Soft Computing (2023) 27:15095–15110
https://doi.org/10.1007/s00500-023-08695-8

OPTIMIZATION

Maximizing the number of completed tasks in MEC considering time and energy constraints

Haijian Yu[1,2] · Jing Liu[1,2] · Chunhua Deng[1,2] · Cen Chen[3] · Keqin Li[4]

Accepted: 29 May 2023 / Published online: 20 July 2023

Abstract
Many tasks generated by mobile user devices are computation-intensive and latency-sensitive, such as autonomous driving and video analysis. However, due to limited energy and computing capacity, a user device may not be able to complete its task within a given time, leading to a poor user experience. Mobile edge computing (MEC) can address this challenge by offloading tasks to edge servers with stronger computing capacity and more resources for execution, which can save energy of user devices and reduce the task computation time. Different offloading strategies will impact the number of tasks completed, latency, energy overhead and so on. This paper investigates the problem of maximizing the number of completed tasks while minimizing the average completion time, energy overhead and cost in MEC under both time and energy constraints. To solve the problem, we develop the mayfly genetic algorithm (MGA), which jointly optimizes task offloading locations and ratios, central processor unit (CPU) frequencies of user devices and computing capacities allocated to user devices by edge servers. Simulation experiments indicate that MGA outperforms state-of-the-art algorithms in terms of the number of completed tasks.

Keywords Delay · Energy · Mobile edge computing · Task offloading

1 Introduction

With the prevalence of mobile devices in recent years, a considerable number of new applications have emerged, like autonomous driving, video analysis, online games and augmented reality (Wang et al. 2020b; Chen et al. 2020; Weng et al. 2021). These applications generally have higher requirements for computing resources (e.g., CPUs, memory), energy consumption, and latency (Luan et al. 2015; Abbas et al. 2018; Aazam et al. 2018; Saeik et al. 2021). Although CPUs are becoming more and more powerful, the physical size of user devices still limits their computing capacity and available energy, so that they are unable to meet the requirements of some new applications (Aazam et al. 2018). Cloud computing allows user devices to offload their computing tasks to central cloud servers with abundant computing resources and powerful computing capability for execution, thus resolving the issue of insufficient computing resources and energy of user devices. However, cloud servers and user devices are usually far apart, such as tens of kilometers, resulting in long propagation latency (Kubade et al. 2018; Mao et al. 2017), which is not suitable for latency-sensitive applications.

Jing Liu, Chunhua Deng, Cen Chen, Keqin Li have contributed equally to this work.

✉ Jing Liu
luijing_cs@wust.edu.cn

✉ Chunhua Deng
dchzx@wust.edu.cn

Haijian Yu
yuhaijian@wust.edu.cn

Cen Chen
chenc@i2r.a-star.edu.sg

Keqin Li
lik@newpaltz.edu

1 Department Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430000, China

2 Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan 430000, China

3 Institute for Infocomm Research, Singapore 999002, Singapore

4 State University of New York, New Paltz, New York 12561, USA

To figure out the long propagation latency in cloud computing, a new computing conception called mobile edge computing has emerged. MEC extends computing and storage to the edge of the network by placing storage and computing resources close to user devices. This allows for faster processing and lower latency compared to cloud computing. An MEC system is usually composed of edge servers, base stations, user devices, wireless and wired communication links. Edge servers have larger computing capacity than resource-constrained user devices. Offloading tasks from use devices to edge servers for execution not only alleviates the computing pressure on user devices, but also guarantees the constraints such as latency and energy overhead for computation-intensive and latency-sensitive applications.

Low-energy consumption, low latency, and low cost (e.g., money) in a computing system are the eternal pursuit in both industry and academia, making them hot topics all the time. Among these three goals, low latency is conflicted with the other two goals. In addition, an MEC system may have multiple heterogeneous servers and each edge server may contain multiple heterogeneous CPUs, which will allow a task to have multiple offloading strategies. Different task offloading strategies will produce different overheads (e.g., latency, energy consumption). Thus, it is challenging to design efficient task offloading algorithms to optimize latency with energy consumption or cost simultaneously in MEC.

Extensive multiuser computation offloading strategies on energy and time constraints have been studied in MEC. Zhou and Jadoon explored how to minimize task completion time within a given deadline (Zhou and Jadoon 2021). Zhou et al. investigated how to minimize the completion time of tasks and the energy overhead of user device in a rechargeable MEC system with time and energy constraints (Zhou et al. 2021). Authors in Qi et al. (2022); Jahandar et al. (2022); Guo et al. (2022) designed efficient strategies to reduce time, energy consumption and cost to complete a task under time and energy constraints. However, focusing solely on reducing the time, energy and cost of task execution without considering the number of completed tasks in a computing system may result in numerous incomplete tasks as the time, energy and cost of executing tasks are being reduced, thereby seriously affecting user experience quality. Therefore, maximizing the number of completed tasks in a computing system is an essential aspect that cannot be overlooked. A few of work discussed how to maximize the number of completed tasks in MEC. Han et al. (2022) proposed the energy-efficient service placement algorithm to minimize the total power consumption and maximize the number of accepted services in MEC without energy constraint. To the best of our knowledge, no previous research has aimed at maximizing the number of completed tasks while minimizing the average completion time, energy overhead and cost under time and energy constraints in MEC.

In this paper, we study the problem of maximizing the number of completed tasks while minimizing the average energy overhead, completion time and cost with time and energy constraints in MEC. To work out the problem, we design an efficient and novel algorithm called the mayfly genetic algorithm, which combines the global search capability of the genetic algorithm with the fast convergence speed and high accuracy of the mayfly algorithm. Simulated experimental results indicate that our proposed algorithm works well in maximizing the number of completed tasks in given constraints.

In this paper, our contributions are as follows.

- We consider a computation offloading problem with multiple users and multiple edge severs for mobile edge computing, where user devices have strict energy constraints and tasks have deadlines. If the energy required to complete a task exceeds user device's energy limit, or if the task processing time goes beyond the specified deadline during task execution, the task is considered incomplete.
- We formulate the problem of maximizing the number of tasks completed while minimizing the average time, energy and cost of completed tasks in MEC with time and energy constraints by jointly optimizing CPU frequencies of user devices, offloading locations of tasks, offloading ratios of tasks and computing capacities allocated to user tasks by edge servers.
- We propose a novel and efficient algorithm to solve the above problem.
- Compared with the genetic algorithm and the mayfly algorithm, our proposed algorithm is more efficient because it significantly improves the number of tasks completed. Additionally, our proposed algorithm on the whole outperforms state-of-the-art algorithms.

The residual organization of this article is as follows: Section 2 outlines related researches. Section 3 establishes all models and the problem addressed in this paper. Section 4 introduces our proposed algorithm. Section 5 evaluates the performance of the proposed algorithm through extensive simulation experiments. Section 6 concludes the paper.

2 Related researches

The computation offloading problem in edge computing has been extensively studied in recent years.

Chen et al. presented a distributed approach by using game theory to solve the computing offloading problem of multiple users in mobile-edge cloud computing, and the goal is to reach a Nash equilibrium in terms of maximizing the number of users whose tasks are offloaded to the cloud for execution while minimizing the system-wide computation overhead (Chen et al. 2016). Luo et al. studied the computation offloading problem of minimizing the weighted sum of the whole latency and energy consumed by devices in MEC, and presented a new algorithm by applying the reformulation–linearization technique and the branch-and-bound method (Luo et al. 2019). Liu et al. addressed the computing offloading optimization problem with stochastic tasks in MEC from two timescales, and proposed an optimal search algorithm to deal with the problem, aiming to minimize the average delay of each task under power constraint (Liu et al. 2016). Ali et al. developed an efficient algorithm by using deep learning to tackle the computing offloading problem in MEC with the objective of minimizing the weighted sum of energy consumption and service delay (Ali et al. 2021). All above work did not take time constraint into account.

However, it is not reasonable to disregard task deadline for latency-sensitive applications. Therefore, some researches take time constraint into account to study computation offloading problems in MEC. Yi et al. presented an algorithm to solve the problem of multi-user transmission scheduling and computation offloading for delay-sensitive applications in MEC, aiming to maximize the network social welfare and reach a game equilibrium among users under time constraint (Yi et al. 2019). Labidi et al. studied the problem of computation offloading in small cell base stations through jointly optimizing offloading decisions and radio resources to minimize average consumed energy by multiple users with average tolerable delays (Labidi et al. 2015). Guo et al. presented a computation offloading method by using genetic algorithm and particle swarm optimization algorithm to minimize total energy consumption of all user equipments under time constraint in MEC by optimizing computation assignment, channel assignment, offloading decision, and transmission power (Guo et al. 2018). Qiao et al. presented a heuristic algorithm to work out the task offloading and migration problem in MEC, aiming to minimize the average delay of all tasks with time constraint (Qiao et al. 2022). You et al. designed efficient algorithms to solve the multi-user computing offloading and resource allocation problem in an MEC system, aiming to minimize the weighted sum of energy consumed by mobiles within a specified deadline (You et al. 2016). Zhang et al. stated the computing offloading problem in a multi-small cell and muti-user environment integrated with MEC, and presented an artificial fish swarm algorithm to minimize the network offloading energy consumption under time constraint (Zhang et al. 2017). Ding et al. investigated the computing offloading problem in both horizontal and hierarchical end-edge-cloud computing architectures, and proposed two efficient game-based algorithms to solve the problem, where every user end separately minimizes the weight sum of its task completion time and energy consumption under time constraint (Ding et al. 2022). All above work did not take energy constraint into account.

