Computation Offloading Strategy for Improved Particle Swarm Optimization in Mobile Edge Computing

Shun Li
School of Electronic Engineering
Xi'an University of Posts and Telecommunication
Xi'an, Shaanxi, China
237666297@qq.com

Haibo Ge
School of Electronic Engineering
Xi'an University of Posts and Telecommunication
Xi'an, Shaanxi, China
gehaibo2417@aliyun.com

Xutao Chen
School of Electronic Engineering
Xi'an University of Posts and Telecommunication
Xi'an, Shaanxi, China
chenxutaoa@163.com

Linhuan Liu
School of Electronic Engineering
Xi'an University of Posts and Telecommunication
Xi'an, Shaanxi, China
1436944238@qq.com

Haiwen Gong
School of Electronic Engineering
Xi'an University of Posts and Telecommunication
Xi'an, Shaanxi, China
gheaven0201@163.com

Rui Tang
School of Electronic Engineering
Xi'an University of Posts and Telecommunication
Xi'an, Shaanxi, China
tr191109@163.com

2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA) | 978-1-6654-2311-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCBDA51879.2021.9442609

***Abstract*—Mobile Edge Computing (MEC) reduces latency and energy consumption by migrating computing resources to the edge of the network. Computing offloading is one of the means to reduce latency and energy consumption in MEC. Reasonable offloading decisions can effectively reduce system cost. Aiming at the increase in system delay and energy consumption caused by the deployment of the MEC server in the 5G communication scenario, a computing offloading strategy EIPSO based on an improved particle swarm optimization (PSO) algorithm is proposed. Establish a delay, energy consumption and multi-objective optimization model, and for delay-sensitive mobile applications, the model is transformed into a delay minimization problem under energy consumption constraints, and a penalty function is added to balance delay and energy consumption. Through the proposed calculation offloading decision, the calculation task is reasonably allocated to the corresponding MEC server. The simulation results show that compared with the ALL-Local algorithm, MECR algorithm and PSAO algorithm, the total system cost of this algorithm is the smallest, and the EIPSO strategy can reduce the delay in the MEC and balance the load of the MEC server.**

Keywords—mobile edge computing, computation offloading, particle swarm algorithm

I. Introduction

With the widespread popularity of mobile devices, new mobile applications such as computationally intensive and time-sensitive mobile applications continue to emerge and bring a good user experience to users. 5G has the characteristics of proximity, low latency, and high bandwidth. It uses 5G to connect a large number of intelligent terminal devices to make up for the shortcomings of traditional communication networks and realize timely data sharing and interaction [1]. Emerging 5G applications, such as virtual reality, image recognition, Internet of Vehicles, and online games, have higher requirements for time delay and energy consumption. However, it is difficult for mobile devices with limited resources to meet the needs of these emerging mobile applications [2]. Mobile Edge Computing (Mobile Edge Computing, MEC) transforms the end-cloud architecture into an end-side cloud architecture, which reduces latency to a certain extent, and improves the throughput of computing tasks through a reasonable computing offloading strategy, which can better meet User experience quality requirements. Computing offloading is one of MEC's research directions, offloading computing tasks to cloud servers or edge servers, so as to achieve the purpose of alleviating the limitations of computing and storage and reducing latency.

The current offloading strategy control methods are mainly centralized control and distributed cooperative control. Single-user single-MEC, multi-user single-MEC and multi-user multi-MEC are used in modeling scenarios, and the optimization goals mainly include minimization under energy consumption constraints. An offloading strategy that minimizes energy consumption and balances energy consumption and delay under delay and delay constraints. In the real communication scenario, the tasks to be uninstalled by numerous users and the number of edge servers require an optimal solution for the uninstallation strategy; the task requirements of users in the 5G communication environment (such as AR, VR, etc.) will have more stringent requirements for delay, So pay more attention to delay requirements.

This paper considers the computing offloading strategy in the 5G communication scenario. In the multi-user and multi-MEC scenario, the problem of minimizing the time delay under the energy consumption constraint is studied, and the offloading of high-calculation tasks to the MEC server is carried out through centralized control.

