Joint Optimization Strategy of Computation Offloading and Resource Allocation in Multi-Access Edge Computing Environment

Huilin Li, Haitao Xu, *Member, IEEE*, Chengcheng Zhou, Xing Lü, and Zhu Han, *Fellow, IEEE*

***Abstract*—In order to help user terminal devices (UTDs) efficiently handle computation-intensive and time-delay sensitive computing task, multi-access edge computing (MEC) has been proposed. However, due to the differences among the performance of UTDs, and the resource limitation of MEC servers, the joint optimization between the offloading decisions of UTDs and the allocation of resources in network is still a focus of the research. This paper studies the joint computation offloading and resource allocation strategy in multi-user and multi-server scenarios. Firstly, we formulate the joint optimization problem of computation offloading and resource allocation as a mixed integer nonlinear programming (MINP) problem to minimize the energy consumption of UTDs, by constraining the offloading decision, channel selection, power allocation and resource allocation. Secondly, we propose a two-stage heuristic optimization algorithm based on genetic algorithms, which divides the joint optimization problem of computation offloading and resource allocation in two stages. Based on the coupling relationship between the offloading decision and the resource allocation scheme, we iteratively update the solution of the problem, and finally obtain the stable convergence solution of the optimization problem. Finally, the proposed algorithm is compared with other classical methods to prove the effectiveness.**

***Index Terms*—Computation offloading, resource allocation, MEC, MINP, genetic algorithm.**

I. Introduction

WITH the rapid development of communication technology, the way of communication is undergoing great changes. On the one hand, a series of brand-new communication concepts, such as Internet of everything and mass machine communication, have been put forward. Along with these new communication concepts came an influx of networking devices. Smart phones, smart sensors, wearable smart devices and other kinds of user terminal devices (UTDs) all need to be connected to the network, and will undoubtedly have a huge impact on the existing network [1]. On the other hand, a variety of new communication services are beginning to enter people's daily life, such as driverless technology, virtual reality technology, augmented reality technology, natural language processing technology, and so on [2], [3]. Different from the traditional communication services, these coming technologies all require real-time and rapid processing of a large amount of data generated by the applications, and have relatively strict requirements on the time delay performance [4]–[6]. However, due to the requirements for mobility, most of the networking devices are lacking in terms of physical size, battery power, and computing power [7]. Therefore, it is still a huge challenge to execute the communication and computation tasks perfectly on UTDs.

Multi-access edge computing (MEC) is considered as an effective method to solve the limited resources problems in UTDs [8]. In MEC environments, servers with certain computing capabilities are deployed on the edge of the network, which are closer to the UTDs and can provide computing services for the UTDs [9]–[11]. Meanwhile, UTDs can offload the computation tasks to the MEC servers to achieve better performance. When the UTDs execute some computing tasks with a large amount of calculation and strict delay limitations, they can choose to offload the computing tasks to the MEC servers. In the upload process, the UTDs only need to upload some necessary parameters to the MEC servers for the computing tasks. After the calculation process which is executed by the MEC servers, the final results will be returned to the UTDs. In these way, the idle resources of the network can be effectively used. Moreover, the energy consumption of UTDs and the delay in executing computing tasks are reduced, the quality of service (QoS) can be improved.

However, the computing energy of MEC servers and the computing resources in the MEC network are not unlimited, which has caused some problems [12]. For UTDs, the purpose of using MEC mainly includes two aspects: reducing the energy consumption and reducing the task execution delay. In order to achieve these two goals, offloading decisions of UTDs and resources allocation in the network need to be jointly optimized. The offloading decisions of UTDs solve the problem of whether or not the UTDs perform computation offloading, and where to

Manuscript received April 15, 2020; revised June 13, 2020; accepted June 17, 2020. Date of publication June 19, 2020; date of current version October 13, 2020. This work was supported in part by the National Science Foundation Project of China under Grants 61971032 and 61931001, in part by Fundamental Research Funds for the Central Universities under Grants FRF-TP-18-008A3, and in part by U.S. Multidisciplinary University Research Initiative under Grants 18RT0073, NSF EARS-1839818, CNS-1717454, CNS-1731424, and CNS-1702850. The review of this article was coordinated by Dr. A.-C. Pang. *(Corresponding author: Haitao Xu.)*

Huilin Li, Haitao Xu, and Chengcheng Zhou are with the Department of Communication Engineering, University of Science and Technology Beijing (USTB), Beijing 100083, China (e-mail: 18813176122@163.com; alex_xuht@hotmail.com; czho9311@163.com).

Xing Lü is with the School of Science, Beijing Jiaotong University, Beijing 100044, China (e-mail: XLV@bjtu.edu.cn).

Zhu Han is with the Electrical and Computer Engineering Department, University of Houston, Houston, TX 77004 USA (e-mail: zhan2@uh.edu).

Digital Object Identifier 10.1109/TVT.2020.3003898

offload [13]. When the offloading decisions change, the energy consumption and delay of the UTDs in executing computing tasks also change. If a large number of UTDs offload computing tasks to the same MEC server at the same time, there may be interference problems among the UTDs. It also may lead to increased task execution delay and poor quality of service (QoS) [14]. In addition, although the MEC servers are located at the edge of the network that is closer to the UTDs, the distances between different UTDs and different MEC servers are still different. If the offloading decisions of the UTDs are not restricted, it may happen that the UTD selects a remote MEC server for computation offloading, and the final performance is not optimal. Furthermore, the transmission power and channel selection of the UTDs, and the resources provided by the MEC servers will also affect the performance [15]. When the transmission power of the UTDs increases, although the information transmission rate can be increased and the communication delay can be reduced, it will cause the energy consumption of the UTDs to increase. Therefore, how to balance the optimization between energy consumption and time delay, and to formulate an optimal offloading strategy and resource allocation strategy are the key issues of research. Therefore, we need a joint optimization strategy for computation offloading and resource allocation in the MEC environment.

Unfortunately, the joint optimization problem for computation offloading and resource allocation in the MEC environment is a mixed integer nonlinear programming (MINP) problem, which is difficult to solve directly using the conventional mathematical methods. The global optimal solution can be obtained by using exhaustive methods, nevertheless, the time complexity is too high to be accepted. Although many scholars have studied the computation offloading strategy and resource allocation strategy in the MEC environment, there is currently no universal solution method with high popularity. Based on the above situation, this paper proposes a two-stage heuristic optimization algorithm based on the genetic algorithm called THOA, which decompositions the joint optimization problem of computation offloading and resource allocation into two stages. The main work of this paper is summarized as follows,

- We investigate a scenario of multiple users and multiple servers in the MEC environment. Then we formulate the joint optimization problem of computation offloading and resource allocation as a MINP problem. The object of the problem is to minimize the overall energy consumption of the UTDs by constraining system variables such as the UTDs' offloading decisions, channel selection, power allocation, and resource allocation, while the delay requirement of each UTD is satisfied.
- To solve the proposed MINP problem, this paper proposes a two-stage heuristic optimization algorithm based on the genetic algorithm, which decomposes the joint optimization problem of computation offloading and resource allocation into two stages. In the first stage, the genetic algorithm is used to solve the offloading decisions of the UTDs under the initial situations. In the second stage, the allocation of computing resources is updated through a myopic optimization method based on the current offloading decision. After several iterations according to the coupling relationship between offloading decision and resource allocation, a stable convergent solution can be obtained.
- Extensive experiments have been carried out on specific parameter settings. We compare the proposed algorithm with other baseline methods, and numerical results show that our scheme has a better performance. Furthermore, the impact of changes in the communication environment on UTDs' offloading behavior and algorithm performance have also been fully studied.

The remainder of the paper is organized as follows: Section II introduces the related works. The system model and problem formulation are given in Section III. Section IV provides the approach for the proposed optimization problem, includes optimization algorithm and complexity analysis. Simulation results and discussions are given in Section V. Finally we conclude the work in Section VI.

II. Related Works

At present, many experts and scholars have carried out detailed research on computation offloading and resource allocation in MEC environments. From the original single-user, single-server network to the current multi-user, multi-server network, mobile edge computing technology is developing rapidly [16]. The related research work will be briefly introduced below.

