J Grid Computing (2021) 19: 38
https://doi.org/10.1007/s10723-021-09578-8

Computing Offloading Strategy Using Improved Genetic Algorithm in Mobile Edge Computing System

Anqing Zhu · Youyun Wen

Received: 30 December 2020 / Accepted: 12 July 2021 / Published online: 10 August 2021

Abstract For the current research on computing offloading, most of them only considers the multi-user task offloading decision problem or only considers the wireless resource and computing resource allocation. They have failed to comprehensively consider the impact of offloading decision and resource allocation on computing offloading performance, and it is difficult to achieve efficient computing offloading. For this reason, this paper proposes an edge computing task offloading strategy based on improved genetic algorithm (IGA). First, the weighted sum of task execution delay and energy consumption is defined as the optimization function of total overhead. Besides, the paper comprehensively considers the impact of users' offloading decision, uplink power allocation related to task offloading and MEC computing resource allocation on system performance. Secondly, Genetic Algorithm (GA) is substituted to establish communication model, the offloading strategy is corresponding to the chromosome in algorithm and the gene is encoded by integer coding. Finally, IGA is used to solve the task to achieve efficient offloading. Among them, the use of integer coding, knowledge-based crossover and the mutation of population segmentation improves the optimization ability of this algorithm. Finally, experimental results show that the performance of IGA is the best, and the overall cost is about 52.7% of All-local algorithm and 28.8% of Full-edge algorithm.

Keywords Computing offloading · Mobile edge computing (MEC) · Improved genetic algorithm (IGA) · Computing resource · Task allocation · Offloading decision

A. Zhu (✉) · Y. Wen
Management School, Guangdong University of Foreign Studies South China Business College, Guangzhou 510000 Guangdong, China
e-mail: zhuaq18@126.com

1 Introduction

In recent years, the development of mobile terminal equipment technologies such as smart phones and tablet computers has promoted a large number of services and applications that require high-quality transmission rates and processing rates. This brings severe challenges to network service providers. Although the central processing unit, battery capacity and other software and hardware of smart phones continue to upgrade, they are still limited by physical design and cannot handle applications that require large-scale calculations in a short time. The opportunity and challenge have promoted the development of Mobile Cloud Computing (MCC), which allows mobile users to access and use cloud computing [1].

The MEC mode deploys distributed computing and cache resources at the edge of wireless access network close to mobile terminal equipment. This can effectively save network bandwidth transmitted to the central node, reduce the cross-congestion of data transmission and help solve the problem of intensive computing tasks requiring instant response. By offloading intensive tasks

to MEC servers, users use the server's computing resources to save energy, improve the endurance of devices and extend the battery life [2]. The MEC server can provide real-time information about the user's location and behavior, which helps to enable context-aware services [3, 4]. In addition, MEC can support wireless power transmission to mobile terminal equipment [5–7].

Computing task offloading is a commonly used implementation method for MEC systems. Computing offloading refers to the technology of migrating computing tasks of user equipment to MEC servers or core cloud data centers [8]. The offloading strategy is usually determined according to the computing resources of mobile user equipment, the available resources in MEC servers and cloud servers. For the current research on MEC computing offloading, most of literatures only considers the multi-user task offloading decision problem. Or they only consider the wireless resource and computing resource allocation in the process of computing offloading and fail to comprehensively consider the impact of offloading decision and resource allocation on computing offloading performance. In addition, some literatures have studied the joint optimization problem of task offloading and resource allocation. However, few literatures consider the problem of MEC server computing resource allocation, there are problems of single consideration and high computing complexity. To solve the above problems, an edge computing task offloading strategy based on IGA is proposed. The innovations of this paper are summarized as follows:

(1) Aiming at the joint optimization problem of multi-user task offloading and resource allocation in MEC single-cell scenario, this paper defines the weighted sum of task execution delay and energy consumption as the total overhead optimization function. It also considers the impact of offloading decisions, uplink power allocation related to task offloading and MEC computing resource allocation on system performance.
(2) In order to reduce the time complexity of genetic algorithm and adapt to the structured characteristics of our proposed problem task, this paper adopts genetic algorithm to substitute the established communication model. In this model, the offloading strategy corresponds to the chromosome in algorithm and the gene is encoded by integer coding. Each gene represents the execution position of a task in the workflow. The reciprocal of overall system overhead is selected as the fitness function, and genetic algorithm is improved according to the characteristics of this problem. These ensure the effectiveness of mutation and avoid invalid mutation to improve the local search ability of this algorithm.

2 Related Work

In recent years, many domestic and foreign scholars have conducted research on the joint optimization of wireless and computing resources in different application scenarios. Literature [9–11] mainly focused on single-user scenarios, combining the cycle frequency of mobile device CPU to perform computing tasks and transmission power required for offloading to study offloading strategies. Literature [9] proposed Lyapunov's dynamic computing task offloading strategy with low computing complexity. The user made a computing task offloading decision in each time slot. After the decision was made, when the computing task was executed locally on terminal devices, CPU cycles were allocated. The transmission power was allocated when the computing task was offloaded to MEC servers for execution. Literature [10] comprehensively considered the computing power of terminal equipment and MEC server and the characteristics of communication channel between mobile devices and MEC servers. It designed a computing task offloading strategy, minimized the task completion time delay, and used a one-dimensional search algorithm to solve it. Literature [11] proposed a computing task offloading strategy based on minimizing MD energy consumption under the condition of time delay constraints, and transformed the optimization problem into a Markov process solution. Two strategies were proposed in the paper, one of them was an online learning strategy, which dynamically adjusts the offloading strategy according to task calculation required by the application used by terminal devices. The other was an offline prediction strategy. The computing task offloading strategy was designed according to the relevant knowledge of the application used by terminal devices and communication channel conditions. Literature [12] mainly studied the balance between energy consumption of mobile terminal devices and calculation delay in a multi-user system. It minimized energy consumption and takes system stability as a constraint condition under the time-delay limitation, and used