Since the battery capacity of user device is limited, energy constraint needs to be considered when researchers study the computation offloading problem for user devices in MEC. Zhou and Jadoon presented a partial offloading strategy to minimize the computation time of tasks with the constraint of energy in MEC, considering bandwidth resource allocation and the ratio of task offloading (Zhou and Jadoon 2021). Mao et al. designed a low-complexity, dynamic, Lyapunov optimization-based, and online algorithm to minimize the weighted sum of the task delay and the cost of task dropping in an MEC system for energy harvesting devices, via jointly optimizing the CPU-cycle frequencies of mobile devices, the offloading decision, and the transmit power of computation offloading under energy constraint (Mao et al. 2016). Zhou et al. investigated the computation offloading problem in a multi-input multi-output MEC environment with energy harvesting, proposed a dynamic computation offloading method, and aimed to minimize the average weighted sum of energy consumption and latency via jointly optimizing CPU-cycle frequency, transmission covariance matrix, and offloading ratio under time and energy constraints (Zhou et al. 2021). Li first proposed a two-stage technique to tackle two optimal computation offloading problems, where one is to minimize energy consumed under the constraint of time, and the other is to minimize execution time under energy constraint (Li 2021a); and then he took a greedy-based algorithm to attain a computation offloading strategy by considering multiple aspects such as the communication channels and the task characteristics. Li presented a deep reinforcement learning-based computing offloading algorithm to minimize the weight sum of energy consumption and delay in MEC for Internet of Vehicles, considering time and energy constraints (Li 2021b). All above work took energy constraint into account.

Among the aforementioned studies related to computation offloading in MEC, no one considered how to improve the number of completed tasks. If not to improve the number of completed tasks, there will be a considerable tasks that cannot be completed in given time, which will cause bad quality of experience (QoE) of users. A few of work discussed the number of completed tasks in MEC. Han et al. (2022) proposed the energy-efficient service placement algorithm to minimize the total power consumption and maximize the number of accepted services in MEC under time constraint. Tan et al. (2022) proposed a two-stage method based on ant colony system and deep Q-learning to solve the multi-user collaborative task offloading problem in MEC network, with the objective of minimizing the total energy consumed by all user devices in given deadlines through jointly optimizing resource allocation, collaboration decision, and offloading decision, and they also considered the task completion rate.

Table 1 Comparisons of our work and other researches

Work	Time constraint	Energy constraint	Objective
(Chen et al. 2016)	No	No	Maximize the number of offloaded tasks and minimize the system-wide computation overhead
(Luo et al. 2019)	No	No	Minimize the weight sum of energy consumption and latency
(Liu et al. 2016)	No	No	Minimize the average delay
(Ali et al. 2021)	No	No	Minimize the weighted sum of energy consumption and latency
(Yi et al. 2019)	Yes	No	Maximize the network social welfare
(Labidi et al. 2015)	Yes	No	Minimize average energy consumption
(Guo et al. 2018)	Yes	No	Minimize energy consumption
(Qiao et al. 2022)	Yes	No	Minimize the average delay
(You et al. 2016)	Yes	No	Minimize energy consumption
(Zhang et al. 2017)	Yes	No	Minimize the network offloading energy consumption
(Ding et al. 2022)	Yes	No	Minimizes the weight sum of its task completion time and energy consumption
(Zhou and Jadoon 2021)	No	Yes	Minimize latency
(Mao et al. 2016)	No	Yes	Minimize the weighted sum of the task delay and the task dropping cost
(Zhou et al. 2021)	Yes	Yes	Minimize the average weighted sum of energy consumption and latency
(Li 2021a)	Yes	No	Minimize the energy consumption
(Li 2021a)	No	Yes	Minimize latency
(Li 2021b)	Yes	Yes	Minimize the weighted sum of energy consumption and latency
(Han et al. 2022)	Yes	No	Minimize the total power consumption and maximize the number of accepted services
(Tan et al. 2022)	Yes	No	Minimize the total energy consumption
Ours	Yes	Yes	Maximize the weighted sum of the number of completed tasks, average latency, cost and energy consumption

Different from above work, this paper investigates the problem how to maximize the number of completed tasks and minimize the average time, energy and cost of completed tasks in an MEC system by optimizing the CPU frequency of user devices, the offloading location, the offloading ratio and the computing capacity allocated to users by edge servers under time and energy constraints. The problem is a multi-objective optimization problem, and we will change it into a single-objective optimization problem later. Obviously, the problem is complex. Moreover, reducing time conflicts with reducing energy and cost, which make it challenging to explore an efficient algorithm to produce suitable computation offloading decisions and reasonable resource allocations. Table 1 compares our work with other researches.

3 Models and problem definition

This section will introduce the models and problem focused on in this paper.

3.1 System model

This paper considers an edge computing system with M base stations, and each base station is equipped with an edge server. Suppose that there are N user devices (also called users) and they are randomly spread in the common coverage of M base stations. Every user can choose any one of the M base stations as the offloading target. The set of edge servers and user devices are represented as $M_B = \{1, 2, 3, \ldots, M\}$ and $U = \{1, 2, 3, \ldots, N\}$, respectively. Every user device has one task to process. The task that user i needs to process is defined as $task_i = (d_i, T_i^m)$, where d_i is the amount of data owned by the task of user i, and T_i^m is the deadline to finish the task of user i. Suppose that tasks are divisible and there is no dependency between the divided new tasks. Therefore, we can divide the task of each user into two parts, and compute it side by side on the local user device and an edge server. For example, virus scanning applications and image compression applications are divisible tasks (Wang et al. 2016). Figure 1 shows an example of an MEC system.

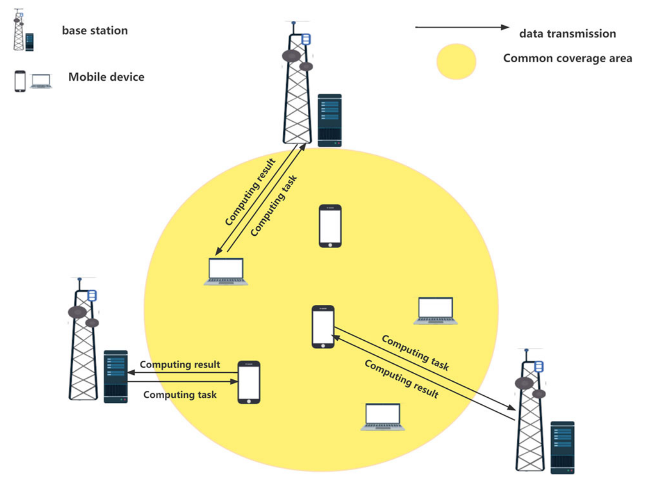


Fig. 1 An example of an MEC system

3.2 Time model

The time required to process a user task is composed of two parts: local computation time and remote computation time. It is the maximum between the local computation time and the remote computing time. The local computation time of the task for user i is calculated by

$$T_i^l = \frac{(1-\alpha_i)\, d_i c_i}{f_i}, \tag{1}$$

where α_i is the offloading percentage of user device i's task, c_i is the CPU clock cycles needed by user device i to compute one bit of data, f_i represents the computing capacity used by user device i to process its task, f_i^c represents the maximum computing capacity of the user device i, and $f_i \leq f_i^c$.

The remote computation time includes the data uplink transmission time, the task computation time on edge server, and the data downlink transmission time. The data downlink transmission time is usually much smaller than the uplink transmission and the task computation time on edge server (Wang et al. 2019), so we do not consider the data downlink transmission time in this paper. Also, we suppose that the communication channels are independent of each other, that is, there is no mutual interference between communication channels. A user device can offload part or all of its task to an edge server for execution.

Let L_i be the task offloading location of user i. $\alpha_i = 1$ and $L_i = j$ mean that the task of user i will be fully offloaded to edge server j for execution, $j \in M_B$. $0 < \alpha_i < 1$ and $L_i = j$ mean that the task of user i will be partially offloaded to edge server j for execution, $j \in M_B$. $\alpha_i = 0$ and $L_i = j$ mean that the task of user i will be executed locally on itself. The transmission time for transferring the task of user device i from itself to the edge server j is computed by

$$T_{i,j}^{tr} = \frac{d_i \alpha_i}{R_{i,j}}, \tag{2}$$

where $R_{i,j}$ is the data rate of wireless transmission between edge server j and user device i, and is calculated as shown in Eq. (3):

$$R_{i,j} = W \log_2\left(1 + \frac{p_i^{tr} h_{i,j}}{\sigma^2}\right), \tag{3}$$

where W is a constant, representing the uplink channel bandwidth. We suppose that the bandwidth is the same for all users devices. p_i^{tr} is the power of transmitting a unit of data from user device i to an edge server. σ^2 is the variance of Gaussian noise. $h_{i,j}$ is the channel gain between user device i and edge server j. $h_{i,j} = d^{-l}$, where d is the distance from edge server j to user device i, and l is the factor of path loss.