Focusing on single-user MEC scenarios, [17] proposed an adaptive classification framework to solve the problem of whether UTDs execute computation offloading. To pursue the ultra-low latency in 5G cellular networks, [18] proposed the concept of the fog-radio access network (F-RAN) and provided a cooperative task computing operation algorithm to solve the problem of communication resource allocation and computing task assignment. Based on the proposed algorithm, the ultra low-latency services can be achieved by F-RAN via cooperative task computing. With the continuous expansion of the network scale, the research on multi-user and multi-server communication scenarios is also deepening. As an extension of the research in [18], [19] utilized the F-RAN to achieve the ultra-low latency, and proposed a latency-driven cooperative task computing algorithm with one-for-all concept based on dynamic programming. It is proved that the low latency services can be achieved by the proposed cooperative task computing algorithm. [20] comprehensively considered the system energy consumption and delay, and developed a multi-user, single-server computation offloading and resource allocation strategy in the vehicular network. In single-cell and multi-user scenarios, Bi and Zhang [21] considered the energy supply of UTDs and jointly optimized the computing mode and system time allocation of UTDs. For a multi-cell environment with multiple small cells, a joint framework for computation offloading and interference management was proposed in [22]. However, this article does not take into account the problem of different computing capabilities of UTDs caused by the diversity of UTDs.

The optimization goals of computation offloading and resource allocation in the MEC environment mainly include energy consumption and delay. In order to minimize the energy consumption, a joint optimization strategy for computation offloading, subcarrier allocation, and resource allocation was proposed in [23]. Guo *et al.* [24] studied the energy-saving offloading scheme of MEC systems and proposed a new optimization algorithm based on genetic algorithm and particle swarm algorithm. In [25], the authors designed the algorithm with time delay as part of the optimization goal. [26] optimized the cost of computation offloading and content caching, and maximized system utilization through convex optimization, but only considered the case of one edge server. [27] aimed to reduce the computation overhead of the offloading problem in 5G HetNets and proposed a game-theoretical offloading scheme to reduce the energy consumption and the processing delay. Although the proposed scheme can achieve offloading performance when the number of computation tasks increase, the offloading decision and the resource allocation were not considered jointly. [28] studied the optimal task offloading and resource allocation problem and proposed a bi-section search method to obtain the optimal resource allocation. A reduced-complexity Gibbs Sampling algorithm was proposed to obtain the optimal offloading decisions. But the access selection problem were not studied in the multi-server scenario.

Computing tasks can also be partially offloaded. [29] offered an approach for computation offloading and considered fixed CPU frequency and elastic CPU frequency two cases for the UTDs. A SDR-based algorithm was utilized to find the optimal solution. [30], [31] adopted the method of partial offloading, which allows the UTDs to divide the computing task into two parts and execute them at the local or MEC servers, respectively. With the help of modern technology, [32] and [33] achieved the goal of minimizing the weighted sum of offloading latency and energy consumption by using deep reinforcement learning methods. But the network model needs to be trained before the algorithm is executed, which may take a long time. In [34], the UTDs can offload computing tasks to multiple MEC servers, which provides a new idea for solving the problem of insufficient computing resources on the MEC server–collaborative offloading. [35], [36] applied device to device (D2D) communication technology to MEC networks, but only considered the single-server scenario. [37] considered a multi-user, multi-server communication scenario, and maximized the benefits of the system through collaboration between MEC servers. Except from the collaboration within the edge layer, edge layer devices can also collaborate further with cloud layer devices. [38] proposed a method that integrates fog and cloud computing. UTDs can collaboratively offload a series of applications to nearby fog nodes or cloud centers.

Different from the existing research work, this paper mainly studies the problem of computation offloading and resource allocation in multi-user, multi-server scenarios. The superiorities of this paper are as follows. Firstly, the communication scenario studied in this article is multi-user and multi-server, which is in line with the current communication environment of mass intelligent terminal equipment and multiple base stations. Secondly, we comprehensively analyze the interaction between the offloading decision and resource allocation, and jointly optimized the two. Finally, the algorithm proposed in this paper divides the optimization problem into two parts to solve, which greatly reduces the complexity of the algorithm, and can obtain the stable convergent optimal solution of the problem in a short time.

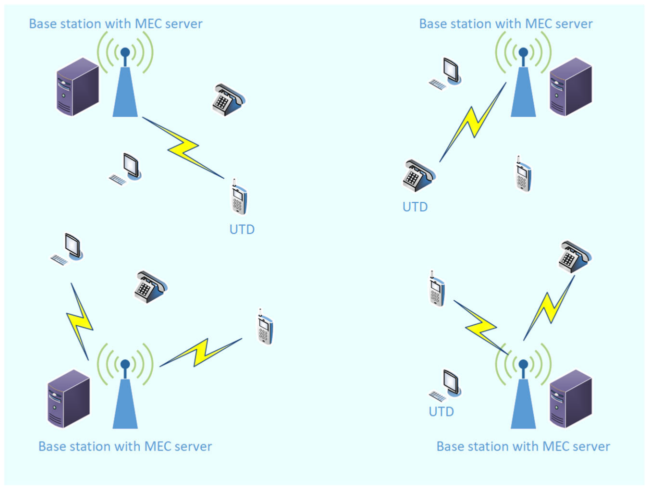
Base station with MEC server
Base station with MEC server
UTD
UTD
UTD
Base station with MEC server
Base station with MEC server

Fig. 1. System Model.

III. System Model and Problem Formulation

The communication scenario studied in this paper is shown in Fig. 1. Within a certain rectangular area, there are S MEC servers and N UTDs. The MEC servers are evenly distributed in the area, the set of which is denoted by $S = \{1, 2, 3, \ldots, S\}$. The coverage of each MEC server is assumed to include the entire rectangular area. UTDs are randomly distributed in the rectangular area, and is represented by a set $N = \{1, 2, 3, \ldots, N\}$. Each UTD has a pending computing task that can be executed locally or offloaded to any MEC server. The offloading decisions of the UTDs are represented by a matrix $A = a_n, n \in N$, and the values of element in matrix are as follows,

- $a_n = 0$, which means UTD n chooses to execute the computing task locally.
- $a_n = s, s \in S$, which indicates UTD n chooses to offload the computing tasks to the MEC server for execution.

Due to different offloading behaviors of UTDs, there are two execution model that will be described in detail in the following subsection.

A. Local Computing

When choosing to execute the computing tasks locally, the computing tasks of UTD n can be described by three items: the amount of input data in bits, express as D_n, the number of computer cycles required to calculate unit data in CPU cycles/bit, denoted as C_n, and the maximum tolerance time, presented as τ_n. When a UTD executes the computing task locally, it needs to allocate computing resources to the task, and the computing capacity allocation matrix is denoted as $FL = \{fl_n, n \in N\}$, where fl_n represents the computing capacity of UTD in Hz.

The time required for UTD to execute the computing task locally is,

$$T_n^l = \frac{D_n C_n}{fl_n}. \tag{1}$$

The energy consumption of executing tasks locally depends on the number of computer cycles required [39], which is given as,

$$E_n^l = k_0 fl_n^2 D_n C_n, \tag{2}$$

where $k_0 > 0$ is the effective capacitance coefficient.

B. MEC Server Computing

When the UTDs offload computing tasks to the MEC servers, the UTDs need to send input parameters required for calculation to the servers first. Then the MEC servers calculate tasks based on the received data. After the calculation, the MEC servers return the results to the UTDs. It is worth noting that the amount of results data will be greatly reduced compared to the input data. Hence, this article does not discuss the task execution time and energy consumption during the results return phase. We only analyzes the data transmission phase and task execution phase under the MEC server computing situation.