Lyapunov optimization algorithm to determine the computing task offloading strategy online. In the literature [13], when considering the trade-off between energy consumption and time delay, the priority was determined by setting the weight value. At the same time, this literature considered the problem of computing task offloading strategies in a multi-channel environment. They not only judged whether to offload computing tasks to MEC servers according to the weight parameter, but also selected channels with the best communication quality for data transmission. Obviously, solving the optimal solution is an NP-hard problem in a multi-user scenario, it is difficult to find a centralized solution, and there is room for improvement. Therefore, the paper proposes an approximate algorithm based on this, approximates centralized and distributed.

The above studies all involve the problem of centralized resource allocation. Centralized computing offloading is generally managed by a unified central controller, which requires prior knowledge of the computing resources of each mobile device and node. Under this premise, a better offloading strategy for computing tasks is selected for scheduling. It generally involves solving Mixed Integer Nonlinear Programming (MINP) problems, which are complex and difficult to find the optimal solution. And most only give numerical results to illustrate performance. Since only the same MEC server can be shared, the expected delays spent by different users in the system required by edge computing are coupled. Therefore, directly optimizing the MINP problem is very complicated. At the same time, due to users are very smart and hope to maximize their own interests, it is not easy to directly control the offloading strategy. In addition, the centralized optimization problem requires direct control of the users' decision-making and revenue function. This is hard to come by, especially when users are motivated to cheat. Thus, some scholars turn to distributed resource allocation.

Literature [14] was based on game theory to achieve efficient offloading of MCC tasks. The decision-making problem of distributed computing offloading among multiple mobile users was abstracted into a distributed computing task offloading game. By analyzing the structural properties of this game, it was found that this game can always achieve Nash equilibrium. Thus, a distributed computing offloading algorithm was designed to achieve Nash equilibrium, and the efficiency ratio was quantified through a centralized optimization method. Literature [15] mainly solved the problem of realizing energy-saving computing offloading under the hard constraint of terminal device application completion time. For this reason, dynamic energy-saving computing offloading and resource scheduling strategies were proposed to save energy consumption and reduce application completion time. Literature [16] proposed a multi-dimensional optimization problem including the formulation of offloading strategies, load balancing and computing resource allocation. It minimized the weighted sum of total delay and energy consumption of all multi-smart mobile devices (SMD) in a multi-MEC server and SMD network, and performs power control. A low-complexity heuristic algorithm was used to obtain an offloading strategy while ensuring load balance among multiple MEC servers, and used Lagrangian dual decomposition method to solve the sub-problem of computing resource allocation. This method was effective in shortening the task processing delay, but it was not conducive to reducing system energy consumption.

However, most existing studies always assume that the computing power of MEC servers is unlimited. But due to MEC servers are located at the edge of network, their computing power should be limited, especially in a workload-intensive network. In this case, a suitable strategy is needed to control the users' offloading task and ensure the normal operation of system, considering the impact of offloading decision and resource allocation on computing offloading performance.

3 System Model

3.1 Network Model

Consider a single-cell-multi-user network model using MEC technology, as shown in Fig. 1. Among them, the MEC server is deployed near a Base Station (BS). The MEC server can handle computing tasks offloaded from users. In this way, each BS has a certain storage capacity and task operation processing capacity, and the BSs communicate with each other through wireless signals. When the base station handles the task, it adopts the dynamic voltage adjustment technology. The total number of users in the network is K, and the collection of all mobile users is represented as $\kappa = \{1, 2, \cdots, K\}$. Assume that each user k has a computing task $\varpi_k = \{b_k, s_k, T_k^{\max}\}$ to be executed, where b_k represents the amount of