Computational offloading is an important research direction in MEC. A large number of researchers have conducted a lot of research on computational offloading in MEC and have made breakthrough progress. These studies mainly focus on unloading decisions to achieve the purpose of reducing time delay and saving equipment energy consumption. The current research on computing offloading strategies mainly optimizes the single goal of the system, aiming to reduce system delay or energy consumption. Literature [3] uses the method of queuing network to analyze the system delay, and proposes an algorithm with linear complexity, which can significantly reduce the system delay. Literature [4] proposes a fine-grained offloading strategy, which expresses the task offloading problem as a constrained 0-1 planning problem, and uses the BPSO algorithm to minimize energy consumption under delay constraints. Literature [5] proposed a dynamic unloading scheme, which regards the unloading task as a two-level stochastic optimization problem. Through the task queuing model, the mobile device decides whether the task is executed locally or on the MEC server, using Markov The chain decision process is used to deal with this problem, and an effective one-dimensional search algorithm is proposed to solve the power-constrained time delay problem. Literature [6] studied a MEC system with energy harvesting devices, and considered the factors of data channel transmission and CPU calculation during unloading, and proposed a dynamic computing unloading algorithm based on Lyapunov optimization to minimize the execution delay of the application. Although the literature [5-6] provides a solution to the problem of offloading a single task to the MEC, there are often multiple mobile devices that need to offload tasks in a real communication cell. At this time, a distributed strategy is needed to perform offloading. Literature [7] studies the trade-off between offload delay and reliability in the mobile edge computing environment, and uses three algorithms based on heuristic search, reconstruction linearization technology and semi-definite relaxation to achieve system delay and offload failure. The rate is the smallest. [8], [9] Use game theory's distributed algorithm offloading strategy to optimize the single goal of delay or energy consumption. Literature [10] proposed a fast hybrid multi-site computing offloading solution based on text computing, which can achieve the best and near-optimal offloading partition according to the size of the mobile application. When the scale is expanded, the paper proposes particle swarm optimization to obtain Get a better uninstall optimization solution. Aiming at the multi-objective optimization problem, literature [11] designed a task offloading strategy based on genetic algorithm, which converges after about 100 iterations and reduces the total overhead of the system. However, the traditional genetic algorithm will have the problem of premature and premature convergence, and the feasible solution range is small, which makes it impossible to find the optimal strategy.

The above research work on the computing offloading problem in the mobile edge computing environment, although the proposed system or algorithm has better performance in terms of energy consumption and delay optimization, it does not consider the mobile in the communication cell in the actual 5G scenario. The user multitasks to calculate the goal of offloading low latency within the acceptable range of energy consumption. Therefore, it is necessary to propose a complete offloading strategy that meets the 5G scenario of multi-user and multi-MEC and minimizes the delay under the constraints of energy consumption. This paper proposes the EIPSO computing offloading algorithm, the main work is as follows:

(1) In the real 5G scenario, a time delay and energy consumption model is constructed, and the computational offloading problem is transformed into a model of minimizing time delay under energy consumption constraints in the MEC system.

(2) In order to avoid the defects of premature convergence or prematurity of traditional intelligent algorithms and improve the algorithm's global search ability, this paper improves the particle swarm optimization (PSO) and designs an improved optimization algorithm based on particle swarm (EIPSO).

(3) The unloading strategy proposed by the algorithm in this paper is compared with the local unloading strategy, the MEC benchmark unloading strategy, and the BPSO unloading strategy through simulation experiments. The computational offloading strategy proposed in this paper can achieve a relative balance between energy consumption and time delay.

The rest of this article is composed as follows. In the second part, the system model and problem formulation are given, including system model, time delay model, energy consumption model, and calculation model. The third part describes the implementation steps of the optimized computational offloading algorithm based on the improved particle swarm optimization. The fourth part gives the simulation results and analyzes the results. The last part summarizes this article.

II. System Model and Problem Formulation

A. Network Model

The deployment scenario of the MEC system is shown in Fig. 1. The MEC server is mainly aimed at delay-sensitive tasks and tasks with a large amount of calculation. Local equipment (smart equipment, industrial equipment, etc.) mainly performs preliminary data processing. When disconnected from the MEC server, intelligence depends on its own capabilities to handle simple tasks. When performing delay-sensitive and computationally intensive tasks, it needs to The task is transferred to the MEC server for calculation. In this article, the MEC server is deployed on the side of the microcell base station close to the user, and the user uploads the task to the edge server wirelessly, and the edge server returns the calculation result to the user after performing the calculation.