The computation time of the task offloaded by user device i to execute on edge server j is

$$T_{i,j}^c = \frac{d_i \alpha_i C_j}{f_{i,j}}, \tag{4}$$

where C_j is the clock cycles needed by edge server j to compute one bit data and $f_{i,j}$ is the computing capability allocated to user i by edge server j. F_j^m is the maximum computing capability of edge server j, and $f_{i,j} \leq F_j^m$. The total time for task offloading includes transmission time and remote computing time. When the local computing has been completed but the offloading computing has not been completed, the user device needs to wait the result returned by edge server. The waiting time of user device i is

$$T_i^w = \begin{cases} 0, & T_{i,j}^{tr} + T_{i,j}^c - T_i^l \leq 0; \\ T_{i,j}^{tr} + T_{i,j}^c - T_i^l, & T_{i,j}^{tr} + T_{i,j}^c - T_i^l > 0. \end{cases} \tag{5}$$

The total time for processing task i is then calculated by

$$T_i = \max\left(T_i^l, T_i^{tr} + T_{i,j}^c\right). \tag{6}$$

3.3 Energy consumption model

Both edge server and user device consume energy when they perform and transmit tasks. Edge server is located nearby the base station and usually has a stable energy source. Therefore, we only consider the energy overhead of user device. It mainly includes local computing energy, transmission energy and waiting energy. The energy overhead consumed by local computing for user device i is

$$E_i^l = \phi\,(1-\alpha_i)\, d_i c_i f_i^2, \tag{7}$$

where ϕ is the capacitance of effective switching, determined by the chip architecture (Wang et al. 2016). Transmission energy overhead of user device i is shown in Eq. (8):

$$E_i^{tr} = T_i^{tr} p_i^{tr}. \tag{8}$$

The energy consumption of waiting result cannot be ignored for energy-poor user device. The waiting energy overhead of user device i is

$$E_i^w = T_i^w p_i^w, \tag{9}$$

where p_i^w is waiting power of user device i.

The total energy overhead for user device i processing its task is shown in Eq. (10):

$$E_i = E_i^{tr} + E_i^w + E_i^l. \tag{10}$$

3.4 Cost model

Offloading the task to edge server for execution will consume the monetary cost of users. The cost is mainly composed of the data transmission cost of the task for user device and the computing cost of the task on edge server. As Ref. (Wang et al. 2020a), the transmission cost of user i is calculated by

$$C_i^{tx} = d_i \alpha_i c_{tx}, \tag{11}$$

where c_{tx} is the cost of transmitting one bit of data.

As Ref. (Wang et al. 2020a), the computing cost to execute the task of user i on edge server j is

$$C_i^c = f_{i,j} c_c, \tag{12}$$

where c_c is the fee charged for a unit of computing resource and it is determined by the occupied CPU frequency of edge server j. Then, the total cost of offloading the task of user i to edge server j for execution is

$$C_i = C_i^{tr} + C_i^c. \tag{13}$$

3.5 Number model

Any task that does not satisfy time or energy constraints is regarded as uncompleted task. The expression of whether the task of a user device i can be completed or not is shown in Eq. (14):

$$True_i = \begin{cases} 0, & E_i > E_i^m \text{ or } T_i > T_i^m; \\ 1, & E_i \le E_i^m \text{ and } T_i \le T_i^m; \end{cases} \tag{14}$$

where E_i^m is the maximum available energy of user device i. $True_i = 0$ means that the task of user device i cannot be completed due to exceeding the energy limit or the time limit; $True_i = 1$ means that the task can be completed in given energy and time constraints. Therefore, the number of completed tasks in a given MEC system is computed by

$$num = \sum_{i=1}^{N} True_i. \tag{15}$$

We do not consider the time, energy and cost of tasks that are unable to be completed after offloading decisions and resource allocation are made. Tasks that are not to be completed will be discarded before offloading. They will be recalled in next time when an offloading decision is generated or the user device is fully charged, which will not be considered here. We only consider the time, energy, and cost consumed by tasks that can be completed.

The average completion time of tasks that can be completed is calculated by

$$T = \frac{\sum_{i=1}^{N} (T_i True_i)}{num}. \tag{16}$$

The average cost of tasks that can be completed is calculated by

$$C = \frac{\sum_{i=1}^{N} (C_i True_i)}{num}. \tag{17}$$

The average energy overhead of tasks that can be completed is calculated by

$$E = \frac{\sum_{i=1}^{N} (E_i True_i)}{num}. \tag{18}$$

3.6 Problem description

The studied problem is as follows: Given above addressed MEC system with M edge servers and N user devices, the goal is to maximize the number of completed tasks and minimize the average energy overhead, time, and cost of completed tasks under time and energy constraints, considering the offloading location, offloading rate, local computation frequency, and remote allocation of resources. Obviously, this is a multi-objective optimization problem. To facilitate the solution, we convert the problem into a single-objective optimization problem. Let $L = \{L_1, L_2, ..., L_N\}$, $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_N\}$, and $f = \{f_1, f_2, ..., f_N\}$, $f_{IJ} = \{f_{11}, f_{12}, ..., f_{1M}, ..., f_{N1}, f_{N2}, ..., f_{NM}\}$, and the new single-objective problem can be mathematically expressed as follows:

$$P: \max_{L,\alpha,f,f_{IJ}} \{num - (\omega T + \gamma C + \zeta E)\}, \tag{19}$$

$$\begin{aligned}
&C1: \sum_{1 \le j \le M} (L_i = j) \le 1, \forall i \in U,\\
&C2: \alpha_i \in [0, 1], \forall i \in U,\\
&C3: 0 \le f_i \le f_i^c, \forall i \in U,\\
&C4: 0 \le f_{i,j} \le F_j^m, \forall i \in U,\\
&C5: \sum_{i \in U} f_{i,j} True_i \le F_j^m, \forall j \in M_B,
\end{aligned}$$

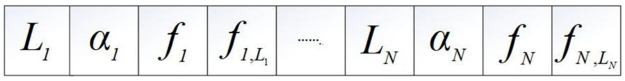
ω, α, ξ, f_i^c, and F_j^m are the weighting factors of time, energy overhead, cost, the maximum frequency of user device i, and the maximum frequency of edge server j, respectively. Constraint $C1$ indicates that the task of user device i can be executed locally on itself or partially offloaded to at most one edge server for execution. $C2$ indicates the range of offloading ratio for the task of user device i. Constraints $C3$ indicates that the computing capacity of user device i cannot exceed the maximum computing capacity of device i. Constraints $C4$ indicates that the computing capacity allocated to the task of user device i cannot exceed the maximum computing capacity of edge server j. $C5$ shows that the computing capacity assigned by edge server j cannot exceed its maximum computing capacity, and here only achievable tasks need to be considered.

It is observed that P is a mix integer nonlinear programming problem of high dimension. Researchers have proved that offloading problem in MEC is an NP-hard problem. With the number of user devices increasing, P will become more and more complex. Thence, an efficient algorithm is necessary.

4 The proposed technique

In this paper, we design a novel technique, the mayfly genetic algorithm (MGA), which combines the advantages of the mayfly method and the genetic method to cope with our studied problem.

The mayfly method was first proposed by Zervoudakis and Tsafarakis in 2020 (Zervoudakis and Tsafarakis 2020). It is a new simulation optimization algorithm inspired by the flight behavior and mating process of mayflies, combining the main advantages of evolutionary algorithms and population intelligence. The mayfly method has the characteristics of fast speed and high accuracy of convergence. However, it is easy to fall into the local optimum solution, and especially in high-dimensional problems, the global search ability is poor. Since the problem to be solved in this paper is a high-dimensional optimization problem, the mayfly algorithm cannot solve this problem very well. The genetic algorithm is a global search optimization algorithm that mimics the natural evolutionary process based on the reproduction and survival of the fittest (Goldberg 2010). Although the genetic algorithm has strong global search ability, its local search ability is poor and its convergence speed is slow. Clearly, neither the mayfly method nor the genetic method can solve our problem well. However, we can combine the advantages of the two methods-fast speed and high accuracy of convergence of the mayfly method plus the strong global search ability of the genetic method to explore an efficient method to tackle our studied problem. We will introduce our method in detail below.



L_1	α_1	f_1	f_{1,L_1}	......	L_N	α_N	f_N	f_{N,L_N}

Fig. 2 A mayfly

4.1 Constructions of the mayfly individual and fitness function

In our algorithm, there are two kinds of mayflies: male population and female population. Every mayfly is randomly generated as a candidate solution at first, and represented by a $4 \times N$-dimensional vector $x = (x_1, ..., x_{4N})$, indicating the position of a mayfly.