1) Data Transmission Phase: Assuming that the data transmission process between the UTDs and the MEC servers uses orthogonal frequency division multiple access (OFDMA) technology, then the interference among the UTDs can be ignored. Let K denote the total number of channels in the MEC environment, and B denote the bandwidth. The channel selection of the UTDs is represented by a matrix $W = \{\varpi_n, n \in N\}$, and ϖ_n in the matrix represents the number of channels occupied by UTD n. The transmission power matrix of the UTDs is denoted by $P = \{p_n, n \in N\}$. Thus, the transmission rate can be given by,

$$R_n = B\varpi_n \log_2\left(1 + \frac{g_n p_n}{\sigma^2}\right), \tag{3}$$

where g_n is the channel gain, and σ is the variance of Gaussian white noise. Based on (3), the time delay during data transmission is denoted by,

$$T_n^{tr} = \frac{D_n}{R_n} = \frac{D_n}{B\varpi_n \log_2\left(1 + \frac{g_n p_n}{\sigma^2}\right)}. \tag{4}$$

Then the energy consumption during the data transmission phase can be expressed as,

$$E_n^{tr} = p_n T_n^{tr} = p_n \frac{D_n}{B\varpi_n \log_2\left(1 + \frac{g_n p_n}{\sigma^2}\right)}. \tag{5}$$

2) Task Execution Phase: It is assumed that the computing capacity of each MEC server is the same, and is denoted by F. The computing capacity of each MEC server is evenly distributed to all UTDs that select this server. In this case, the execution time of computing task is as follow,

$$T_n^{exe} = \frac{D_n C_n}{F_n}, \tag{6}$$

where F_n represents the computing resource obtained by the UTD n. Since the energy consumption during the task execution is borne by the MEC servers, it is not considered as a part of the optimization goal.

C. Problem Formulation

Based on the above descriptions, it can be known that for UTD n, the time required to execute the computing task is as follows,

$$\begin{aligned} &T_n(A, W, P, FL) \\ &= (1 - \varphi(a_n))\, T_n^l + \varphi(a_n)(T_n^{tr} + T_n^{exe}) \\ &= (1 - \varphi(a_n)) \frac{D_n C_n}{fl_n} \\ &\quad + \varphi(a_n) \left(\frac{D_n}{B\varpi_n \log_2(1 + \frac{g_n p_n}{\sigma^2})} + \frac{D_n C_n}{F_n} \right). \end{aligned} \tag{7}$$

The energy consumption is denoted by,

$$\begin{aligned} &E_n(A, W, P, FL) \\ &= (1 - \varphi(a_n)) E_n^l + \varphi(a_n) E_n^{tr} \\ &= (1 - \varphi(a_n)) k_0 fl_n^2 D_n C_n \\ &\quad + \varphi(a_n) \frac{p_n D_n}{B\varpi_n \log_2(1 + \frac{g_n p_n}{\sigma^2})}. \end{aligned} \tag{8}$$

In the above formulas, the function $\varphi(a_n)$ is an indicated function and can be defined as follows,

$$\varphi(a_n) = \begin{cases} 0, & \text{if} \quad a_n = 0, \\ 1, & \text{if} \quad a_n = 1. \end{cases} \tag{9}$$

In summary, the joint optimization problem of computation offloading and resource allocation can be modelled as follow,

$$\begin{aligned} \min_{A,W,P,FL} \quad & \Sigma_{n \in N} E_n(A, W, P, FL) \\ \text{s.t.} \quad & C1: T_n(A, W, P, FL) \le \tau_n, \forall n \in N, \\ & C2: \Sigma_{n \in N} F_(n) \le F, \\ & C3: \Sigma_{n \in N} \varpi_n \le K, \\ & C4: a_n \in \{\{0\} \bigcup s\}, \forall n \in N, \\ & C5: 0 \le P_n \le P_{\max}, \forall n \in N, \\ & C6: 0 \le fl_n \le fl_{\max}, \forall n \in N. \end{aligned} \tag{10}$$

In the above formulation, the objective function is the energy consumption of all UTDs. Constraint C1 guarantees that the delay of each UTD is within a tolerable range. C2 requires that each MEC server's computing resource allocated to the UTDs does not exceed the total computing resource capacity. C3 indicates that the number of channels occupied by all UTDs does not exceed the total number of channels in the network. It should be mentioned that each channel can only be allocated to one UTD. C4 is the range of the UTDs' offloading decisions, and a UTD can only offload the computing tasks to one MEC server at the same time. Constraints C5 and C6 state the range of the variables.

IV. Proposed Approach

A. Problem Analysis

According to the problem formulation in the previous section, it can be seen that the optimization problem studied in this paper is a MINP problem. Moreover, there is a strong coupling relationship among the optimization variables. In order to obtain the optimal solution of the optimization problem, an effective solution method is needed.

Further analyze the objective function given in the previous section, we take part the joint optimization problem of computation offloading and resource allocation into two stages. The first stage is to determine the UTDs' offloading behavior, i.e., to decide the UTDs' offloading decision A, channel selection W, and power allocation P. We can use heuristic algorithms such as the genetic algorithm to solve this stage of the problem. The detailed flow of the algorithm is shown in Section IV-C. The second stage is to solve the resource allocation problem, i.e., to determine the calculation frequency of CPU FL allocated by the UTDs for the calculation of the task. When the UTDs' offloading behavior is determined, there is a quadratic relationship between the energy consumption of the UTDs and the calculation frequency of CPU. We can obtain the lower bound of the calculation frequency of CPU by the maximum tolerance time of UTDs, thereby obtaining the optimal solution of the resource allocation problem. The transformed problem formulation and the detailed flow of the algorithm are shown in Section IV-D. Furthermore, there is a strong coupling between the optimal solution of UTDs' offloading behavior and resource allocation. The optimal solution of one will be used as input to participate in the parameter solving process of the other. Therefore, after several iterative updates of the solution, we can obtain the stable convergent optimal solution of the original optimization problem. Observing the objective function given in the previous section, it can be found that when the UTD's offloading decision, channel selection, and transmission power are constant, the energy consumption of the UTD is a monotonic function about the computing capacity under the conditions of constraints C1 and C6. It is easy to get the optimal solution for computing capacity. Furthermore, other optimization variables can be updated through the computing capability matrix. After multiple iterations, the optimal solution to the optimization problem can be obtained.

B. Two-Stage Heuristic Optimization Algorithm

Based on the above ideas, we propose a two-stage heuristic optimization algorithm called THOA, the two-stage heuristic optimization algorithm. The algorithm flow is shown in Algorithm 1. We use random initialization to initialize the variables according to the constraints in the proposed problem. In particular, because the penalty term mechanism is introduced in the implementation of the genetic algorithm in this paper, constraints C1 and C3 may not be satisfied when the variables

```
Algorithm 1: General Framework of THOA
Input: Set the maximum number of iterations $gen_{max}$,
       initialize the current number of iterations $gen$ to
       $0$, randomly initialize $A$, $W$, $P$ and $FL$ of UTDs
Output: The optimal solution of the problem and its
        corresponding optimal objective function value
1 while $gen < gen_{max}$ do
2     Compute the optimal offloading decisions $A_*$, $W_*$,
      $P_*$ under the given $FL$ via algorithm 2;
3     Calculate the optimal resource allocation $FL_*$ based
      on $A_*$, $W_*$, $P_*$ via algorithm 3;
4     Calculate the energy consumption of the UTDs using
      $A_*$, $W_*$, $P_*$ and $FL_*$, update the minimum energy
      consumption record, and update the optimal solution
      record;
5     $FL = FL_*$, $gen = gen + 1$;
6 end
```

are initialized. The initialization of variables is as follows:

$$a_n = randi(\{0\} \bigcup S) \tag{11}$$

$$\varpi_n = \begin{cases} randInt(K), & \text{if } a_n \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

$$p_n = \begin{cases} rand(p_{\max}), & \text{if } a_n \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

$$fl_n = \begin{cases} rand(fl_{\max}), & \text{if } a_n \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where the function $randi(Set)$ is used to randomly select one of the values from the given set Set. The function $randInt(x)$ and $rand(x)$ are used to randomly obtain an integer and a real number from the range of $[0, x]$, respectively. After the initialization is completed, the main iterative process of the algorithm begins. Except for the first iteration, the input of the subsequent iteration process is determined by the result of the previous iteration. During each iteration, the offloading decision and resource allocation are solved, and the minimum energy consumption of the UTDs and the corresponding optimal offloading decision and resource allocation scheme are updated. Until the number of iterations reaches the set value, the algorithm stops and the final result is obtained.