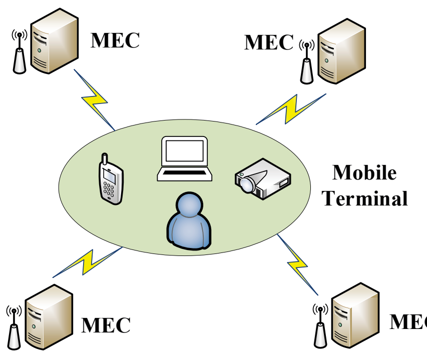

Fig. 1. network model.

Assume that the current scenario includes K mobile devices and N MEC servers. After each mobile device generates a task, it is executed by a server or a local server, and finally the optimal execution position of all tasks is found, that is, the offload vector V is calculated. Among them, the task is executed locally or in any one of the N MEC servers.

B. Time Delay Model

Depending on the task running on different devices, the delay model will also be different: if the current task is executed on the local device and each task has no queuing delay constraint, only the local computing delay needs to be considered at this time; if the task is offloaded to When the MEC server executes, the delay required to complete the current task will increase, including transmission delay, MEC calculation delay, and feedback delay. However, the difference between the feedback delay and the transmission delay is a large order of magnitude, so in this article Ignore the feedback delay.

1) Local calculation delay

$$T_l = B_i * \frac{f_i}{f_{u,i}} \tag{1}$$

Among them, B_i represents the amount of data used to process the task, f_i represents how many clock cycles are required for the processing of each bit of data, $f_{u,i}$ represents the CPU cycle frequency of the local device (cycles/second), so it can also be concluded that $B_i * f_i$ represents the current The amount of calculation of the task. In formula (1), the local calculation delay (T_l) is proportional to the reciprocal of the local CPU cycle frequency ($f_{u,i}$).

2) Transmission delay

For the transmission delay of MEC, the transmission rate between channels needs to be defined first. According to Shannon's definition, we can get:

$$r_{i,j} = W * lb\left(1 + \frac{P_i * H_{i,j}}{W * N_0}\right) \tag{2}$$

Among them, $r_{i,j}$ is the transmission rate from the local device i to the j-th MEC server, W is the transmission bandwidth, P_i is the transmit power of a single local device, $H_{i,j}$ is the channel gain, and N_0 is the noise power spectral density.Formula (2) is the transmission rate of the current task in the channel, $B_i * f_i$ is the calculation amount of the current task, so the transmission delay is:

$$T_{t,i,j} = B_i * f_i / r_{i,j} \tag{3}$$

3) MEC calculation delay

$$T_{s,i} = B_i * f_i / f_{s,i} \tag{4}$$

Among them, $f_{s,i}$ represents the MEC server CPU clock frequency, and in formula (4), it can be seen that the MEC calculation delay is proportional to the reciprocal of the MEC server cycle frequency.

C. Energy Consumption Model

1) *Calculate energy consumption:* The computing power of the CPU is closely connected with its internal chip architecture. Each generation of the CPU architecture has a fixed energy consumption ratio. Under the same architecture, the power consumption is proportional to the voltage and frequency, and the energy consumption between devices is also Will be affected by the amount of calculation tasks, so the transmission energy consumption of the i-th task is as in formula (5):

$$E_i = C_i * L^2 * f_{u,i} * B_i * f_i \tag{5}$$

Among them, the C_i capacitance depends on the effective switching capacitance, and L represents the voltage.

2) *Transmission energy consumption*: Transmission energy consumption is targeted at tasks that are offloaded to the MEC server. In the delay model and the calculation of the transmission delay, the transmission energy consumption is:

$$E_{t,i,j} = P_i * T_{t,i,j} \tag{6}$$

D. Calculation Model

The total delay to complete an offloading task is the sum of the transmission delay and the server calculation delay:

$$T_i = T_{t,i,j} + T_{s,i} \tag{7}$$

The total delay for the local device to perform a task is:

$$T_i = T_l \tag{8}$$

From equations (7) and (8), we can get two different solution formulas for a calculation task in the calculation of the local equipment and the calculation in the MEC server. Therefore, the current problem is jointly expressed as:

$$P1: \min_{v_i} \sum_{j=1}^{n} \sum_{i=1}^{k} T_i \tag{9}$$

$s.t.$

$$E_i < E_{\max} \tag{10}$$

$$0 \le P_i \le P_{\max} \tag{11}$$

$$0 \le f_{u,i} \le f_{u,\max} \tag{12}$$

$$0 \le f_{s,i} \le f_{s,\max} \tag{13}$$

Among them, equation (9) represents finding the offloading decision vector that minimizes the time delay, and v_i represents the server number to which task i is specifically allocated; equation (10) represents the energy consumption constraint, and the average energy consumption of each task is less than the maximum energy Consumption; Formula (11) indicates that the CPU cycle frequency of the local device is less than the maximum CPU cycle frequency; Formula (12) indicates that the CPU cycle frequency of the MEC server is less than the maximum CPU cycle frequency. $P1$ pays more attention to the problem of delay in the entire uninstall process.

In addition, if one of all MEC servers has better performance, according to the modeling method of equation (9), because queuing delay is not considered, most tasks will be offloaded to the MEC server with better performance. It will cause its average power consumption to be higher than the maximum energy consumption, so offloading the current task to other MEC servers can alleviate the load pressure. Therefore, the problem P1 is transformed into the P2 problem of increasing the penalty function: when the average equipment energy is greater than the maximum energy consumption, the total system cost of offloading tasks to the current MEC server is increased by increasing the penalty function form.

$$P2: \min_{v_i} \sum_{j=1}^{n}\sum_{i=1}^{k} T_i + penalty(V) \tag{14}$$

$$penalty(V) = g * \sum_{j=1}^{n}\sum_{i=1}^{k} (E_i - E_{\max}) \tag{15}$$

s.t.

$$0 \le P_i \le P_{\max}$$
$$0 \le f_{u,i} \le f_{u,\max}$$
$$0 \le f_{s,i} \le f_{s,\max}$$

V represents the unloading vector that needs to be solved; g represents the penalty coefficient, as the calculation consumes more energy, the greater the penalty term, if it does not exceed the maximum energy consumption, no penalty. The meaning of increasing the penalty function is to balance energy consumption and time. Extension. Problem P2 pays more attention to the balance of delay and energy consumption during the entire unloading process.

III. Problem Solution

The biggest disadvantage of the PSO algorithm is that it is easy to fall into a local optimal solution, which leads to high task delay and energy cost. In order to solve this problem, this paper proposes an improved optimization algorithm based on particle swarm (EIPSO). First, the PSO algorithm is improved. By improving the fixed inertia weight in the standard particle swarm into an adaptive inertia weight, it can avoid falling into the local optimal solution to a certain extent, and it can better balance the local search and the global search, and improve the global optimization ability. , Which in turn reduces the energy consumption to meet the delay requirements.

Assuming that the size of the particle swarm is U, K represents the current number of local tasks, N represents the number of server MECs, and the set of all tasks can be expressed as $M = \{m_1, m_2, ..., m_k\}$. First, the task is quantitatively described. It is assumed that the mobile device numbered i generates $m_i(B_i, f_i, E_{\max})$ tasks to be calculated. Among them: B_i represents the amount of input data, f_i represents the calculation density, and $E_{\max}$ represents the energy consumption constraint of the task. The periodic frequency set $S = \{s_1, s_2, ..., s_n\}$ of all MEC servers includes the periodic frequencies of all MEC servers in the current system.

$$H = \begin{bmatrix} H_{1,1} & H_{1,2} & \cdots & H_{1,n} \\ H_{2,1} & H_{2,2} & \cdots & H_{2,n} \\ \vdots & \vdots & & \vdots \\ H_{K,1} & H_{K,2} & \cdots & H_{K,n} \end{bmatrix}$$

Use the channel gain matrix H to represent the maximum transmission rate of the computing task transmitted to the MEC server, the channel gain of $H_{i,j}$ is 0, and $H_{i,j}(1 \le i \le K, 1 \le j \le N, i \ne j)$ represents the channel gain required for the task generated by the mobile device i to be transmitted to the server numbered j.

Individual particles are coded by integers, and the element of each particle can be any integer between 1 and N. The dimension of the particle code is the same as the number of task sets. Assuming that the current task set contains 4 tasks $M = \{m_1, m_2, m_3, m_4\}$, the particle code is $[3,0,4,1]$ indicating the execution position of the task, and the first element 3 indicates that task m_1 is executed on the MEC server numbered 3. The second element 0 indicates that task m_2 is directly calculated locally, and the third element 4 indicates that task m_3 is executed on the MEC server numbered 4, and so on.