Figure 2 shows an example of a mayfly. The mayfly consists of four components: the offloading locations of all user tasks, the offloading ratios of all user tasks, the CPU frequencies of users to execute their tasks, and the computing capacity allocated by an edge server to execute a task that is offloaded to the edge server.

We use the fitness function f to assess the quality of a mayfly's position, and it is computed by Eq. (20),

$$f = \{num - (\omega T + \gamma C + \xi E)\} - penalty, \tag{20}$$

where $penalty$ represents the penalty and is calculated according to Eq. (21),

$$\begin{aligned}
penalty = {} & p_0(N - num)\\
& + p_1\left(\sum_{i \in U} \max\left(0, \sum_{1 \le j \le M} (L_i = j) - 1\right)\right)\\
& + p_2\left(\sum_{i \in U} \max(0, \alpha_i - 1)\right)\\
& + p_3\left(\sum_{i \in U} \max\left(0, f_i - f_i^c\right)\right)
\end{aligned}$$

$$+p_4\left(\sum_{j\in M_B}\max\left(0,\sum_{i\in U}f_{i,j}\,\mathrm{True}_i-F_j\right)\right)$$
$$+p_5\left(\sum_{i\in U}\max\left(0,f_{i,j}-F_j\right)\right), \tag{21}$$

where p_0 is the penalty factor for uncompleted tasks, and p_1-p_5 are the penalty factors that do not satisfy constraints $C1$-$C5$.

Also, each mayfly has a velocity $v = (v_1, ..., v_{4N})$. According to the velocity, a mayfly can change its position. In the following, we will introduce how male and female mayflies change their positions.

Essentially, the process of computing a solution is the process of particles updating their positions and velocities.

4.2 Movement of male mayflies

Male mayflies prefer to gather in swarms and will change their positions based on both their own and their neighbors' experiences. Denote x_i^{t+1} and x_i^t separately to be the positions of male mayfly i at steps $t+1$ and t, and then the position of male mayfly i is updated by

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \tag{22}$$

where the vector v_i^{t+1} is the velocity of male mayfly i at step $t+1$. Denote x_{ij}^{t+1} and v_{ij}^{t+1} to be the components of x_i^{t+1} and v_i^{t+1} on dimension j, respectively. Then, we have

$$\begin{aligned} v_{ij}^{t+1} =& \eta \times v_{ij}^t + \mu_1 e^{-\delta\epsilon_p^2}\left(pb_{ij} - x_{ij}^t\right)\\ &+ \mu_2 e^{-\delta\epsilon_g^2}\left(gb_j - x_{ij}^t\right). \end{aligned} \tag{23}$$

η is the gravity coefficient, making a balance between the exploitation and exploration, and calculated by

$$\eta = \frac{\exp(2(1-t/\kappa)) - \exp(-2(1-t/\kappa))}{\exp(2(1-t/\kappa)) + \exp(-2(1-t/\kappa))}, \tag{24}$$

where κ and t are the maximum and the current iteration number, respectively. μ_1 and μ_2 are positive attraction constants that measure the cognitive and social aspect contributions, respectively. δ is the visibility coefficient, and controls the visibility range of mayflies. pb_i is a vector, representing the best position reached by male mayfly i, and pb_{ij} is its j^{th} component. We have

$$pb_i = \begin{cases} x_i^{t+1}, \text{if } f\left(x_i^{t+1}\right) > f\left(pb_i\right); \\ pb_i, \text{otherwise.} \end{cases} \tag{25}$$

The larger the fitness function value corresponding to a mayfly position, the better the position. gb_j denotes the component on dimension j of the vector gb which is the best position reached by both male and female mayflies. ϵ_p denotes the cartesian distance between vectors x_i^t and pb_i. ϵ_g is the cartesian distance between vectors x_i^t and gb. The cartesian distance between two n-dimensional vectors $x = (x_1, x_2, ..., x_n)$ and $\chi = (\chi_1, \chi_2, ..., \chi_n)$ is calculated as Eq. (26)

$$\|x - \chi\| = \sqrt{\sum_{i=1}^{n}(x_i - \chi_i)^2}. \tag{26}$$

To make our algorithm work better, the best male mayflies who have the largest fitness function value should continue their characteristic up-and-down matrimonial dance. Their velocities are computed by

$$v_{ij}^{t+1} = \eta \times v_{ij}^t + d \times r, \tag{27}$$

where d is the matrimonial dance factor, and r is a random constant belonging to the closed interval $[-1, 1]$.

The detail of how male mayflies move (that is, change their positions) is shown in Algorithm 1.

Algorithm 1 Movement of male mayfies

```
Require: $\delta, \eta, pb, gb, NP, \mu_1, \mu_2, d, x_i (i = 1, 2, ..., NP), v_i (i = 1, 2, ..., NP)$.
Ensure: $x_i (i = 1, 2, ..., NP)$
1: for $i = 1$ to $NP$ do
2:     Calculate $\epsilon_p, \epsilon_g$ by (26)
3:     if $f(x_i) < f(gb)$ then
4:         Update $v_i$ by (23)
5:     else
6:         Update $v_i$ by (27)
7:     end if
8:     Update $x_i, pb_i$ by (22), (25)
9:     Update global optimal location $gb$
10: end for
```

4.3 Movement of female mayflies

In order to breed, female mayflies should fly toward male mayflies. Their positions are adjusted by

$$y_i^{t+1} = y_i^t + v_i^{t+1}, \tag{28}$$

where y_i^{t+1} and y_i^t are the positions of female mayfly i at steps $t+1$ and t, respectively. v_i^{t+1} denotes the velocity of female mayfly i at step $t+1$.

To improve the global search performance, we make some changes to the traditional attraction method. The changes are as follows: The best female mayfly (having the largest fitness function value among female mayflies) is attracted by the best male mayfly (having the largest fitness function value among male mayflies), the second best female mayfly (having the second largest fitness function value among female mayflies) is attracted by the second best male mayfly (having the second largest fitness function value among male mayflies), and so on. If the fitness function value of the best female mayfly is smaller than that of the best male mayfly, the best female mayfly will not move toward the current position of the best male mayfly, but move toward the historical optimal position of the best male mayfly. At the same time, the best female mayfly has a certain probability to move toward the global optimal position of mayflies. If the fitness function tion value of the second best female mayfly is smaller than that of the second best male mayfly, the second best female mayfly will not move toward the current position of the second best male mayfly, but move toward the historical optimal position of the second best male mayfly. At the same time, the second best female mayfly has a certain probability to move to the global optimal position of mayflies, and so on. The velocities of female mayflies are calculated by

$$v_{ij}^{t+1} = \begin{cases} \eta \times v_{ij}^{t} + \mu_2 e^{-\delta\epsilon_p^2}\left(pb_{ij} - y_{ij}^{t}\right), \text{ if } p < c_1; \\ \eta \times v_{ij}^{t} + \mu_2 e^{-\delta\epsilon_g^2}\left(gb_j - y_{ij}^{t}\right), otherwise. \end{cases} \tag{29}$$

v_{ij}^{t+1} and v_{ij}^{t} are the velocities of female mayfly i in dimension j at steps $t+1$ and t, respectively. y_{ij}^{t} denotes the position of female mayfly i on dimension j at step t. p is a random value and $p \in [0, 1]$. c_1 is a constant and $c_1 \in [0, 1]$.

Algorithm 2 Movement of female mayflies

```
Require: $\delta$, $NP$, $\eta$, $\mu_1$, $\mu_2$, $fl$, $y_i(i = 1, 2, ..., NP)$, $x_i(i = 1, 2, ..., NP)$, $v_i(i = 1, 2, ..., NP)$.
Ensure: $y_i(i = 1, 2, ..., NP)$
1: for $i = 1$ to $NP$ do
2:     Calculate $\epsilon_p, \epsilon_g$ according to (26)
3:     if $f(y_i) < f(x_i)$ then
4:         Update $v_i$ by (29)
5:     else
6:         Update $v_i$ by (30)
7:     end if
8:     Update $y_i$ by (28)
9:     if $\min\{f(pb_1), f(pb_2), \ldots f(pb_{NP})\} \le f(y_i) < f(gb)$ then
10:        Update $y_i$ by (31)
11:    else if $f(y_i) < \min\{f(pb_1), f(pb_2), \ldots f(pb_{NP})\}$ then
12:        Update $y_i$ by (32)
13:    else
14:        Update $gb$ by (33)
15:    end if
16: end for
```

