

7. Conclusion

In this paper, we study a task offloading problem for a MEC system consisting of three layers of cloud–edge-device, where each user terminal device supports EH. While solving the optimization problem, the Lyapunov optimization theory is applied to convert the long-term problem into problem of each time slot and stabilize the task queue of each user. In order to reasonably allocate resources between the users and the MEC server, the Stackelberg game theory is employed to regulate resources. Combining the above two theories, we apply the MDTOPMA to solve the optimal offloading strategy of each user and the pricing strategy of the MEC server in each time slot. The simulation experiment results indicate that, when compared to other algorithms, MDTOPMA not only enhances user benefits but also reduces the backlog of user task queues. In the simulation experiment of parameter performance optimization, the adjustment of parameter value also leads to better choice of offloading decision and pricing decision.

CRediT authorship contribution statement

Jing Mei: Writing – review & editing, Investigation, Funding acquisition, Formal analysis. **Cuibin Zeng:** Writing – original draft, Resources, Methodology. **Zhao Tong:** Writing – review & editing, Methodology, Funding acquisition, Conceptualization. **Longbao Dai:** Writing – review & editing, Investigation, Conceptualization. **Keqin Li:** Writing – review & editing, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the Program of National Natural Science Foundation of China (grant No. 62072174, 61502165), Provincial Natural Science Foundation of Hunan, China (grant No. 2020JJ5370, 2022JJ40278, 2023JJ30083), Scientific Research Fund of Hunan Provincial Education Department, China (grant No. 22A0026, 22A0592), Graduate Research Innovation-Program of Hunan Province (Grant No. CX20240548).

Data availability

Data will be made available on request.

References

[1] Ana Bera, 80 IoT statistics (infographic), 2019, https://safeatlast.co/blog/iot-statistics/, Website.

[2] K.P. Naveen, R. Sundaresan, Double-auction mechanisms for resource trading markets, IEEE/ACM Trans. Netw. 29 (3) (2021) 1210–1223, http://dx.doi.org/10.1109/TNET.2021.3058251.

[3] D. Ma, G. Lan, M. Hassan, W. Hu, S.K. Das, Sensing, computing, and communications for energy harvesting iots: A survey, IEEE Commun. Surv. & Tutorials 22 (2) (2020) 1222–1250, http://dx.doi.org/10.1109/COMST.2019.2962526.

[4] J. Peng, H. Qiu, J. Cai, W. Xu, J. Wang, D2d-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for mec, IEEE Trans. Wirel. Commun. 20 (8) (2021) 4858–4873, http://dx.doi.org/10.1109/TWC.2021.3062616.

[5] S.-H. Kim, S. Park, M. Chen, C.-H. Youn, An optimal pricing scheme for the energy-efficient mobile edge computation offloading with ofdma, IEEE Commun. Lett. 22 (9) (2018) 1922–1925, http://dx.doi.org/10.1109/LCOMM.2018.2849401.

[6] M. Li, Q. Wu, J. Zhu, R. Zheng, M. Zhang, A computing offloading game for mobile devices and edge cloud servers, Wirel. Commun. Mob. Comput. 2018 (2018) 1–10.

[7] Z. Liu, J. Fu, Y. Zhang, Computation offloading and pricing in mobile edge computing based on stackelberg game, Wirel. Netw. 27 (7) (2021) 4795–4806, http://dx.doi.org/10.1007/s11276-021-02767-z.

[8] Z. Ning, J. Huang, X. Wang, J.J.P.C. Rodrigues, L. Guo, Mobile edge computing-enabled internet of vehicles: Toward energy-efficient scheduling, IEEE Netw. 33 (5) (2019) 198–205, http://dx.doi.org/10.1109/MNET.2019.1800309.

[9] Y. Mao, J. Zhang, K.B. Letaief, Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems, in: 2017 IEEE Wireless Communications and Networking Conference, WCNC, 2017, pp. 1–6, http://dx.doi.org/10.1109/WCNC.2017.7925615.

[10] X. Zhao, L. Zhao, K. Liang, An energy consumption oriented offloading algorithm for fog computing, in: Quality, Reliability, Security and Robustness in Heterogeneous Networks: 12th International Conference, QShine 2016, Seoul, Korea, July 7–8, 2016, Proceedings 12, Springer, 2017, pp. 293–301.

[11] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, X. Shen, Energy efficient dynamic offloading in mobile edge computing for internet of things, IEEE Trans. Cloud Comput. 9 (3) (2021) 1050–1060, http://dx.doi.org/10.1109/TCC.2019.2898657.

[12] F. Li, H. Yao, J. Du, C. Jiang, Y. Qian, Stackelberg game-based computation offloading in social and cognitive industrial internet of things, IEEE Trans. Ind. Inform. 16 (8) (2020) 5444–5455, http://dx.doi.org/10.1109/TII.2019.2961662.

[13] X. Hu, K.-K. Wong, K. Yang, Wireless powered cooperation-assisted mobile edge computing, IEEE Trans. Wirel. Commun. 17 (4) (2018) 2375–2388, http://dx.doi.org/10.1109/TWC.2018.2794345.

[14] F. Wang, J. Xu, X. Wang, S. Cui, Joint offloading and computing optimization in wireless powered mobile-edge computing systems, IEEE Trans. Wirel. Commun. 17 (3) (2018) 1784–1797, http://dx.doi.org/10.1109/TWC.2017.2785305.

[15] P.K. Bishoyi, S. Misra, Enabling green mobile-edge computing for 5g-based healthcare applications, IEEE Trans. Green Commun. Netw. 5 (3) (2021) 1623–1631, http://dx.doi.org/10.1109/TGCN.2021.3075903.