C. Offloading Decision Optimization

To solve the offloading decision optimization problem, the offloading decision, channel selection and power allocation of the UTDs need to be optimized. However, since there are both integer and non-integer variables, the problem is difficult to solve by traditional methods. In this section, genetic algorithm is chosen to optimize the offloading decision. The genetic algorithm does not require that the optimization problem to be solved is continuous or differentiable, and the execution direction of the algorithm does not need to be controlled. The algorithm flow is shown in Algorithm 2.

Algorithm 2: Genetic Algorithm

```
Input: Population size $L$, Crossover probability $p_c$,
       Mutation probability $p_m$
Output: The optimal offloading decisions $A_*$, $W_*$, $P_*$
1  Initialization: Use the initialization parameters in
   Algorithm 1 if it is the first iteration, otherwise use the
   results after the last iteration. Define an empty set $G$ to
   represent the individuals of the next generation;
2  for $i \in L$ do
3      if $I_i$ is feasible then
4          $mortality_i = E_i(I_i, FL_i)$;
5      else
6          $mortality_i = E_i(I_i, FL_i) + \rho_i$;
7      end
8  end
9  Record the optimal offloading decisions $A_*$, $W_*$, and $P_*$
   according to the chromosomal information $I_i$ of the best
   individual of this generation, which has the minimum
   mortality value;
10 while the size of $G$ is below than $L$ do
11     Randomly select $L/2$ individuals from the
       contemporary generation;
12     Select the two individuals with the smallest mortality
       value to join the set $G$;
13     Perform crossover operations on them under the
       crossover probability, as shown in Fig.2;
14 end
15 for $j \in G$ do
16     Perform mutation operations on $I_j$ under the
       mutation probability according to (18) and (19), as
       shown in Fig.3
17 end
```

At the first iteration, we randomly initialize a set of feasible solutions as individuals in the population, and thereafter the output of each iteration is used as the input for the next iteration. Then we calculate the mortality value of all individuals in the population and record the best individuals in this iteration. Next, we use the tournament method to screen the next generation of individuals, similar to the natural selection process, this selection makes it easier for individuals with lower mortality value to survive to the next generation, that is, individuals with lower energy consumption of UTDs are more likely to survive. Finally, we perform crossover and mutation operations on the next generation of individuals to prevent the algorithm from falling into a local optimum during the iteration process. With the increase of the number of iterations, the feasible solutions that cause excessive energy consumption of UTDs are gradually eliminated. After a limited number of iterations, we can get the optimal solution for the offloading decision problem.

The following section will introduce some related knowledge of genetic algorithm.

1) Chromosome and Fitness: Drawing on the idea of natural selection [40], the genetic algorithm introduced the concept of chromosomes. In the genetic algorithm, the chromosome corresponds to the solution of the optimization problem. The offloading decision, channel selection, and power allocation of the UTDs constitute the individual's chromosome information. For an individual $i(i \in [1, L])$, its chromosomal information can be represented by a matrix with the following structure,

$$I_i = [A_i, W_i, P_i]^T = \begin{bmatrix} a_1^i, a_2^i, a_3^i, \ldots, a_n^i \\ \varpi_1^i, \varpi_2^i, \varpi_3^i, \ldots, \varpi_n^i \\ p_1^i, p_2^i, p_3^i, \ldots, p_n^i \end{bmatrix}. \tag{15}$$

Fitness is used to measure how well an individual adapts to the environment. Individuals with poor adaptability will be eliminated. Because the problem of minimization is studied in this article, this article defines the mortality as a measure of the quality of the solution. In addition, while calculating the mortality, a penalty term ϱ is added to ensure that the final solution of the algorithm is within the constraints. The mortality of individual $i(i \in [1, L])$ is computed as follows,

$$mortality_i = \begin{cases} E(I_i, FL_i), & \text{if } i \text{ is feasible} \\ E_{worst} + \varrho_i, & \text{otherwise} \end{cases} \tag{16}$$

If the solution corresponding to this individual is a feasible solution, the mortality is the value of the objective function. Otherwise, the mortality is the objective function value of the worst feasible solution in the current population plus a penalty term. If there is no feasible solution in the current population, f_{worst} is set to zero. The penalty terms are defined as follows,

$$\varrho_i = \sum_{n \in N} [T_n(A_i, W_i, P_i, FL_i) - \tau_n] + \sum_{\substack{n \in N, \\ \varpi_n \in W_i}} \varpi_n - K. \tag{17}$$

After adding a penalty term to the mortality, the following can be guaranteed [41],

- The mortality of feasible solutions is lower than the mortality of infeasible solutions.
- Between two feasible solutions, the mortality of feasible solution with a smaller value of the objective function is lower.
- Between two infeasible solutions, the mortality of infeasible solution with smaller constraint deviations is lower.

2) Select, Crossover, and Mutation: In order to select the parental group of the next generation, this paper uses the tournament method to screen the contemporary individuals. When the population size is L, half of the individuals are randomly taken from the population at each time. Then the two individuals with the lowest mortality are selected from the tournament. This process is repeated until the number of parental candidate populations reaches the original population size. Through the chromosomal crossover operation between the parents, the superior genes of the parents can be inherited, so that the solution of the problem can be developed in a direction that can make the value of the objective function smaller. Two parental individuals selected in each round of the tournament conduct chromosomal crossover with probability p_c, and two offspring individuals are generated to participate in the next round of iteration. Taking the offloading decision information of the UTD in the chromosome information as an example, the crossover operation is shown in Fig. 2.

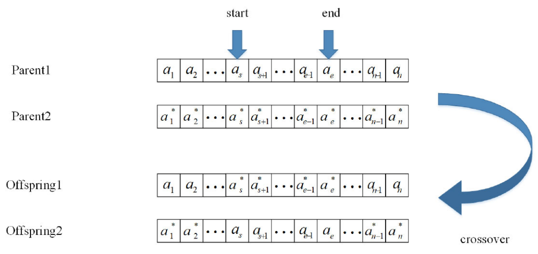

Fig. 2. Crossover.

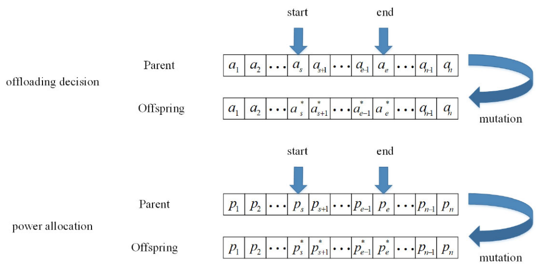

Fig. 3. Mutation.

In order to prevent the genetic algorithm from quickly falling into a local optimal solution, the diversity of solutions needs to be increased. Then the concept of mutation is introduced. The following part uses the offloading decision (integer value) and power allocation (real value) of the UTD as examples to explain the mutation process, as shown in Fig. 3.

The mutation principle is shown as follows,

$$a_i^* = \begin{cases} round(a_i + \alpha(a_{\max} - a_i)), & \text{if } \beta \geq 0.5 \\ round(a_i - \alpha(a_i - a_{min})), & \text{if } \beta < 0.5 \end{cases} \tag{18}$$

$$p_i^* = \begin{cases} p_i + \alpha(p_{\max} - p_i), & \text{if } \beta \geq 0.5 \\ p_i - \alpha(p_i - p_{min}), & \text{if } \beta < 0.5 \end{cases} \tag{19}$$

where function $round(\cdot)$ is used to round the variables. α and β are random numbers in the range of (0, 1), where α determines the step size of the mutation and β determines the direction of the mutation.