If the fitness function value of the best female mayfly is larger than or equal to that of the best male mayfly, the best female mayfly will not move toward to the current position of the best male mayfly, but move toward a randomly generated position. It is true for the second best female mayfly, the third best female mayfly, and so on. The velocities of female mayflies will be updated by

$$v_{ij}^{t+1} = \eta \times v_{ij}^{t} + fl \times r, \tag{30}$$

Algorithm 3 The operations to generate offsprings

```
Require: $p_c$, $p_m$, $y_i(i = 1, 2, ..., NP)$.
Ensure: $y_i(i = 1, 2, ..., NP)$
1: Select the population using the tournament method
2: for $i = 1$:$NP$:2 do
3:     $p = random(0, 1)$
4:     if $p < p_c$ then
5:         Randomly select cross-segments
6:         Cross the corresponding segments of $y_i$ and $y_{i+1}$
7:     end if
8: end for
9: for $i = 1$:$NP$ do
10:    $p$=random(0,1)
11:    if $p < p_m$ then
12:        Randomly select a location for mutation
13:    end if
14: end for
```

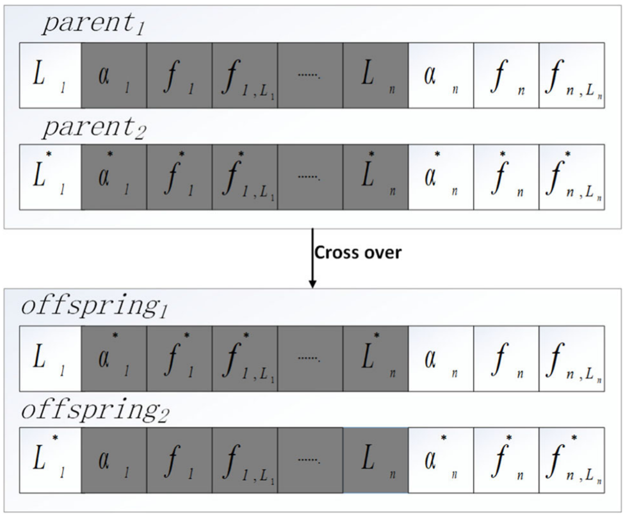


Fig. 3 The crossover of female mayflies

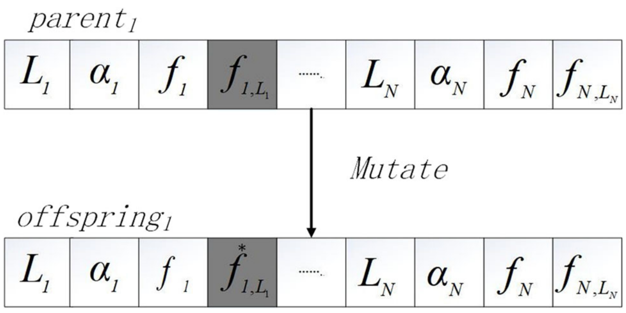


Fig. 4 The mutation of female mayflies

Table 2 Experimental parameters

Parameters	Value range
The maximum frequency of user device i: f_i^c	1.0−1.5Ghz
CPU cycles required by user devices to calculate 1 bit data: c_i	200−400
User device transmission power: p_i^{tr}	0.25−1.0W
User device waiting power: p_i^w	0.05−0.2W
Available energy for user devices: E_i^m	3−30 joules
The maximum computing capacity of edge servers: F_j^m	75−100Ghz
CPU cycles required for edge server to process 1 bit data: C_j	50−100
Energy consumption factor: ϕ	1e−26
Algorithm parameters: $\delta, \mu_1, \mu_2, d, p_m, p_c, fl, c_1, c_2$	0.1/1.2/1.0/0.05/0.6/0.05/0.1/0.9/0.4

where fl is a random wandering coefficient, and r is a random constant belonging to the closed interval $[-1, 1]$.

If the fitness function value of a female mayfly is greater than or equal to $\min\{f(pb_1), f(pb_2), ..., f(pb_{NP})\}$ and less than $f(gb)$ after the female mayfly moves to a new position, the position of the female mayfly will be updated by

$$y_i^{t+1} = \begin{cases} gb, & \text{if } p < c_2; \\ pb_r, & \text{otherwise}. \end{cases} \tag{31}$$

pb_r is the historical optimal position of male mayfly r, where r is an integer and randomly generated from the interval $[1, NP]$. c_2 is a constant and $c_2 \in [0, 1]$. After the position is updated, the velocity of the female mayfly is initialized to be 0.

If the fitness function value of a female mayfly is less than or equal to $\min\{f(pb_1), f(pb_2), ..., f(pb_{NP})\}$, we update the position of the female mayfly by

$$y_i^{t+1} = gb. \tag{32}$$

After the position is updated, the velocity of the female mayfly is initialized to be 0.

If the position of female mayfly i is better than the global historical optimal position, the global optimal historical position is replaced with the current position of the female mayfly i. That is,

$$gb = y_i^{t+1}. \tag{33}$$

The movement of female mayflies is detailedly described in Algorithm 2.

Inspired by the genetic algorithm, we use the selection, crossover and mutation operations to generate offsprings. Algorithm 3 shows the details of the three operations to produce the offsprings of female mayflies. First, the algorithm uses the tournament method to create the female mayfly population. The tournament method first chooses a certain number of female mayflies from the female mayfly population each time (put-back sampling), and then select the best one which has the largest fitness function value to enter the offspring population; repeat this operation until the offspring population size reaches the original population size. Then, the algorithm randomly takes two parents from offspring population to cross over at the probability of p_c to generate two new offsprings. The crossover operation is shown in Fig. 3. Crossover operation requires fragment correspondence to ensure that the variables of new offsprings remain in the feasible region after crossover. Finally, the algorithm mutates the newly generated offsprings. For each female mayfly, the location for mutation is randomly selected at the probability of p_m. Also, the variables of the offsprings by mutation cannot exceed the feasible region. The mutation operation is shown in Fig. 4.

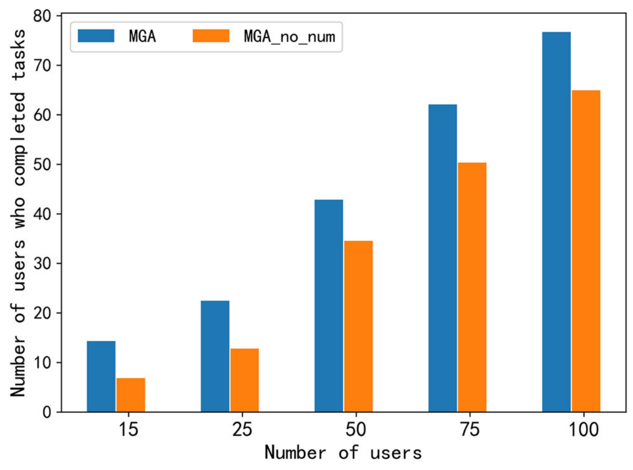
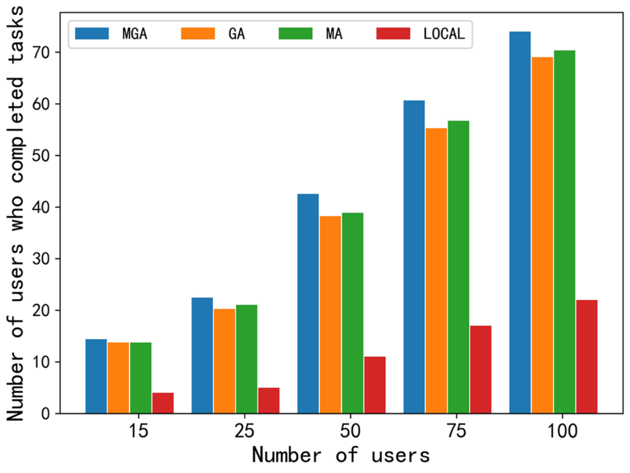
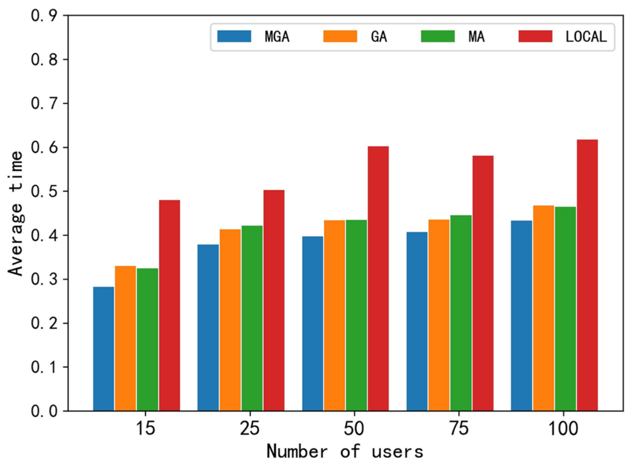


Fig. 5 Comparison of the number of completed tasks when the number of users changes for algorithms MGA and MGA_no_num

Algorithm 4 introduces the complete flow of MGA. First, the populations of males and females, the velocities of mayflies, current iteration t, pb and gb are initialized. Then, a while loop is used to find the solution. First, it updates the gravity weight factor η; second, it updates the male population by Algorithm 1 and sort the mayflies in the female population and the mayflies in the male populations accord-