[16] F. Zeng, Q. Chen, L. Meng, J. Wu, Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing, IEEE Trans. Intell. Transp. Syst. 22 (6) (2021) 3247–3257, http://dx.doi.org/10.1109/TITS.2020.2980422.

[17] S. Xia, Z. Yao, Y. Li, S. Mao, Online distributed offloading and computing resource management with energy harvesting for heterogeneous mec-enabled iot, IEEE Trans. Wirel. Commun. 20 (10) (2021) 6743–6757, http://dx.doi.org/10.1109/TWC.2021.3076201.

[18] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, F. Tian, Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud ran, IEEE Int. Things J. 7 (4) (2020) 3282–3299, http://dx.doi.org/10.1109/JIOT.2020.2967502.

[19] M. Guo, W. Wang, X. Huang, Y. Chen, L. Zhang, L. Chen, Lyapunov-based partial computation offloading for multiple mobile devices enabled by harvested energy in mec, IEEE Int. Things J. 9 (11) (2022) 9025–9035, http://dx.doi.org/10.1109/JIOT.2021.3118016.

[20] C. Qiu, Y. Hu, Y. Chen, Lyapunov optimized cooperative communications with stochastic energy harvesting relay, IEEE Int. Things J. 5 (2) (2018) 1323–1333, http://dx.doi.org/10.1109/JIOT.2018.2793850.

[21] B. Cao, S. Xia, J. Han, Y. Li, A distributed game methodology for crowdsensing in uncertain wireless scenario, IEEE Trans. Mob. Comput. 19 (1) (2020) 15–28, http://dx.doi.org/10.1109/TMC.2019.2892953.

[22] M. Tao, K. Ota, M. Dong, H. Yuan, Stackelberg game-based pricing and offloading in mobile edge computing, IEEE Wirel. Commun. Lett. 11 (5) (2022) 883–887, http://dx.doi.org/10.1109/LWC.2021.3138938.

[23] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, H. Zhang, Incentive mechanism for computation offloading using edge computing: A stackelberg game approach, Comput. Netw. 129 (DEC.24) (2017) 399–409.

[24] Y. Li, S. Xia, M. Zheng, B. Cao, Q. Liu, Lyapunov optimization-based trade-off policy for mobile cloud offloading in heterogeneous wireless networks, IEEE Trans. Cloud Comput. 10 (1) (2022) 491–505, http://dx.doi.org/10.1109/TCC.2019.2938504.

[25] X. Chen, Z. Lu, W. Ni, X. Wang, F. Wang, S. Zhang, S. Xu, Cooling-aware optimization of edge server configuration and edge computation offloading for wirelessly powered devices, IEEE Trans. Veh. Technol. 70 (5) (2021) 5043–5056, http://dx.doi.org/10.1109/TVT.2021.3076057.

[26] M.S.S. Rao, S.A. Soman, Marginal pricing of transmission services using min–max fairness policy, IEEE Trans. Power Syst. 30 (2) (2015) 573–584, http://dx.doi.org/10.1109/TPWRS.2014.2331424.

[27] L. Chen, S. Zhou, J. Xu, Computation peer offloading for energy-constrained mobile edge computing in small-cell networks, IEEE/ACM Trans. Netw. 26 (4) (2018) 1619–1632, http://dx.doi.org/10.1109/TNET.2018.2841758.

[28] Y. Dai, K. Zhang, S. Maharjan, Y. Zhang, Deep reinforcement learning for stochastic computation offloading in digital twin networks, IEEE Trans. Ind. Inform. 17 (7) (2021) 4968–4977, http://dx.doi.org/10.1109/TII.2020.3016320.

[29] J. Mei, L. Dai, Z. Tong, X. Deng, K. Li, Throughput-aware dynamic task offloading under resource constant for mec with energy harvesting devices, IEEE Trans. Netw. Serv. Manag. (2023) 1, http://dx.doi.org/10.1109/TNSM.2023.3243629.

[30] X. Lyu, W. Ni, H. Tian, R.P. Liu, X. Wang, G.B. Giannakis, A. Paulraj, Optimal schedule of mobile edge computing for internet of things using partial information, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2606–2615, http://dx.doi.org/10.1109/JSAC.2017.2760186.

Jing Mei received the Ph.D degree in computer science from Hunan University, China, in 2015. She is currently an associate professor in the College of Information Science and Engineering in Hunan Normal University. Her research interests include cloud computing, fog computing and mobile edge computing, high performance computing, task scheduling and resource management, etc. She has published more than 30 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, Journal of Supercomputing.*

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 at 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 received the Ph.D degree in computer science from Hunan University, Changsha, China in 2014. He was a visiting scholar at the Georgia State University from 2017 to 2018. He is currently an associate professor at the College of Information Science and Engineering of Hunan Normal University, the young backbone teacher of Hunan Province, China. His research interests include parallel and distributed computing systems, resource management, big data and machine learning algorithm. He has published more than 25 research papers in international conferences and journals, such as IEEE-TPDS, Information Sciences, FGCS, NCA, and JPDC, PDCAT, etc. He is a senior member of the China Computer Federation (CCF) and a Member of the IEEE.

Longbao Dai received the B.S. degree in computer science and technology from Hunan University of Science and Engineering, Yongzhou, China, in 2021. He is currently working toward the M.S. degree at the College of Information Science and Engineering, Hunan Normal University, Changsha, China. His research interests focus on distributed parallel computing, modeling and resource pricing and allocation in mobile edge computing systems, and combinatorial optimization.

Keqin Li 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. 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 authored or co-authored over 850 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He holds over 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top 5 most influential scientists in parallel and distributed computing in terms of both single–year impact and career–long impact based on a composite indicator of Scopus citation database. 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*. He is an IEEE Fellow and an AAIA Fellow. He is also a Member of Academia Europaea (Academician of the Academy of Europe).