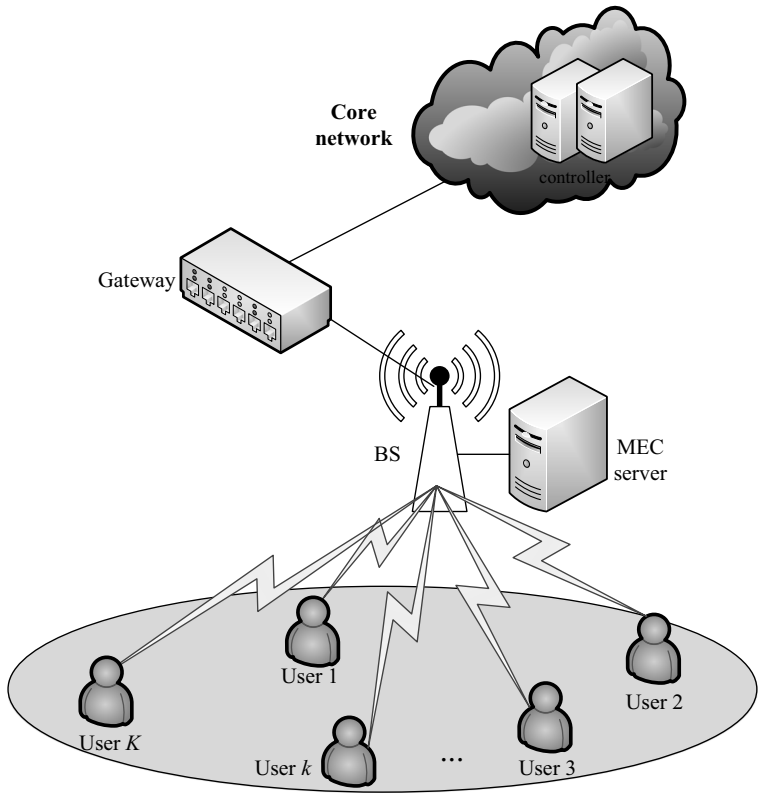

Fig. 1 The MEC network model

input data for computing tasks, s_k represents the number of CPU cycles required to complete computing tasks, and $T_k^{\max}$ represents the user's tolerance for processing tasks. This paper considers the total offloading scheme (that is, the task cannot be divided and can only be processed as a whole). Let $a_k \in \{0, 1\}$ denote the user's offloading decision. When $a_k = 1$ means that user k chooses to offload computing tasks to MEC servers for processing; otherwise, the user chooses to execute tasks locally. The set of users who choose to offload computing tasks is recorded as κ^c, and the potential of the set is $|\kappa^c| = K^c = \sum_{k=1}^{K} a_k$, which represents the number of users it contains. Similarly, the set of users who choose to execute locally is denoted as κ^l, and the potential of the set is $|\kappa^l| = K^l = K - K^c$. For the convenience of analysis, assume a quasi-static scenario, that is, within the same offloading decision period, the user set K remains unchanged.

3.2 Communication Model

In this model, it is assumed that each user in the cell adopts uplink transmission channels orthogonal to each other. Therefore, all users will not cause co-frequency interference between each other during task offloading. When user k transmits with p_k power, its uplink rate $R_k(p_k)$ is expressed as:

$$R_k(p_k) = W_k \log_2\left(1 + \frac{p_k h_k}{\sigma_{0,k}^2}\right), \forall k \in \kappa \tag{1}$$

where W_k represents the uplink bandwidth corresponding to user k, and $B = \sum_{k=1}^{K} W_k$ represents the total uplink bandwidth of the system. h_k represents the uplink channel gain between user k and BS, and $\sigma_{0,k}^2$ represents the uplink noise power of BS corresponding to user k.

3.3 Calculation Model

(1) When user k chooses to execute tasks locally, let f_k^l denote the computing power (the number of CPU cycles) that user k local device can provide. The time T_k^l to complete local computing tasks is:

$$T_k^l = \frac{s_k}{f_k^l} \tag{2}$$

It can be seen from literature [17] that the energy consumption E_k^l in local calculation is:

$$E_k^l = \kappa s_k \left(f_k^l\right)^2 \tag{3}$$

where the energy consumption coefficient κ is a constant related to the chip structure of mobile devices, here $\kappa = 10^{-26}$ is taken.

When user k chooses to execute computing tasks locally, its total cost includes the weighted sum of energy consumption generated in the process and local execution delay. According to formulas (2) and (3), it can be expressed as:

$$z_k^l = \gamma_k^e E_k^l + \gamma_k^t T_k^l \tag{4}$$

where the coefficients γ_k^e and γ_k^t respectively represent the trade-off factor of task execution energy consumption and time delay when making the offloading decision, satisfying $\gamma_k^e, \gamma_k^t \in [0, 1]$, $\gamma_k^e + \gamma_k^t = 1$. When γ_k^e is larger, it means that the power of user equipment is lower at this time. Pay more attention to user equipment energy consumption when making offload decisions. When the γ_k^t is large, it indicates that the computing task is delay-sensitive at this time. Pay more attention to task completion delay when making offloading decisions [18, 19]. Users can dynamically adjust according to their own situation.

(2) When user k chooses to offload tasks to MEC servers for execution, let $T_k^c(f_k, p_k)$ denote the processing delay of tasks on MEC servers corresponding to the user's remote end. It can be expressed as:

$$T_k^c(f_k, p_k) = T_k^{ul}(p_k) + T_k^{exe}(f_k) \tag{5}$$

where $T_k^{ul}(p_k)$ and $T_k^{exe}(f_k)$ respectively represent the time delay corresponding to the task input data uploaded to MEC servers via the uplink and the task execution on MEC servers. And there has:

$$T_k^{ul}(p_k) = \frac{b_k}{W_k \log_2(1 + \omega_k p_k)} \tag{6}$$

Among them, $\omega_k = h_k / \sigma_{0,k}^2$. After the task is uploaded to MEC servers, the MEC server will allocate computing resource f_k for it. At this time, the execution delay $T_k^{exe}(f_k)$ of tasks can be expressed as:

$$T_k^{exe}(f_k) = \frac{s_k}{f_k} \tag{7}$$

User k chooses to transfer computing tasks to the remote MEC server's energy consumption E_k^c expressed as:

$$E_k^c(p_k) = \frac{p_k}{\zeta} T_k^{ul}(p_k) = \frac{p_k}{\zeta} \frac{b_k}{W_k \log_2(1 + \omega_k p_k)} \tag{8}$$

Among them, ζ is the efficiency of equipment transmission power amplifier.

Based on the above evaluation indicators, when user k chooses to offload tasks to MEC servers for processing, the total cost includes energy consumption and execution delay of remote MEC servers. According to formulas (5)–(8), it can be expressed as follows:

$$z_k^c = \gamma_k^e E_k^c(p_k) + \gamma_k^t T_k^c(f_k, p_k) \tag{9}$$

In this chapter, the main consideration is the impact of task offloading on user-side delay and energy consumption. At the same time, MEC servers have more powerful computing power than the local equipment of users. Thus, the energy consumption when computing tasks are executed on MEC servers is omitted. Besides, the amount of result data after computing tasks are executed on MEC servers is generally small. Thus, the energy consumption and time delay that the user needs to bear in the process of returning the execution result of computing tasks to users are ignored.