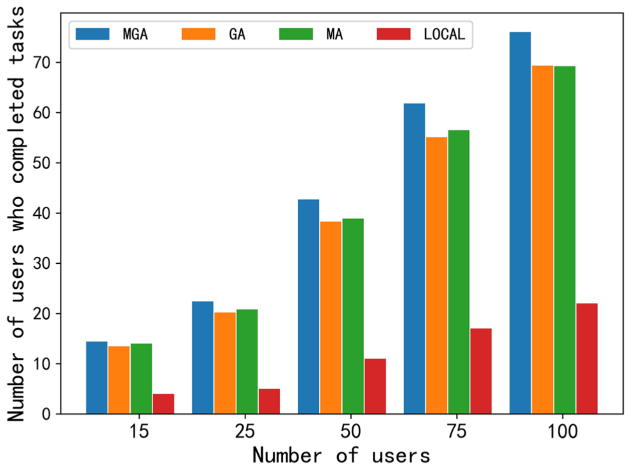


(a)

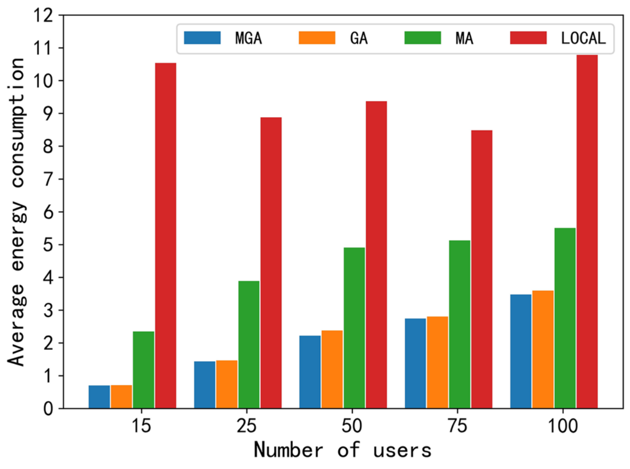


(b)

Fig. 6 (**a**) and (**b**) show that the number of completed tasks and the average completion time of the task of each user under different number of users only when the number of users changes and the average completion time is considered



(a)



(b)

Fig. 7 (**a**) and (**b**) show that the number of completed tasks and the average energy consumption of the task of each user under different number of users only when average energy consumption is considered

ing to their fitness function values, respectively; third, it updates the female population by calling Algorithms 2 and 3; fourth, it increases t by one; fifth, repeats these operations until the iteration number reaches the maximum threshold; finally, it computes the number of completed tasks, the average time, cost, and energy consumption of completed tasks. Note that during each iteration, the positions and velocities cannot exceed their limited ranges.

Algorithm 4 MGA

```
Require: The size of the population $NP$, number of iterations $\kappa$, probability of crossover $p_c$, probability of variation $p_m$, the visibility coefficient $\delta$, the gravity coefficient $\eta$, the positive attraction constants $\mu_1$, $\mu_2$, random wandering coefficient $fl$, the dance factor $d$.
Ensure: $gb$, $Num$, $T$, $C$ and $E$
1: Initialize male and female populations $x_i (i = 1, 2, ..., NP)$, $y_i (i = 1, 2, ..., NP)$, the velocities of male and female populations, $gb$ and $t = 0$.
2: while $t < \kappa$ do
3:     Update $\eta$ by (24)
4:     Perform Algorithm 1
5:     Separately sort male mayflies and female mayflies in descending order according to the fitness function value
6:     Perform Algorithm 2
7:     Perform Algorithm 3
8:     $t = t + 1$
9: end while
10: Compute $Num$, $T$, $C$ and $E$ by Eqs. (15)-(18).
```

4.4 Time complexity analysis

In this section, we shows how to compute the time complexity of our proposed MGA algorithm. When initializing the population, the time complexity depends on the number of user devices and the number of male mayflies in the population. The time complexity of initializing the mayfly

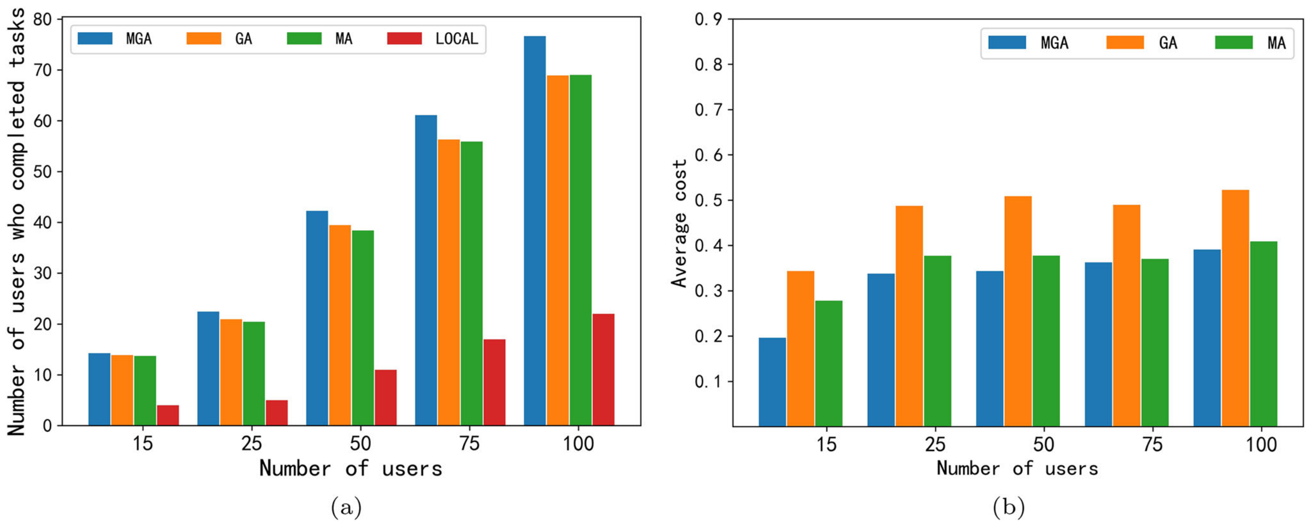


Fig. 8 (**a**) and (**b**) show that the number of completed tasks and the average cost of the task of each user under different number of users only when average cost is considered

population is $O(NP \times N \times 4)$. The time complexity of calculating the fitness function value is $O(N)$. The selection operation has the time complexity of $O(NP \times N_s)$, where N_s is the number of mayfies for one round of tournament. The time complexity of crossover and mutation operations are $O(NP \times N_c)$, where N_c is the segment length of crossing. Hence, the time complexity of Algorithm 1 and Algorithm 2 are $O(NP \times N \times 4)$, respectively; the time complexity of Algorithm 3 is $O(NP \times N_s) + O(NP \times N_c) + O(NP) \approx O(NP \times N_s)$. Consequently, the time complexity of MGA algorithm is $O(NP \times N \times 4) + O(\kappa \times NP \times N \times 4) + O(\kappa \times NP \times N \times 4) + O(\kappa \times NP \times N_s) \approx O(\kappa \times NP \times N)$, where κ is the overall iteration number.

5 Simulation experiments

We will test the performance of the proposed MGA algorithm in this section.

5.1 Experimental parameter setting

After investing many related references to our problem and algorithm, such as (Guo et al. 2018; Chen et al. 2016; You et al. 2016; Zhou et al. 2021; Zervoudakis and Tsafarakis 2020; Goldberg 2010; Li 2021a; Ding et al. 2022), we set the values of our experimental parameters as follows. Assume that the targeted MEC system includes five base stations, users are randomly spread within 500 ms from the base stations, the bandwidth of the wireless channel is $2Mhz$, and the ambient noise $\sigma^2 = -70$dbm. The values of parameters related to user devices, severs and the MGA algorithm are shown in Table 2. We use four schemes as baselines: the MGA algorithm not considering the number of users whose task can be completed, denoted as MGA_no_num, the genetic algorithm (GA), the mayfly algorithm (MA), and the only local computing denoted by LOCAL. To guarantee the accuracy of experiments, we repeated all the experiments 100 times to take the average results.

5.2 Simulation results and analysis

Figure 5 shows the number of completed tasks by MGA and MGA_no_num. The MGA_no_num scheme that does not take into account the number of users who can complete their tasks only considers the average energy overhead, completion time, and cost of users. Since each user has one task to be processed, the number of users whose tasks have been finished is equal to the number of completed tasks. It is observed that MGA has a larger number of completed tasks than MGA_no_num. The reason is that MGA makes an optimization to the number of completed tasks under given time and energy constraints but MGA_no_num does not do.

Figure 6 shows that the number of completed tasks and the average completion time of the task of each user device with number of users changing only when the average completion time is considered in the objective, that is, $\omega = 1$, $\gamma = 0$, and $\zeta = 0$, by algorithms MGA, GA, MA, and LOCAL. It is observed that MGA can achieve larger number of completed tasks and lower average completion time than other three baselines.