D. Resource Allocation Optimization

The resource allocation optimization is given as follows,

$$\begin{aligned} g(FL) = \min_{FL} \sum_{n\in N} &(1 - \varphi(a_n))k_0(fl_n)^2 D_n C_n \\ &+ \varphi(a_n)\frac{p_n D_n}{B\varpi_n \log_2(1 + \frac{g_n p_n}{\sigma^2})} \end{aligned} \tag{20}$$

$$\begin{aligned} \text{s.t.} \quad C1 : &(1 - \varphi(a_n))\frac{D_n C_n}{fl_n} \\ &+ \varphi(a_n)\left(\frac{D_n}{B\varpi_n \log_2(1 + \frac{g_n p_n}{\sigma^2})} + \frac{D_n C_n}{F_n}\right) \leq \tau_n \\ C2 : &0 \leq fl_n \leq fl_{\max}, \forall n \in N \end{aligned}$$

When the offloading decision, channel selection and power allocation of the UTDs are constant, the energy consumption of the UTDs is a quadratic function of the calculation frequency of CPU FL, as shown in (20). Since the calculation frequency of CPU assigned by the UTDs for the calculation of the task is a non-negative number, as it increases, the energy consumption of the UTDs also increases. In order to minimize the energy consumption of UTDs, the calculation frequency of CPU needs to be reduced as much as possible. However, a low calculation frequency of CPU can cause long calculation delays, even exceeding the maximum tolerance time of UTDs. Therefore, the lower bound of the calculation frequency of CPU can be obtained by the maximum tolerance time of UTDs. When this value exceeds the maximum computing frequency that the UTDs can provide, the optimal solution for resource allocation is the maximum computing frequency of the UTDs, otherwise the optimal solution is the lower bound of the calculation frequency of CPU. In summary, the optimal computing resource allocation solution can be obtained through a myopic optimization method. The algorithm is as follows. When the UTDs have certain strategies for offloading decision, channel selection, and power allocation, the energy consumption of the UTDs will be a strictly monotonous function of the computing resource. The problem under these situations is shown as in (20). Therefore, the optimal computing resource allocation scheme can be obtained through a monotonic optimization method. The algorithm is as follows.

Algorithm 3: Myopic Optimization Method

```
Input: Optimal offloading decision $A$, channel selection
       $W$, and power allocation $P$ obtained by genetic
       algorithm
Output: Optimal computing resource allocation $FL$
1  for $n \in N$ do
2      if $a_n \neq 0$ then
3          $fl_n = 0$;
4      else
5          if $\frac{D_n C_n}{\tau_n} \geq fl_{max}$ then
6              $fl_n = fl_{max}$;
7          else
8              $fl_n = D_n C_n / \tau_n$;
9          end
10     end
11 end
```

E. Computation Time Complexity of THOA

In the initialization phase, the input parameters of the algorithm need to be initialized, and the time complexity is $\mathcal{O}(L \times N)$. In the genetic algorithm, the time complexity of calculating the individual mortality is $\mathcal{O}(L \times N)$, the time complexity of the tournament selection process and the crossover process is $\mathcal{O}(L^2)$, the time complexity of the mutation process is $\mathcal{O}(L \times N)$. Because $L > N$, the time complexity of genetic algorithm is $\mathcal{O}(L \times N)$. The time complexity of the myopic optimization method is $\mathcal{O}(N)$. The time complexity of updating the optimal solution and the optimal value is $\mathcal{O}(N)$.In each iteration, the genetic algorithm, the myopic optimization method and the update of optimal solution and optimal value will be executed once. Assume the total number of iterations are T, the time complexity of the algorithm proposed is $\mathcal{O}(T \times L \times N)$.

V. Numerical Results and Discussion

In this section, we conduct simulation experiments and analyze the performance of the proposed algorithm. The proposed algorithm is also compared with other methods. Firstly, we introduces the simulation scene and specific experimental parameters. Then, the influence of various parameters of the genetic algorithm on the experimental results is analyzed. And the influence of parameters, such as the number of channels, the number of UTDs, and the number of MEC servers, are discussed. In order to clearly quantify the algorithm performance, we mainly considers two performance indicators: user terminal energy consumption and the number of users performing offloading. In addition, to improve the reliability of the experimental results, all data are the average of the results of 10 independent experiments.

A. *Simulation Setting*

The communication scenario in the simulation is set to a rectangular area of 1000 m $\times$ 1000 m, which contains several MEC servers and UTDs. The MEC servers are evenly distributed in the area, and the UTDs are randomly distributed. In order to study the effect of the number of UTDs, the number of MEC servers, and the number of channels on the performance of the algorithm, the number of UTDs is set from 50 to 150, the number of servers is set to $[4, 8, 16, 24, 32]$ and the number of channels is set from 50 to 150. Other simulation parameters such as the computing capacity of UTDs, the input data size, the number of CPU cycles required for unit data, maximum transmission power of UTDs, the channel bandwidth, and white Gaussian noise are similar to those in [23] and [24]. According to [24] and [38], the computing capacity of the server is usually 2–5 times that of the UTDs, so the computing capacity of the server is set to 3 times that of the UTDs. Unless otherwise specified, the default simulation parameters are shown as follows.

To verify the performance of the algorithm, this paper compares the proposed algorithm with the following three methods,

- Local execution algorithm (LEA): All of the UTDs do not perform computation offloading, and all computing tasks are executed locally.
- Nearest offloading algorithm (NOA): Ensure that as many UTDs as possible perform computation offloading. When the channel is sufficient, all UTDs perform computation offloading, otherwise, UTDs are randomly selected for computation offloading until the channel resources are exhausted. UTDs offload computing tasks to the nearest MEC server, and computing resources in the network are randomly allocated.
- Random offloading algorithm (ROA): The UTDs randomly decide to perform a computing task locally or offload it to a random MEC server, and the computing resources in the network are randomly allocated.

B. *Performance Discussion*

Depending on the population size, crossover probability, and mutation probability, the execution results of the genetic algorithm may be different, and thus the performance of the algorithm proposed in this paper will also be affected. Fig. 4 shows how the energy consumption of UTDs changes with the size of the genetic algorithm population. It can be seen from the figure that the energy consumption of UTDs decreases first and then increases as the population size increases. This is because the individual diversity is insufficient when the population is small, and it is difficult to obtain the optimal solution within a limited number of iterations. However, when the population is too large, elimination of bad individuals is too slow. The effect of the crossover probability on the performance of the algorithm is shown in Fig. 5. When the crossover probability is 0.6, the algorithm performs the best. This is because when the crossover probability is low, the superior genes of the parents cannot be inherited to the next generation of individuals. Too high cross probability will lead to too many non-optimal individuals, which is also not conducive to the implementation of the algorithm. According to Fig. 6, we can know that the most appropriate mutation probability is 0.02.

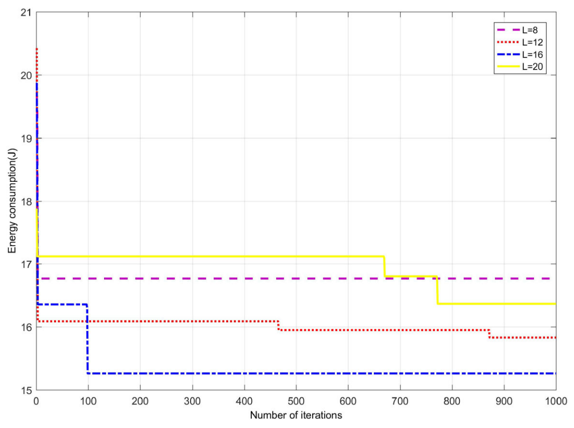

Fig. 4. The energy consumption versus the population size.

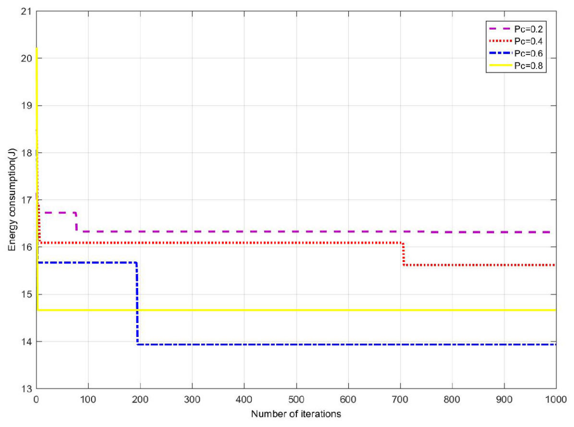

Fig. 5. The energy consumption versus the crossover probability.