3.4 Problem Description

Through the above analysis, the overhead function of user k in entire task offloading process can be expressed as:

$$z_k = (1 - a_k) z_k^l + a_k z_k^c \tag{10}$$

In this chapter, the optimal offloading decision $\mathbf{A}^* = \{a_1, a_2, \cdots, a_k\}$, uplink power allocation $\mathbf{P}^* = \{p_1, p_2, \cdots, p_k\}$ and MEC are calculated by minimizing the total overhead function of all users to calculate resource allocation strategy $\mathbf{F}^* = \{f_1, f_2, \cdots, f_k\}$. According to the above discussion, in the single-cell MEC scenario, the

optimized objective function of computing task offloading is expressed as:

$$\min_{A,P,F} Z = \sum_{k=1}^{K} (1-a_k) z_k^l + a_k z_k^c \; s.t.C1$$
$$: a_k \in \{0,1\}, \forall k \in \kappa C2 : 0 < p_k \leq p_{\max}, \forall k \in \kappa^c C3$$
$$: \sum_{k \in \kappa^c} a_k f_k \leq f_{\max} C4 : f_k > 0, \forall k \in \kappa^c C5$$
$$: \sum_{k \in \kappa} a_k W_k \leq B \quad (11)$$

where $C1$ represents the offloading decision of users; $C2$ represents that the uplink power when offloading transmission shall not exceed its maximum transmission power $p_{\max}$. $C3$ indicates that the computing resources allocated to the offloading user cannot exceed maximum computing resource $f_{\max}$ owned by MEC servers; $C4$ indicates that the computing resources allocated to offloading users by the MEC server are non-negative. $C5$ represents the limitation of system bandwidth, specifically $\sum_{k \in \kappa} a_k \leq \lfloor B/W_k \rfloor = N$, which means that only N users are allowed to upload data at the same time in the cell.

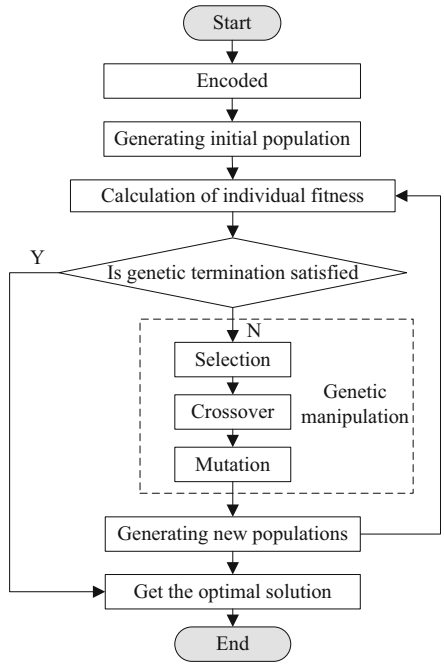

Fig. 2 The flow chart of genetic algorithm

4 Computing Offloading Strategy Based on IGA

4.1 Task Offloading Strategy Based on GA

In this section, we will solve the 0/1 knapsack problem based on genetic algorithm design, and get the best task offloading strategy. Genetic algorithm is influenced by biological genetics and evolution. By simulating the biological evolution process to search for the optimal solution in global scope using a probabilistic optimization method, it has the characteristics of direct, rapid, accurate and flexible [20, 21]. The flow chart of genetic algorithm is shown in Fig. 2 below. The genetic algorithm mainly simulates the following characteristics of biological evolution:

1. Genetic information is attached to chromosomes by coding, and evolutionary screening of chromosomes will affect coding information. Therefore, the nature of the problem solution can be expressed by coding.
2. Genetic information will generate new traits of chromosomes by genetic operations such as selection, crossover and mutation.
3. Genetic information will survive the fittest by natural selection. Thus, when designing genetic algorithms, individuals with strong adaptability will pass chromosomes to the next generation through fitness function screening [22, 23].

The following will introduce the key steps of genetic algorithm from chromosome coding, fitness function design and genetic operator design.

4.2 Computing Offloading Strategy Based on IGA

Classical genetic algorithms update individuals by selection, crossover and mutation operations, and obtain approximately optimal computing offloading strategies after several generations of evolution. Its convergence speed is slow and it is easy to fall into “premature maturity”. This paper proposes an IGA, which combines the knowledge-based crossover operator by improving the crossover and mutation process of classic genetic algorithms, to expand the range of feasible solutions and quickly generate the global optimal solution.

4.2.1 Initialization

The population consists of M chromosomes, and each chromosome corresponds to an offloading strategy. There are n genes on each chromosome and each gene corresponds to a task. Use integer coding to encode genes, and each gene represents the execution position of a task in the workflow. The coding example is shown in Fig. 1. Among them, $y_i = 0$ means that the tail task is executed locally, and $y_i = 1$ means that the first task executed is executed on the edge (Fig. 3).

4.2.2 Fitness Function Design

The fitness function is the criterion for evaluating the merits of individuals. The value of function shrinks, and the individual value is easily retained in evolution. The fitness function is calculated by formula (10). Due to time delay, the smaller the accumulation, the better the system performance. Thus, the reciprocal of the calculation result of formula (10) is taken as the fitness function. Besides, because workflow tasks are sensitive to delay, individuals who do not meet the delay constraint are deleted in the selection phase. Individuals that satisfy the time delay constraints constitute feasible solutions to the goal problem.