Figure 7 shows the number of completed tasks and the average energy overhead of the task of each user device with number of users changing only when the average energy over-

Table 3 Comparisons of the number of completed tasks under different parameter values

Parameter values	15 user devices	25 user devices	50 user devices	75 user devices	100 user devices
$\omega = 0.6, \gamma = 0.3, \zeta = 0.1$	14.31	22.17	42.85	61.49	76.17
$\omega = 0.6, \gamma = 0.1, \zeta = 0.3$	14.34	22.44	42.87	62.07	76.71
$\omega = 0.6, \gamma = 0.2, \zeta = 0.2$	14.26	22.41	42.23	61.00	76.60
$\omega = 0.3, \gamma = 0.6, \zeta = 0.1$	14.20	22.09	42.69	60.97	76.29
$\omega = 0.1, \gamma = 0.6, \zeta = 0.3$	14.23	22.37	42.46	61.86	76.51
$\omega = 0.2, \gamma = 0.6, \zeta = 0.2$	14.09	22.43	42.51	61.20	76.63
$\omega = 0.1, \gamma = 0.3, \zeta = 0.6$	14.17	22.46	42.17	60.63	76.54
$\omega = 0.2, \gamma = 0.2, \zeta = 0.6$	14.14	22.40	42.60	61.69	75.54
$\omega = 0.3, \gamma = 0.1, \zeta = 0.6$	14.33	22.41	42.83	61.23	76.20
$\omega = 0.3, \gamma = 0.3, \zeta = 0.3$	14.17	22.29	42.77	61.71	76.17

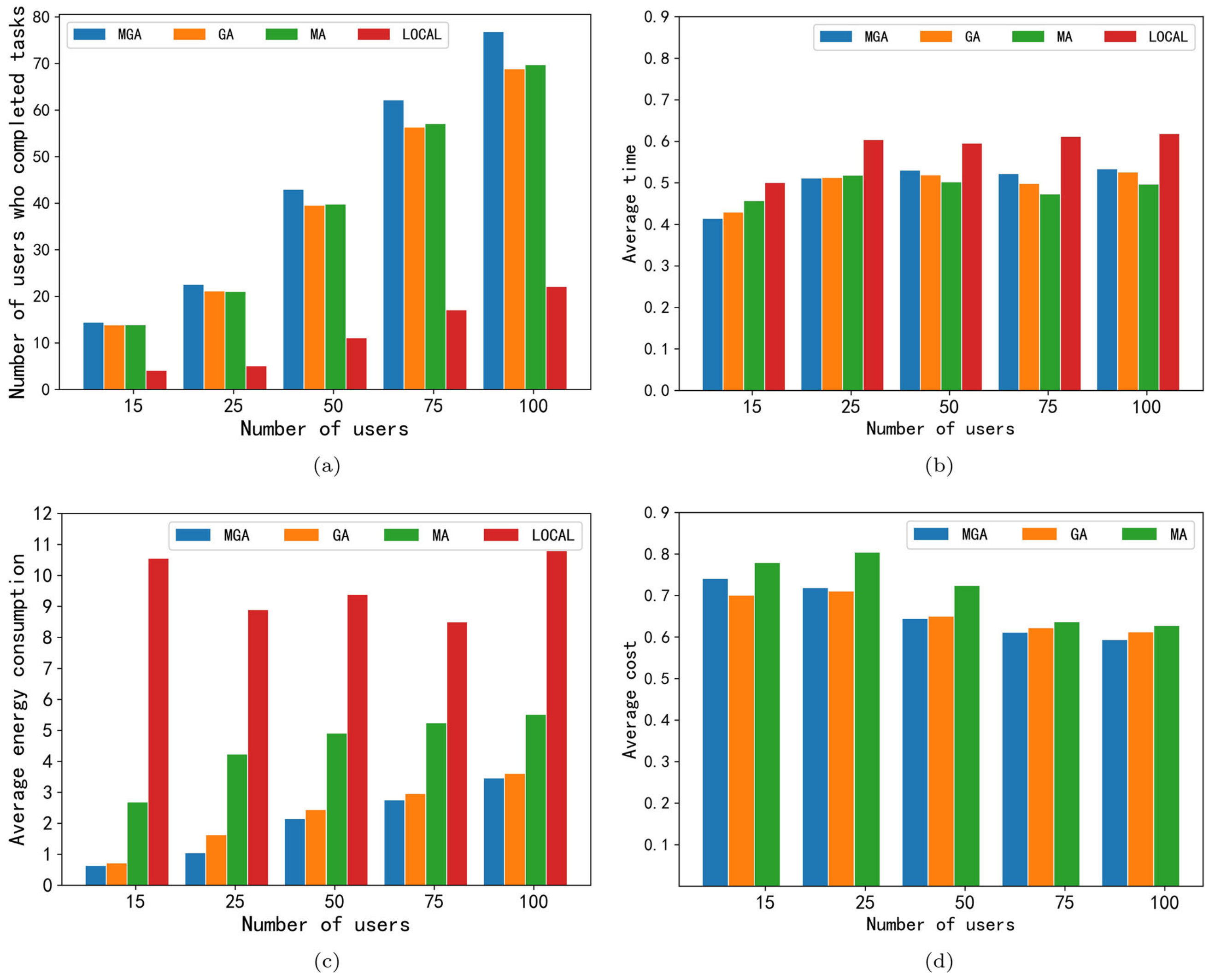


Fig. 9 (**a**), (**b**), (**c**) and (**d**) show that the number of completed tasks, the average completion time, the average energy consumption and the average cost of the task of each user under different number of users when average time, average energy consumption and average cost are considered together

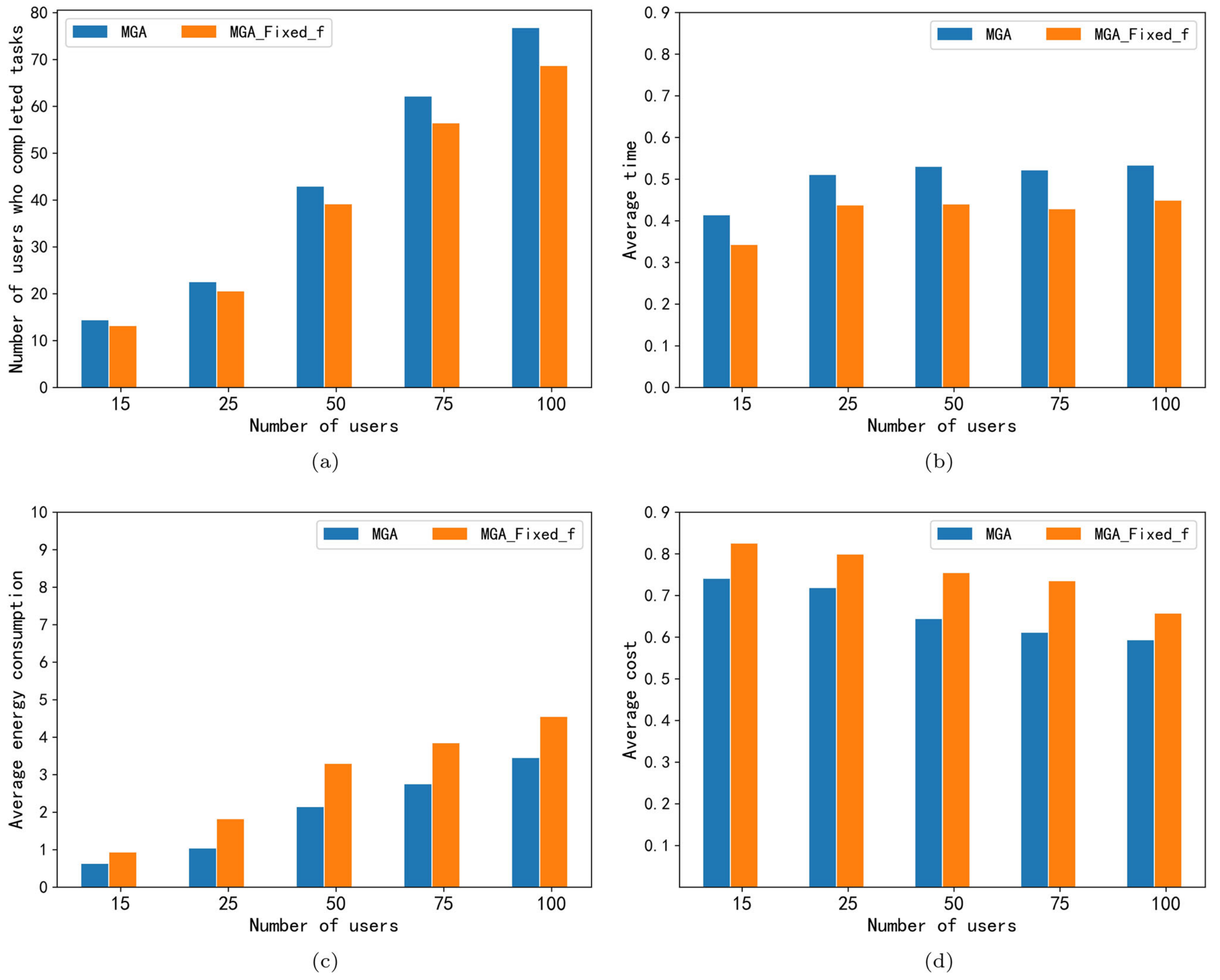


Fig. 10 (**a**), (**b**), (**c**) and (**d**) show that the number of completed tasks, the average completion time, the average energy consumption and the average cost of the task of each user under different number of users at fixed and dynamic frequencies

head is considered in the objective, that is, $\omega = 0$, $\gamma = 0$, and $\zeta = 1$, by algorithms MGA, GA, MA, and LOCAL. It is observed that MGA can achieve larger number of completed tasks and lower average energy overhead of completed tasks than other three baselines.