The particles represent the specific MEC server to which all current tasks will be offloaded. The unloading decision vector $V = \{v_1, v_2, ..., v_k\}$ of each particle represents the optimal execution position of all tasks. Among them, vi is randomly selected from 0 to N. When $v_i = 0$, the current tasks are executed locally; $v_i = j(1 \le j \le N)$ means that the current task is offloaded to the j -th MEC server for execution.

The particle speed $X_i = \{x_1, x_2, ..., x_k\}$ indicates the trend of task distribution to other MEC servers, and it is rounded up during the initialization and update process. Assuming that the task set contains 4 tasks $M = \{m_1, m_2, m_3, m_4\}$, the particle velocity is $[3,0,4,1]$, where the first element 3 means that the task m_1 is processed by the MEC server which is shifted 3 bits behind, and the second task is processed by The original MEC server processes, and so on.

The update equations of particle velocity and position are as follows:

p_{ij} represents the local optimal position of each particle i, and p_{gi} represents the global optimal position found so far by the overall particle.

$$x_{ij} = \omega \cdot x_{ij} + \theta_1 r_1 \lfloor p_{ij} - v_{ij} \rfloor + \theta_2 r_2 \lfloor p_{gi} - v_{ij} \rfloor \tag{16}$$

$$v_{ij} = v_{ij} + x_{ij} \tag{17}$$

Among them, r_1 and r_2 are two uniform random numbers [0,1], θ_1 and θ_2 are learning factors, also called acceleration constants, and ω is the inertia weight.

This article takes the total energy consumed when the task is completed as the goal, and the specific formula of the fitness function is as follows:

$$fitness(V) = \sum_{j=1}^{n}\sum_{i=1}^{k} T_i + penalty(V) = \sum_{j=1}^{n}\sum_{i=1}^{k} T_i + g * \sum_{j=1}^{n}\sum_{i=1}^{k} E_i - E_{\max} \tag{18}$$

In the PSO algorithm, by comparing the fitness of each particle with the global optimal value, the position state of the particle can be described more accurately. The calculation method of the success value is as follows:

$$S(i,t)=\begin{cases}1, fitness(p_i^t)>fitness(p_i^{t-1})\wedge fitness(p_i^t)>fitness(p_g^t)\\ 0.7, fitness(p_i^t)>fitness(p_i^{t-1})\wedge fitness(p_i^t)<fitness(p_g^t)\\ 0.3, fitness(p_i^t)>fitness(p_i^{t-1})\wedge fitness(p_i^t)=fitness(p_g^t)\\ 0, fitness(p_i^t)=fitness(p_i^{t-1})\end{cases} \quad (19)$$

Among them, $S(i,t)$ is the success value of particle i in the t-th iteration, p_i^t is the optimal position of particle i in the t-th iteration, p_g^t is the global optimal value, and fitness(p_i^t) is the particle i .The fitness of the optimal position in the t-th iteration, fitness (p_g^t) is the fitness of the global optimal position. If the current optimal position fitness value is greater than the last iteration optimal position fitness value and greater than the global optimal position fitness value, the success value is set to 1; if the current position fitness value is greater than the last iteration optimal position fitness value If the current location fitness value is greater than the optimal location fitness value of the last iteration and equal to the global optimal location fitness value, it is better to set the success value to 0.7 through statistical experiments. The success value is set to 0.3. Similarly, 0.3 is also the experimental value; if the current optimal position fitness value is equal to the optimal position fitness value of the previous iteration, the success value is set to 0. This calculation method obtains a more accurate success value by fine-graining the particle state, further improves the success rate of the particle, thereby improving the self-adaptability of the inertia weight ω , and effectively avoiding the particle from falling into the local optimum prematurely in the optimization process.