4.2.3 Selection

Use random tournament method to improve scale quality. Randomly select chromosomes into the replacement group, and select chromosomes with additional fitness function values to enter the chromosomes. The calculation expression for the probability of selecting chromosome $c(0 \le c \le M)$ into the increment by random tournament methods is

$$Q_c = \frac{F_c}{\sum_{c=1}^{M} F_c} \tag{12}$$

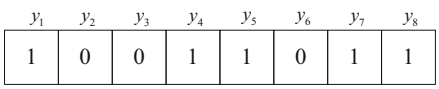

Fig. 3 The examples of chromosome coding

4.2.4 Cross

Crossover operation is an important operation in gene recombination. Through the crossover operation, part of the child chromosome replaces part of parent chromosome to form a new population individual. The crossover operation of classic genetic algorithms uses a single point crossover method, and its crossover probability is Q_c. The specific operation method is to randomly set a cross point in the chromosome string. When crossover is performed, the partial structures of the two chromosomes before or after the point are exchanged, and two new individuals are generated.

The crossover operator combines individuals in the selection phase and expects to produce high-quality offspring individuals. IGA usually uses the standard single point crossover operator. But when the standard single-point crossover operator is applied to a specific problem, the effect is generally poor. Thus, GA algorithm is improved according to the characteristics of problem, so that it has a knowledge-based crossover. The algorithm selects an individual from its parents R_1 and R_2 in the selection phase. And each gene of an individual is a more adaptive gene among the corresponding genes of its parents. By calculating the weighted combination of execution time and energy consumption, the local fitness of each gene can be compared:

$$f_k = \alpha_m * d_m^k + (1-\alpha_m) * e_m^k \tag{13}$$

where d_m^k and e_m^k respectively represent the delay cost and energy cost of k subtask. As shown in Fig. 4, the bolded genes mean higher local fitness. Thus, the offspring's genes include y_2, y_4 from R_1, and y_1, y_3, and y_5 from R_2.

4.2.5 Mutation

The mutation operation of classical genetic algorithms is to select a mutation bit for mutation. Use the mutation probability Q_c to change the value of a gene in chromosome to calculate the fitness of new chromosome. If the

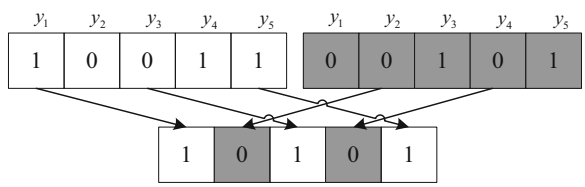

Fig. 4 The improved cross operation

fitness is less than original value, select and mutate again until chromosome fitness after mutation is greater than or equal to its original fitness.

Randomly select tasks in the task sequence as mutation points, and use mutation probability to determine whether to mutation. If a mutation occurs. The task execution position will change to any position except the current execution position. When the computing power is limited, users upload tasks to the edge and the cloud for execution by the computing offloading decision. The specific steps are as follows.

Step 1 Code the feasible offloading strategy that meets the requirements of maximum tolerable delay as a chromosome. Calculate individual fitness according to formula (10).

Step 2 Select the chromosome with smaller fitness function value to enter the population. Use NDX operator to perform crossover and mutation operations on individuals.

Step 3 According to the constraints, delete infeasible solutions in the population.

Step 4 In order to expand the space of feasible offloading strategy, repeat the above steps 2–3 continuously. When the number of iterations is greater than 200, or the fitness function value error of the chromosomes produced by two adjacent iterations is less than 2%, the above process is stopped. In order to ensure the minimum system overhead, the individual with the smallest fitness function value is selected from the last generation population to decode. The decoded offloading strategy is regarded as the optimal offloading strategy.

5 Experiment

5.1 Simulation Setting

In this section, we will evaluate the performance of our proposed algorithm by simulation and comparison experiments. Consider a single-cell-multi-user network using MEC, where the MEC server is co-located near BS, and users are evenly distributed in the cell coverage. Using Matlab2012a simulation. The channel model in this chapter refers to the 3GPP standard in [24]. The detailed simulation parameter settings are shown in Table 1.

Table 1 Simulation experiment parameter settings

Parameter	Value
Community coverage radius	600 m
User broadband /W_k	1 MHz
System bandwidth /B	16 MHz
The power of background noise /σ_0^2	-100 dBm
The maximum transmit power of users /$p_{\max}$	24 dBm
Input data size /B_k	500~1000 KB
The CPU cycle required for task completion /s_k	0.2~1GMHz
The maximum delay tolerated by tasks /$T_k^{\max}$	1~4 s
The computing power of users /f_k^l	0.1~1GMHz/cycle
The computing power of MEC servers /$f_{\max}$	4GMHz/cycle
The trade-off factor between energy consumption and delay /λ_k^c、λ_k^l	0.25~0.75
Population size /M	50
The maximum number of population evolution /$G_{\max}$	200

5.2 Performance Verification of Algorithm

In order to verify the performance of algorithm, our proposed IGA offloading strategy is compared with All-local where all tasks in the workflow are executed locally, Full-edge where all tasks in the workflow are all offloaded to the edge cloud, and the offloading strategy based on GA algorithm. The comparison indexes mainly differ in the number of users, the number of tasks, delay and energy consumption.

Figure 5 shows the simulation results of the four strategies with different user numbers overhead. We can see that as the number of users increases, the delay and energy consumption of the four strategies have increased. This is because the increase in the number of users leads to an increase in the number of requests, which leads to an increase in the delay and energy consumption of the algorithm. All-local strategy without offloading has the largest delay and energy consumption. When the number of users is in the range of 0–7, the energy consumption of Full-edge strategy is the smallest, and then the energy consumption of IGA algorithm strategy is the smallest. This is because when the number of users is small, offloading tasks to nearby edge servers can effectively reduce latency. As the number of users increases, edge servers are overloaded, which will inevitably lead to delays in decision-making.