Figure 8 shows the number of completed tasks and the average cost of the task of each user device with number of users changing only when the average cost is considered in the objective, that is, $\omega = 0$, $\gamma = 1$, and $\zeta = 0$, by algorithms MGA, GA, MA, and LOCAL. It is observed that MGA can achieve larger number of completed tasks and lower average cost of completed tasks than other baselines. Since cost happens only in data transmission and remote computing on edge servers, we do not consider the cost of local computing, and Fig. 8b does not show the average cost achieved by the LOCAL baseline.

When we comprehensively consider the three parts of time, energy consumption and cost, how to choose the appropriate weight factor for each part is an important problem. Therefore, we conduct experiments to compare the number of completed tasks under different values of each weight factor. The experimental results are seen in Table 3. It is observed that when $\omega = 0.6$, $\gamma = 0.1$, and $\zeta = 0.3$, the number of completed tasks is the largest. Thus, when simultaneously optimizing the average time, cost and energy consumption, we set their weight factor values to be $\omega = 0.6$, $\gamma = 0.1$, and $\zeta = 0.3$, respectively.

Figure 9 shows that when simultaneously optimizing the average time, cost, and energy consumption, MGA is still superior to other three baselines GA, MA, and LOCAL in the number of completed tasks, as well as the average energy consumption of completed tasks. Since cost happens only in data transmission and remote computing on edge servers, we do not consider the cost of local computing, and Fig. 9b does not show the average cost achieved by the LOCAL baseline. Although MGA is slightly higher in average cost than other baselines when the number of user device is small, it achieves lower average cost than other baselines with the increasing of the number of user devices. Due to the larger number of completed tasks, lower average energy overhead and lower average cost, MGA is slightly higher than GA and MA in the average time when the number of user devices increases. On the whole, our proposed MGA outperforms the three baselines.

Figure 10 compares the average time and energy consumption of tasks completed between the dynamic local calculation frequency denoted by MGA and the fixed local calculation frequency denoted by MGA_Fixed_f. If the local calculation frequency is fixed, all tasks are performed at the maximum CPU frequency of user devices. It is observed that MGA is better than MGA_Fixed_f in the number of completed tasks, the average cost and the average energy overhead; the average time of completed tasks obtained by MGA is larger than that of MGA_Fixed_f. On the whole, MGA outperforms MGA_Fixed_f.

6 Conclusion

This paper studies the task offloading problem with time and energy constraints in an MEC system. The goal is to maximize the number of completed tasks as well as minimize the average time, cost and energy overhead of achievable tasks, by jointly optimizing task offloading locations, offloading ratios, the CPU frequencies of user devices and computing resources allotted to the task of user devices by edge servers. We propose a new algorithm to solve the problem. Simulation results indicate the effectiveness of our proposed scheme. In the future work, we will consider that a user device has multiple tasks to execute, and the interdependence between tasks.

Acknowledgements This work has been partially supported by the Key Project of Scientific Research Plan of Hubei Provincial Department of Education (Grant No. D20201102).

Author contributions All authors contributed to the study conception and design.

Funding The authors have not disclosed any funding.

Availability of data and material The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have not disclosed any conflict of interests.

References

Aazam M, Zeadally S, Harras KA (2018) Offloading in fog computing for Iot: review, enabling technologies, and research opportunities. Future Gener Comput Syst 87:278–289
Abbas N, Zhang Y, Taherkordi A et al (2018) Mobile edge computing: a survey. IEEE Internet Things J 5(1):450–465
Ali Z, Abbas ZH, Abbas G et al (2021) Smart computational offloading for mobile edge computing in next-generation internet of things networks. Comput Netw 198(108):356
Chen C, Li K, Teo SG et al (2020) Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks. ACM Trans Knowl Discov Data (TKDD) 14(4):1–23
Chen X, Jiao L, Li W et al (2016) Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Netw 24(5):2795–2808
Ding Y, Li K, Liu C et al (2022) A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing. IEEE Trans Parallel Distrib Syst 33:1503–1519
Goldberg DE (2010) Genetic algorithms in search, optimization, and machine learning. Queen's University Belfast, UK
Guo F, Zhang H, Hong J et al (2018) An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. IEEE/ACM Trans Netw 26(6):2651–2664
Guo M, Li Q, Peng Z et al (2022) Energy harvesting computation offloading game towards minimizing delay for mobile edge computing. Comput Netw 204(108):678
Han P, Liu Y, Zhang X, et al (2022) Energy-efficient service placement based on equivalent bandwidth in cell zooming enabled mobile edge cloud networks. IEEE Transactions on Vehicular Technology 71(11):12,275–12,290
Jahandar S, Kouhalvandi L, Shayea I et al (2022) Mobility-aware offloading decision for multi-access edge computing in 5g networks. Sensors 22(7):2692
Kubade HM, Pallavi M, Chaudhari, et al (2018) An overview of cloud computing. SSRN Electronic Journal 4(3):558–560
Labidi W, Sarkiss M, Kamoun MA (2015) Joint multi-user resource scheduling and computation offloading in small cell networks. 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob) pp 794–801
Li K (2021) Heuristic computation offloading algorithms for mobile users in fog computing. ACM Transactions on Embedded Computing Systems 20(2):1–28
Li X (2021) A computing offloading resource allocation scheme using deep reinforcement learning in mobile edge computing systems. J Grid Comput 19:35
Liu J, Mao Y, Zhang J, et al (2016) Delay-optimal computation task scheduling for mobile-edge computing systems. 2016 IEEE International Symposium on Information Theory (ISIT) pp 1451–1455
Luan TH, Gao L, Li Z, et al (2015) Fog computing: Focusing on mobile users at the edge. arXiv:1502.01815
Luo J, Deng X, Zhang H et al (2019) Qoe-driven computation offloading for edge computing. J Syst Architect 97:34–39
Mao Y, Zhang J, Letaief KB (2016) Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J Sel Areas Commun 34(12):3590–3605
Mao Y, You C, Zhang J, et al (2017) A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys and Tutorials PP(99):1–1
Qi W, Sun H, Yu L, et al (2022) Task offloading strategy based on mobile edge computing in uav network. Entropy 24

Qiao B, Liu C, Liu J, et al (2022) Task migration computation offloading with low delay for mobile edge computing in vehicular networks. Concurrency and Computation: Practice and Experience 34

Saeik F, Avgeris M, Spatharakis D et al (2021) Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions. Comput Networks 195(108):177

Tan L, Kuang Z, Zhao L et al (2022) Energy-efficient joint task offloading and resource allocation in ofdma-based collaborative edge computing. IEEE Trans Wireless Commun 21:1960–1972

Wang K, Hu Z, Ai Q et al (2020) Joint offloading and charge cost minimization in mobile edge computing. IEEE Open Journal of the Communications Society 1:205–216

Wang Q, Guo S, Liu J, et al (2019) Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing. Sustainable Computing: Informatics and Systems 21(MAR.):154–164

Wang X, Han Y, Leung V et al (2020) Convergence of edge computing and deep learning: A comprehensive survey. IEEE Communications Surveys & Tutorials 22(99):869–904

Wang Y, Min S, Wang X et al (2016) Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. IEEE Trans Commun 64(10):4268–4282

Weng T, Zhou X, Li K et al (2021) Efficient distributed approaches to core maintenance on large dynamic graphs. IEEE Trans Parallel Distrib Syst 33(1):129–143

Yi C, Cai J, Su Z (2019) A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications. IEEE Transactions on Mobile Computing pp 1–1

You C, Huang K, Chae H, et al (2016) Energy-efficient resource allocation for mobile-edge computation offloading (extended version). Information Theory

Zervoudakis K, Tsafarakis S (2020) A mayfly optimization algorithm. Comput Ind Eng 145(106):559

Zhang H, Guo J, Yang L, et al (2017) Computation offloading considering fronthaul and backhaul in small-cell networks integrated with mec. 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) pp 115–120

Zhou S, Jadoon W (2021) Jointly optimizing offloading decision and bandwidth allocation with energy constraint in mobile edge computing environment. Computing (99)

Zhou W, Xing L, Xia J et al (2021) Dynamic computation offloading for mimo mobile edge computing systems with energy harvesting. IEEE Trans Veh Technol 70:5172–5177