In order to measure the effectiveness of the algorithm proposed in this paper, we compared the algorithm proposed in this paper with the exhaustive method. The exhaustive method can obtain the global optimal solution of the optimization problem by traversing and comparing all feasible solutions, but the complexity of the algorithm will increase exponentially as the

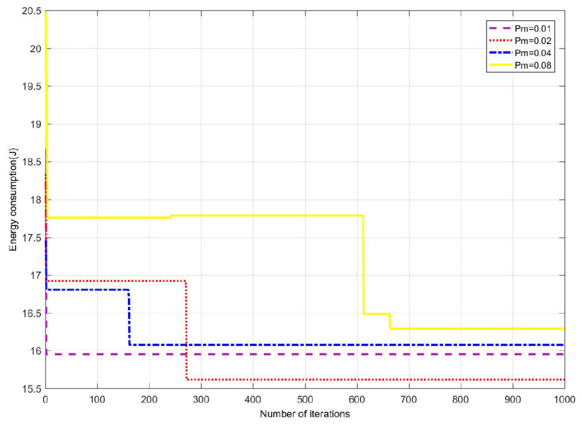

Fig. 6. The energy consumption versus the mutation probability.

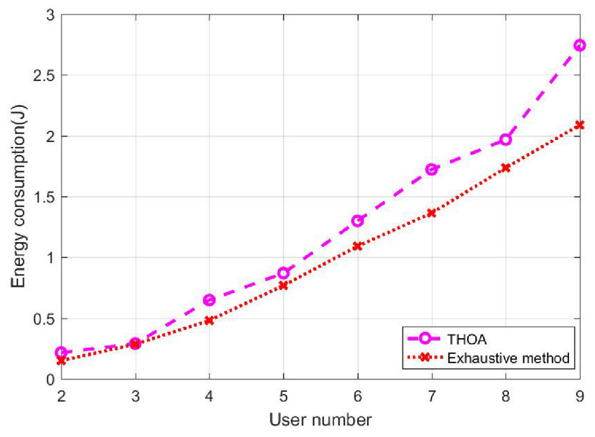

Fig. 7. Performance comparison between the proposed algorithm and exhaustive method.

TABLE I
SIMULATION PARAMETERS SETTING

Simulation Parameters	Value
The number of MEC servers	36
The number of UTDs	100
Computing capacity of UTDs	1-2GHz
Computing capacity of MEC servers	6 GHz
The input data size of UTDs	200-400 KB
The computer cycles required for unit data	500-1000 cycles/bit
The channel bandwidth	12.5kHz
Background noise	10^{-13}
The maximum tolerance time of UTDs	<=0.5s
The maximum transmission power of UTDs	1W
The number of channels	128
The maximum number of iterations	1000
The probability of crossover	0.6
The probability of mutation	0.02
The pass loss exponent	3

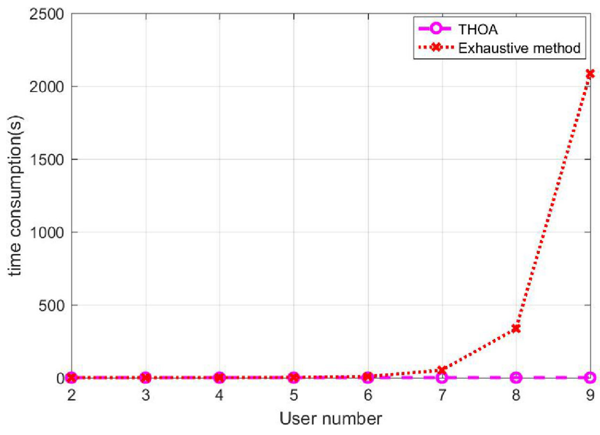

Fig. 8. Comparison of execution time between the proposed algorithm and exhaustive method.

TABLE II
THE RUNNING TIME OF TWO ALGORITHMS UNDER DIFFERENT NUMBER OF UTDS

No. of Users	THOA	Exhaustive method
2	0.118133s	0.008335s
3	0.149898s	0.034804s
4	0.174045s	0.208324s
5	0.173756s	1.291033s
6	0.177355s	8.108498s
7	0.178620s	51.733513s
8	0.202556s	336.257382s
9	0.207990s	2085.578411s
10	0.209620s	Over 2 hours

problem size increases. In a 50 m × 50 m rectangular communication scenario with two MEC servers and ten channels, when the number of UTDs increases, the running time of the algorithms proposed in this paper and the running time of the exhaustive methods are shown in Table II. It can be seen from Table II that when the number of UTDs is 9, the running time of the exhaustive method is 2085.578411 s, which has far exceeded the delay requirements of actual communication scenarios. When the number of UTDs reaches 10, the run time of the exhaustive method even exceeds two hours. In order to avoid the algorithm execution time being too long, we reduced the simulation scene to a rectangular area of 50 m × 50 m, which contains two MEC servers, up to nine UTDs and ten channels. The other simulation parameters are default, as shown in Table I. The performance difference between the two algorithms is shown in Fig. 7 and Fig. 8.

Fig. 7 shows the changes in the energy consumption of UTDs with the number of UTDs when two different algorithms are applied respectively. As can be seen from the figure, the performance of the proposed algorithm is excellent, and the gap between the results obtained and the global optimal solution obtained by the exhaustive method is extremely small. Fig. 8 shows the execution time of the two algorithms. It can be seen from the figure that the execution time of the algorithm proposed in this paper increases linearly, while the execution time of the exhaustive method increases exponentially. Although the execution time of the algorithm proposed in this paper is relatively long at first, as the number of UTDs increases, the execution time of the exhaustive method rapidly exceeds the execution time of the algorithm proposed in this paper. Therefore, in today's large-scale networks, applying the algorithm proposed in this paper can significantly reduce the algorithm execution time and improve execution efficiency.

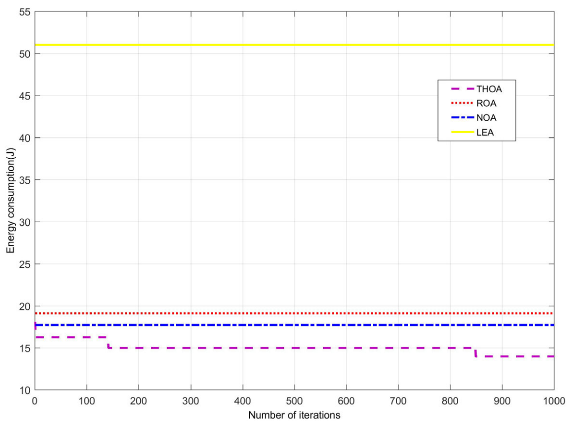

Fig. 9. The comparison of four algorithms.

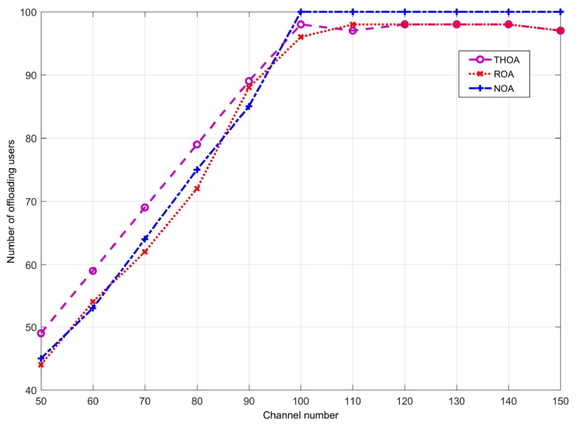

Fig. 11. The number of UTDs performing computation offloading versus the channel number.