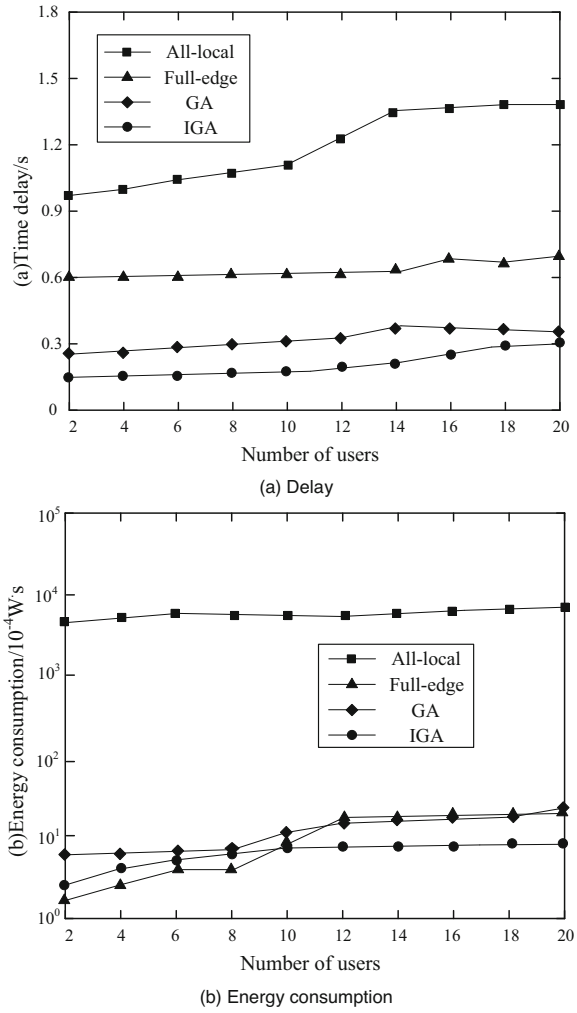

Fig. 5 The simulation results of four strategies with different number of users. (a) Delay, (b) Energy consumption

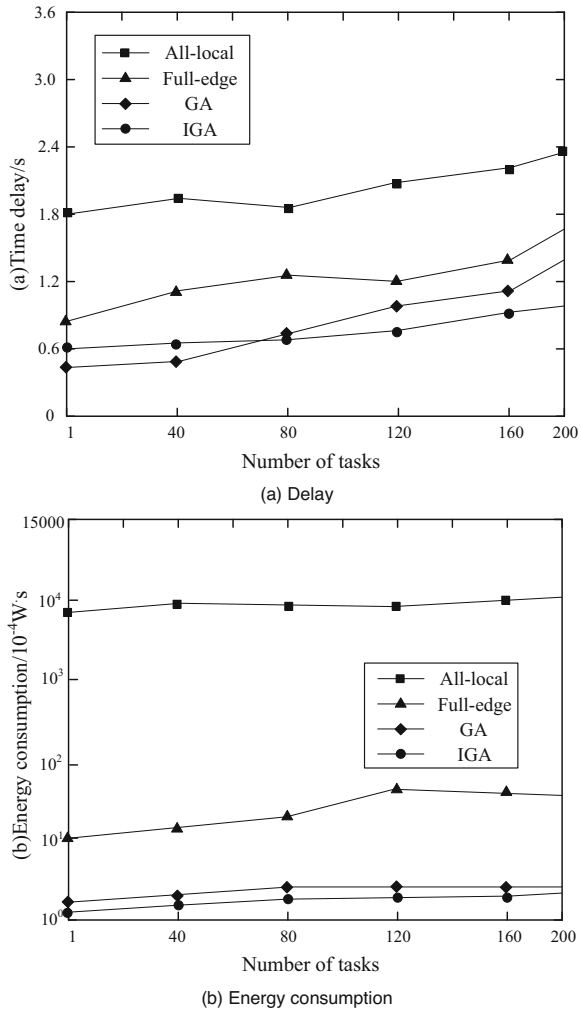

Fig. 6 The simulation results of four strategies with different number of tasks. (**a**) Delay, (**b**) Energy consumption

IGA strategy can quickly make offloading decisions and effectively optimize system overheads.

As the number of users increases, the number of users per CPU service increases. When the user's computing needs exceed the computing capacity of edge servers, queuing occurs, resulting in increased overhead. The overall cost of IGA strategy is about 28.8% of All-local and 52.7% of Full-edge.

Assuming that the number of tasks is uniformly distributed in the range of [0,200], the simulation results of the four strategies with different task number overheads are shown in Fig. 6. It can be seen that as the number of tasks increases, the costs of different strategies are increasing. Because the computing power of terminal equipment is limited, the channel frequency

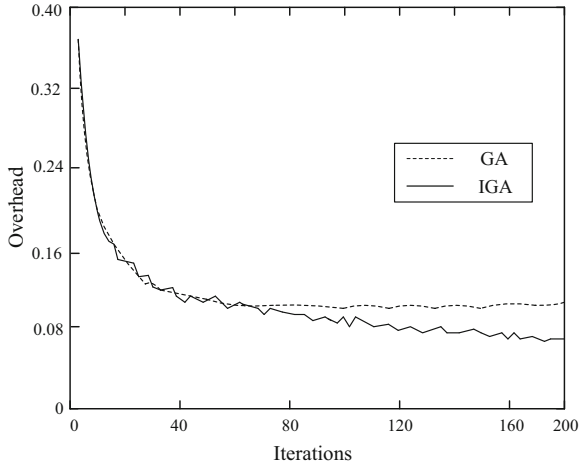

Fig. 7 The simulation results of iterative convergence of two strategies

band resources are limited. As the number of tasks increases, the load of computing equipment increases, and it is difficult for limited wireless resources to cope with the increase in the number of tasks. Because IGA strategy can quickly make offloading decisions and effectively optimize the overhead of system, the overhead of IGA strategy is minimal.

In order to expand the space of feasible offloading strategies, it is necessary to determine the number of iterations when using knowledge-based crossover operator to perform crossover and mutation operations on individuals. The offloading decision is made by fitting classic GA algorithm, and the convergence of IGA strategy and GA strategy is compared. The simulation result is shown in Fig. 7. It can be seen that as the number of iterations increases, the overhead of IGA algorithm is less than that of GA algorithm. GA algorithm converges at 100 iterations, and the cost of continuing iteration is no longer reduced. IGA algorithm gradually converges when the number of iterations reaches 200, and the cost of ownership continues to decrease.

5.3 Comparative Analysis with Advanced Algorithms

In order to further demonstrate the performance of our proposed algorithm in terms of energy consumption and delay, it is compared and analyzed with literature [9], literature [13] and literature [16]. The result is shown in Fig. 8.

It can be seen from Fig. 8 that as the number of requesting devices increases, the computing offloading and resource allocation strategies generated by various algorithms basically increase in terms of energy consumption and delay. The proposed algorithm can achieve better results in terms of delay and load balancing. But the performance is average in terms of energy consumption.

In addition, it can be seen from Fig. 8 that with the increase in the number of devices on request, the computing offloading and resource allocation strategies generated by the algorithm in literature [13] perform poorly in terms of load balancing. And it behaves mediocre in other respects. The algorithm in literature [16] performs better than the algorithm in literature [13] in all aspects except delay. The algorithm in literature [16] comprehensively considers the advantages of the two methods in various aspects, and the performance in terms of energy consumption and load balancing is relatively

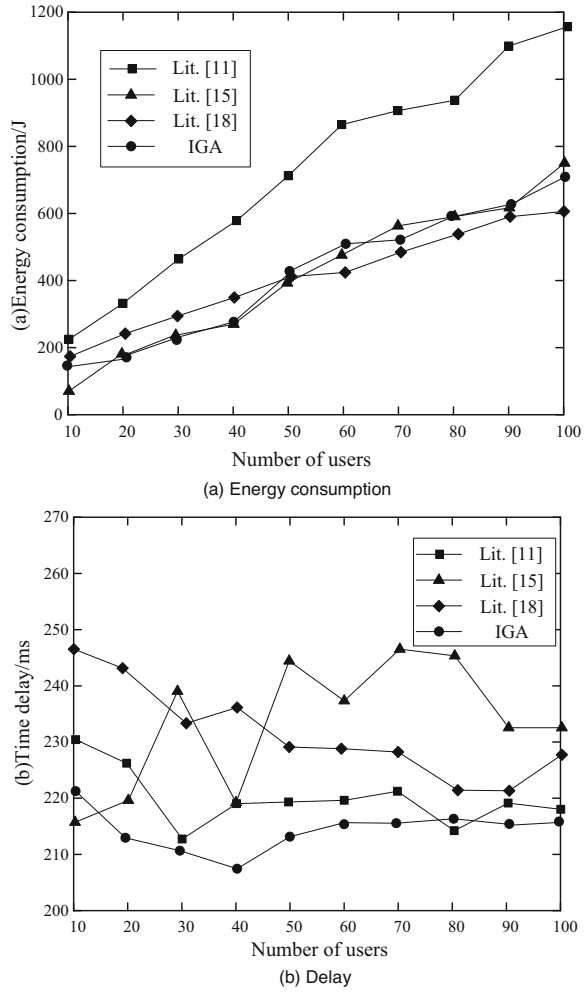

Fig. 8 Performance of each algorithm in task offloading. (a) Energy consumption, (b) Delay

ideal. When the number of devices is large, the strategy generated by the algorithm in literature [9] has a good effect in terms of delay. But the effect in terms of energy consumption is the worst. At the same time, because the algorithm in literature [16] and the algorithm in literature [13] are more inclined to offload subtasks to local devices. When tasks are offloaded to edge computing servers with higher computing power. The performance of the literature [16] algorithm is better than that of the literature [13] algorithm. When the number of applications is large, the time delay of the algorithm in literature [9] shows a downward trend, and its performance is better than the other two comparison algorithms. IGA combined the knowledge-based crossover operator to improve the crossover and mutation process of classic genetic algorithms, thereby expanding the range of

feasible solutions and quickly generating global optimal solutions.

6 Conclusion

This paper studies the problem of MEC computing offloading based on genetic algorithm in a single-cell-multi-user scenario. The total cost of users is minimized by formulating a joint optimization problem of task offloading decision and resource allocation. Classical genetic algorithms update individuals by selection, crossover and mutation operations, and obtain approximately optimal computing offloading strategies after several generations of evolution. However, its convergence rate is slow, and it is easy to fall into "premature maturity". This paper proposes an IGA combined with a knowledge-based crossover operator by improving the crossover and mutation process of classic genetic algorithms. This algorithm expands the range of feasible solutions and quickly produces global optimal solutions.

This paper has done relevant research on computing offloading and load balancing algorithms in MEC networks. However, due to personal research ability and time constraints, there are still some areas for improvement in this paper. The proposed algorithm can achieve better results in terms of delay and load balancing, but its performance in terms of energy consumption is average. The proposed task offloading algorithm provides a new solution for MEC computing offloading technology. However, this scheme mainly considers the calculation offloading under static conditions, and does not consider the impact of user mobility on system. Therefore, how to make real-time task offloading decisions and resource allocation plans based on the movement of users is challenging.