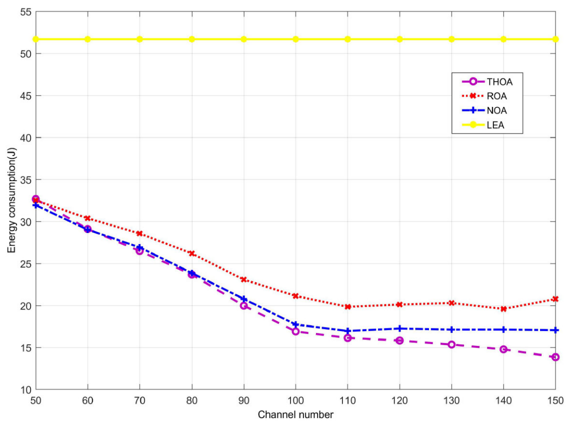

Fig. 10. The energy consumption versus the channel number.

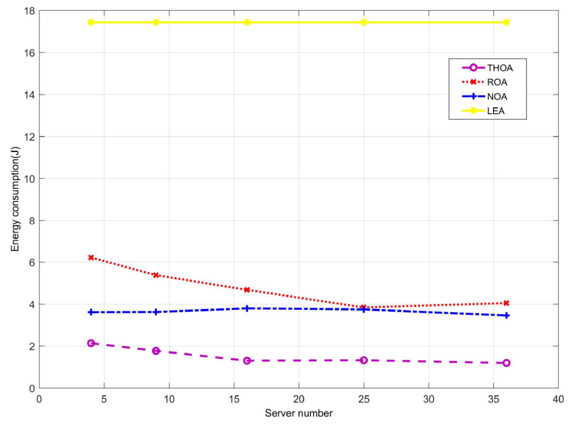

Fig. 12. The energy consumption versus the sever number.

The comparison of the four algorithms under the default simulation parameters is shown in Fig. 9. In LEA, all UTDs execute computing tasks locally and consume much more energy than the other three algorithms. Some of UTDs in ROA offload computing tasks to the MEC servers, which reduces the energy consumption. All UTDs in NOA perform computation offloading, but there is no appropriate optimization strategy, so the optimal energy consumption situation cannot be achieved. The algorithm proposed in this paper has jointly optimized offloading decision and resource allocation, and its performance is better than the other three algorithms. The energy consumption of UTDs decreases with the number of iterations, and eventually converges to the optimal value.

The influence of the number of channels on the energy consumption of the UTDs and the number of UTDs performing computation offloading is shown in Fig. 10 and Fig. 11. As the number of channels increases, the energy consumption of the UTDs decreases, the number of the UTDs for computation offloading increases, and eventually both tend to stabilize. Note that the performance of the proposed algorithm is always better than other algorithms. With the increase of the number of channels, the average number of available channels of the UTDs increases, so the information transmission rate with the MEC servers increases, thereby reducing the energy consumption of the UTDs. In addition, the increase in the number of channels enables more UTDs to perform computation offloading. However, due to the limited computing resources of the MEC servers, too many UTDs entering the network will increase the execution delay of computing tasks, so the system will eventually reach a stable state.

Fig. 12 and Fig. 13 respectively show the changes in the energy consumption of UTDs and the number of UTDs performing computation offloading as the number of MEC servers increases. In order to ensure the simulation effect when the number of MEC servers is small, the number of UTDs in this simulation is set to 30. With the increase of the number of MEC servers, the energy consumption of UTDs in ROA and the algorithm proposed in this paper decreases, and the number of UTDs performing computation offloading increases, and eventually remains stable. The reason is that when the number of MEC servers increases, the average number of UTDs served by each MEC server decreases. UTDs can get more energy consumption and delay benefits through computation offloading, so more UTDs tend to perform computation offloading instead of executing computing tasks locally. However, due to the limitation of the number of channels, the number of UTDs performing computation offloading will reach saturation. The offloading decisions of the UTDs in LEA

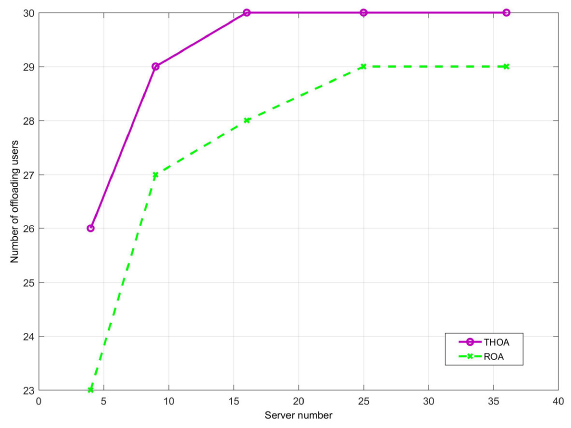

Fig. 13. The number of UTDs performing computation offloading versus the server number.

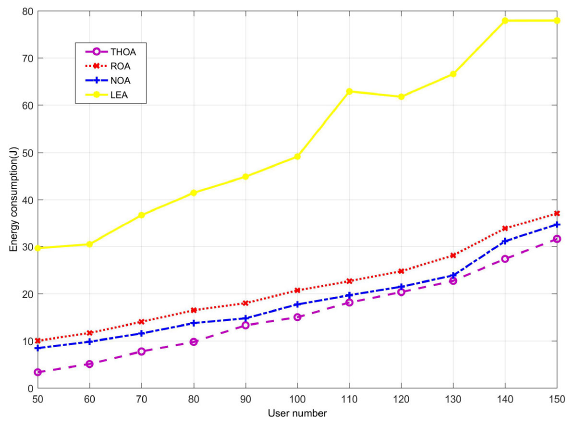

Fig. 14. The energy consumption versus the user number.

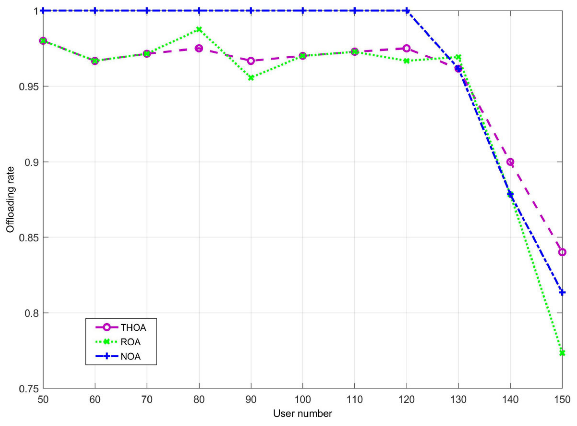

Fig. 15. The offloading rate versus the user number.

and NOA does not change with the number of servers, thus the performance of the algorithm is basically unchanged.

As the number of UTDs increases, so does the energy consumption. It can be seen from Fig. 14 that the energy consumption of the UTDs has a linear relationship with the quantity. The algorithm proposed in this paper has the best performance and always keeps the energy consumption of UTDs to a minimum value. It does not make sense to simply consider the number of UTDs performing computation offloading when the number of UTDs changes, the offloading rate needs to be studied, as shown in Fig. 15. When the number of UTDs is small, the offloading rate is high. As the number of UTDs increases, due to the limited network resources, some UTDs can only choose to execute computing tasks locally, resulting in a reduction in the offloading rate.

VI. Conclusion

In this paper, the joint optimization strategy of computing offloading and resource allocation in a multi-user terminal device and multi-edge server scenario under the multi-access edge computing environment is studied. In order to achieve the goal of minimizing the energy consumption of user terminal devices, the joint optimization problem of computation offloading and resource allocation is modeled as a mixed integer nonlinear programming problem by restricting the offloading decisions, channel selection, power allocation, and resource allocation. Then a two-stage heuristic optimization algorithm based on genetic algorithm is proposed, which decomposes the joint optimization problem into two stages for solving. Finally, simulation experiments are performed to verify the performance of the proposed algorithm. Under the same simulation parameters, the algorithm proposed in this paper fully considers the interaction between the offloading decision and resource allocation, thus can achieve lower energy consumption of user terminal devices than other algorithms. Moreover, our algorithm has high calculation efficiency and can quickly converge in fewer iterations. When the number of available channels and the number of edge servers increase, the energy consumption of user terminal devices tends to decrease, and the number of users performing calculation offload increases. When the number of user terminal devices in the network increases, the sum of the energy consumption of the user terminal devices increases, and the proportion of user terminal devices performing computation offloading decreases. Compared with other algorithms, the algorithm proposed in this paper can always make the energy consumption of user terminal devices the lowest and has good performance.