References

1. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges[J]. IEEE Internet Things J.. **3**(5), 637–646 (2016)
2. Chen, L., Zhou, S., Xu, J.: Computation Peer Offloading for Energy-Constrained Mobile Edge Computing in Small-Cell Networks[J]. IEEE/ACM Trans Networking. **26**(4), 1619–1632 (2018)
3. Yang, Y., Wang, K., Zhang, G., Chen, X., Luo, X., Zhou, M.T.: MEETS: maximal energy efficient task scheduling in homogeneous fog networks[J]. IEEE Internet Things J. **5**(5), 4076–4087 (2018)
4. Shi, B., Yang, J., Huang, Z., et al.: Offloading guidelines for augmented reality applications on wearable devices[C]// the 23rd ACM international conference. Brisbane. 1271–1274 (2015). https://doi.org/10.1145/2733373.2806402
5. Nunna, S., Kousaridas, A., Ibrahim, M., et al.: Enabling Real-Time Context-Aware Collaboration through 5G and Mobile Edge Computing[C], pp. 601–605. International Conference on Information Technology - New Generations, Las Vegas (2015)
6. Bi, S., Zhang, Y.J.: Computation rate maximization for wireless powered Mobile-edge computing with binary computation offloading[J]. IEEE Trans. Wirel. Commun. **17**(6), 4177–4190 (2018)
7. Hu, X., Wong, K., Yang, K.: Wireless powered cooperation-assisted Mobile edge computing[J]. IEEE Trans. Wirel. Commun. **17**(4), 2375–2388 (2018)
8. Wang, F., Xu, J., Wang, X., Cui, S.: Joint offloading and computing optimization in wireless powered Mobile - edge computing systems[J]. IEEE Trans. Wirel. Commun. **17**(3), 1784–1797 (2018)
9. Li, J., Lv, T.: Deep Neural Network Based Computational Resource Allocation for Mobile Edge Computing[C], pp. 1–6. IEEE Global Communications Conference (GLOBECOM), Abu Dhabi (2018)
10. Mao, Y., Zhang, J., Letaief, K.B.: Dynamic computation offloading for Mobile-edge computing with energy harvesting devices[J]. IEEE J. Select. Areas Commun. **34**(12), 3590–3605 (2016)
11. Liu, J., Mao, Y., Zhang, J., et al.: Delay-optimal computation task scheduling for mobile-edge computing systems[C], pp. 1451–1455. IEEE international symposium on information theory (ISIT), Barcelona (2016)
12. Kamoun, M., Labidi, W., Sarkiss, M.: Joint resource allocation and offloading strategies in cloud enabled cellular networks[C], pp. 5529–5534. IEEE international conference on communications (ICC), London (2015)
13. Mao, Y., Zhang, J., Song, S.H., et al.: Power-delay tradeoff in multi-user Mobile-edge computing systems[C], pp. 1–6. IEEE global communications conference (GLOBECOM), Washington (2016)
14. Chen, X., Jiao, L., Li, W., Fu, X.: Efficient multi-user computation offloading for Mobile-edge cloud computing[J]. IEEE/ACM Trans. Networking. **24**(5), 2795–2808 (2016)
15. Chen, X.: Decentralized computation offloading game for Mobile cloud computing[J]. IEEE Trans. Parallel Distrib. Syst. **26**(4), 974–983 (2015)
16. Shulei, L.I., Zhai, D., Pengfei, D.U., et al.: Energy-efficient task offloading, load balancing, and resource allocation in mobile edge computing enabled IoT networks[J]. ence China Inform. ences. **62**(002), 1–3 (2019)
17. Jiao, Z., Hu, X., Zhaolong, N., et al.: Energy-latency tradeoff for energy- aware offloading in Mobile edge computing networks[J]. IEEE Internet Things J. **5**(4), 2633–2645 (2018)
18. Ketyko, I., Kecskes, L., Nemes, C., et al.: Multi-User Computation Offloading as Multiple Knapsack Problem for 5G Mobile Edge Computing[C]// 2016 European Conference on Networks and Communications (Eu CNC), pp. 225–229. IEEE Press, Athens (2016)

19. Le, H.Q., Al-Shatri, H., Klein, A.: Efficient resource allocation in mobile-edge computation offloading: completion time minimization [C]// 2017 IEEE International Symposium on Information Theory (ISIT), pp. 2513–2517. IEEE Press, Aachen (2017)
20. Khair, U., Lestari, Y.D., Perdana, A., Hidayat, D., Budiman, A.: Genetic Algorithm Modification Analysis Of Mutation Operators, pp. 1–6. Max One Problem[C]// 2018 Third international conference on informatics and computing (ICIC), Palembang (2018)
21. Yiqiu, F., Xia, X., Junwei, G.: Cloud Computing Task Scheduling Algorithm Based On Improved Genetic Algorithm[C], vol. 2019, pp. 852–856. 2019 IEEE 3rd information technology, networking, electronic and automation control conference (ITNEC), Chengdu
22. Pyrih, Y., Kaidan, M., Tchaikovskyi, I., Pleskanka, M.: Research of Genetic Algorithms for Increasing the Efficiency of Data Routing[C], vol. 2019, pp. 157–160. 2019 3rd international conference on advanced information and communications technologies (AICT), Lviv
23. Li, T., Lei, G., Wan, F., Shu, Y.: Research on Intelligent Volume Algorithm Based on Improved Genetic Annealing Algorithm[C], vol. 2020, pp. 196–198. 2020 IEEE international conference on power, intelligent computing and systems (ICPICS), Shenyang
24. Evolved Universal Terrestrial Radio Acess(E-UTRA); Further Advancements for E-UTRA Physical Layer Aspects (Release 9), 3rd Generation Partnership Project 3GPP TS 36.814 (2012)