References

[1] M. Shafi *et al.*, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.

[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[3] J. Ren, Y. Guo, D. Zhang, Q. Liu, and Y. Zhang, "Distributed and efficient object detection in edge computing: Challenges and solutions," *IEEE Netw.*, vol. 32, no. 6, pp. 137–143, Nov./Dec. 2018.

[4] J. Shen, H. Tan, J. Wang, J. Wang, and S. Lee, "A novel routing protocol providing good transmission reliability in underwater sensor networks," *J. Internet Technol.*, vol. 16, no. 1, pp. 171–178, Jan. 2015.

[5] Z. Pan, Y. Zhang, and S. Kwong, "Efficient motion and disparity estimation optimization for low complexity multiview video coding," *IEEE Trans. Broadcast.*, vol. 61, no. 2, pp. 166–176, Jun. 2015.

[6] J. Zhang, F. R. Yu, S. Wang, T. Huang, Z. Liu, and Y. Liu, "Load balancing in data center networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2324–2352, Jul.–Sep. 2018.

[7] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.

[8] A. Ndikumana *et al.*, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020.

[9] B. Dab, N. Aitsaadi, and R. Langar, "A novel joint offloading and resource allocation scheme for mobile edge computing," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf.*, Las Vegas, NV, USA, Jan. 2019, pp. 1–2.

[10] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with caching enhancement for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11098–11112, Nov. 2018.

[11] Y. Wang, M. Liu, J. Yang, and G. Gui, "Data-driven deep learning for automatic modulation recognition in cognitive radios," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 4074–4077, Apr. 2019.

[12] Z. Zheng, L. Song, Z. Han, G. Y. Li, and H. V. Poor, "A stackelberg game approach to proactive caching in large-scale mobile edge networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5198–5211, Aug. 2018.

[13] Y. Wei, D. Guo, F. R. Yu, and Z. Han, "Joint optimization of caching, computing, and radio resources for mobile edge networks using natural actor-critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr. 2019.

[14] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[15] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[16] P. Mach, and B. Zdenek, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tut.*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

[17] L. Yang, J. Cao, S. Tang, T. Li, and A. T. S. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *IEEE 5th Int. Conf. Cloud Comput.*, Honolulu, HI, USA, Jun. 2012, pp. 794–802.

[18] T. C. Chiu, W. H. Chung, A. C. Pang, Y. J. Yu, and P. H. Yen, "Ultra-low latency service provision in 5G fog-radio access networks," in *Proc. IEEE 27th Annu. Int. Symp. Personal, Indoor, Mobile Radio Commun.*, Valencia, Spain, Sep. 2016.

[19] A. C. Pang, W. H. Chung, T. C. Chiu, and J. Zhang, "Latency-driven cooperative task computing in multi-user fog-radio access networks," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, Atlanta, GA, Jul. 2017, pp. 615–624.

[20] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019

[21] S. Bi and Y. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[22] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[23] X. Yang, X. Yu, H. Huang, and H. Zhu, "Energy efficiency based joint computation offloading and resource allocation in multi-access MEC systems," *IEEE Access.*, vol. 7, pp. 117054–117062, Aug. 2019.

[24] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.

[25] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[26] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[27] H. Guo, J. Liu, and J. Zhang, "Efficient computation offloading for multi-access edge computing in 5G hetnets," in *Proc. IEEE Int. Conf. Commun.*, Kansas City, MO, USA, 2018, pp. 1–6.

[28] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 235–250, Jan. 2020.

[29] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[30] Y. Wu, L. P. Qian, K. Ni, C. Zhang, and X. Shen, "Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading," *IEEE J. Sel. Topics Signal Process*, vol. 13, no. 3, pp. 392–407, Jun. 2019.

[31] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing system," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[32] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Barcelona, Spain, Jun. 2018, pp. 902–907.

[33] H. Zhang, W. Wu, C. Wang, M. Li, and R. Yang, "Deep reinforcement learning-based offloading decision optimization in mobile edge computing," *IEEE Wireless Commun. Netw. Conf.*, Marrakesh, Morocco, Oct. 2019, pp. 1–7.

[34] Y. Dai, D. Xu, S. Maharjan, Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Vehic. Tech.*, vol. 67, no. 12, pp. 12–25, Dec. 2018.

[35] P. Huang, Y. Wang, K. Wang, and Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE Trans. Cyber.*, vol. 50, no. 10, pp. 4228–4241, Oct. 2020.

[36] Q. Jia *et al.*, "Energy-efficient computation offloading in 5G cellular networks with edge computing and D2D communications," *IET Commun.*, vol. 13, no. 8, pp. 1122–1130, May. 2019.

[37] W. Fan, Y. Liu, B. Tang, F. Wu, and Z. Wang, "Computation offloading based on cooperations of mobile edge computing-enabled base stations," *IEEE Access*, vol. 6, pp. 22622–22633, Dec. 2017.

[38] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Distributed resource allocation and computation offloading in fog and cloud networks with non-orthogonal multiple access," *IEEE Trans. Vehic. Tech.*, vol. 67, no. 12, pp. 12137–12151, Dec. 2018.

[39] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 760–770, Jul./Sep. 2018.

[40] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. 16, no. 1, pp. 122–128, Jan. 1986.

[41] K. Deep, K. P. Singh, M. L. Kansal, and C. Mohan, "A real coded genetic algorithm for solving integer and mixed integer optimization problems," *Appl. Math. Comput.*, vol. 212, no. 2, pp. 505–518, Jun. 2009.

Huilin Li received the bachelor's degree from the University of Science and Technology Beijing, in 2018. Currently, he is working toward the M.S. degree from the School of Computer and Communication Engineering, University of Science and Technology Beijing. His current research area is artificial intelligence, and mobile edge computing.

Haitao Xu (Member, IEEE) received the B.S. degree in communication engineering from Sun Yat-Sen University, China, in 2007, the M.S. degree in communication system and signal processing from the University of Bristol, U.K., in 2009, and the Ph.D. degree in communication and information system from the University of Science and Technology Beijing, China, in 2014.

From 2014 to 2016, he was with the Department of Communication Engineering, University of Science and Technology Beijing, Beijing, China, as a Postdoc. Currently, he is an Associate Professor with the Department of Communication Engineering, University of Science and Technology Beijing, Beijing, China. His research interests include wireless resource allocation and management, wireless communications and networking, dynamic game and mean field game theory, big data analysis, and security. Dr. Xu has co-edited a book titled *Security in Cyberspace* and co-authored over 50 technical papers. He has been serving in the organization teams of some international conferences, like CCT2014, CCT2015.

Chengcheng Zhou received the master's degree from the School of Information Technology, University of Sydney in 2008. Currently, she is working toward the Ph.D. degree at the School of Computer and Communication Engineering, University of Science and Technology Beijing. Her current research areas include artificial intelligence, mobile edge computing, and pattern recognition.

Xing Lü received the Ph.D. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012. He is currently a Professor with the School of Science, Beijing Jiaotong University, China. His research interests include soliton theory, symbolic computation, and optical soliton communication.

Zhu Han (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively.

From 2000 to 2002, he was an R&D Engineer of JDSU, Germantown, Maryland. From 2003 to 2006, he was a Research Associate at the University of Maryland. From 2006 to 2008, he was an Assistant Professor at Boise State University, Idaho. Currently, he is a John and Rebecca Moores Professor in the Electrical and Computer Engineering Department as well as in the Computer Science Department at the University of Houston, Texas. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. Dr. Han received an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the Journal on Advances in Signal Processing in 2015, IEEE Leonard G. Abraham Prize in the field of Communications Systems (best paper award in IEEE JSAC) in 2016, and several best paper awards in IEEE conferences. He was an IEEE Communications Society Distinguished Lecturer from 2015-2018, AAAS Fellow since 2019 and ACM Distinguished Member since 2019. Dr. Han is 1% highly cited researcher since 2017 according to Web of Science.