$\nabla_s(t)$ is the success rate of the particle swarm in the t-th iteration, which means that the position of this iteration is higher than that of the last particle in the particle swarm. The calculation formula for the success rate is as follows:

$$\nabla_s(t)=\frac{\sum_{i=1}^{n}S(i,t)}{n} \quad (20)$$

$\sum_{i=1}^{n}S(i,t)$ represents the sum of the success values of all particles, and n represents the number of particles in the entire particle swarm. Is the inertia weight at the t-th iteration, $\omega(t)$ is used to adjust the particle speed at each iteration.

$$\omega(t)=(\omega_{\max}-\omega_{\min})\nabla_s(t)+\omega_{\min}, 0\le\omega_{\min}\le\omega_{\max} \quad (21)$$

Algorithm 1 EIPSO

```
1: Integer encoding of particles
2: Randomly initialize the position vi and velocity xi of the particles
3: Calculate the fitness of each particle according to formula (16)
4: Initialize particle optimal allocation and global optimal allocation: $p_{ij}$ and $p_{gi}$
5: Number of iterations t=0
6: While $t \le 100$
7: Update the speed and position according to formulas (38) and (39)
8: Update particle optimal allocation, global optimal allocation and total system cost (fitness)
9: $t = t+1$
10: End while
11: Output $p_{gi}(t)$ and fitness value
12: End
```

IV. Simulation Results

A. Simulation Parameter Setting

In the 5G communication cell scenario, the number of devices $K=[50,100,150,200,250]$ the number of MEC servers are 10, and K mobile devices respectively initiate uninstallation requests to 10 servers. The value of the main parameters used in the simulation experiment and the meaning of the symbols are shown in Table I below.

TABLE I. Main Parameter Values

Parameter	Value
The amount of data processed for the task B_i	10~45Mb
The clock period required for each bit of data f_i	800~1200cycle/b
CPU cycle frequency of the local device $f_{u,i}$	2~4 GHz
CPU clock frequency of MEC server $f_{s,i}$	4~8 GHz
Transmission bandwidth W	1MHz
Local device transmit power P_i	100~400mW
Channel gain $H_{i,j}$	$2\times10^{-10}\sim2\times10^{-6}$
Noise average power $W*N_0$	1×10^{-9} W
Maximum power consumption constraint $E_{\max}$	1000J
Local device CPU coefficient k_u	10^{-24}
MEC server CPU coefficient k_s	10^{-26}
Penalty factor g	$10^{-2.5}$
Learning factor c_1 , c_2	1.5

B. Result Analysis

In this section, simulation comparison experiments are carried out on the EIPSO algorithm, ALL-Local algorithm, MEC-R algorithm, and PSAO algorithm in this article.

ALL-Lcoal algorithm: All tasks are executed on the local device.

MEC-R (MEC-Random) algorithm: Randomly assign the execution position of the task between the local equipment and the MEC server.

PSAO algorithm: a computational offloading algorithm based on particle swarm optimization (PSO).

Figure 2 shows the total system cost of different offloading strategies (including the average total delay of the penalty factor), the number of simulated devices $K = 50$, the MEC server is set to 10, and the data volume of each task $B_i = 4$Mb.As shown in Figure 2, as the energy consumption constraints increase, the total cost of the system gradually decreases. Compared with ALL-Local algorithm, MEC-R algorithm, PSAO algorithm, the total system cost of EIPSO algorithm is the smallest. It can be observed from Figure 2 that when the energy consumption constraint is 0, the total system cost of the ALL-Local algorithm is much higher than the other three strategies, because the local equipment processing tasks will produce higher energy consumption. As the energy consumption constraints increase, the model's sensitivity to energy consumption constraints decreases, so when the energy consumption constraint is 800J, the total cost of the ALL-Local strategy starts to be lower than the MEC-R offloading strategy. The total system cost of PSAO and EIPSO offloading strategies is significantly better than that of ALL-Local and MEC-R offloading strategies, but when the energy consumption constraint is 800J, the effect of the two strategies is not very different.

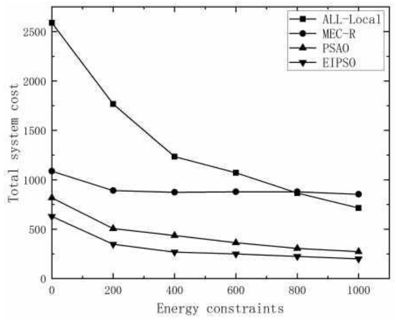

Fig. 2. The impact of energy consumption on the total cost of the system.

Figure 3 shows the performance comparison of the ALL-Local algorithm, MEC-R algorithm, PSAO algorithm, and EIPSO algorithm in the case of different device numbers. The simulated device numbers are 50, 100, 150, 200, and 250 respectively. It can be observed that as the number of devices increases, the total cost of the system gradually increases. This is because the increase in equipment will lead to an increase in the number of tasks to be processed, and the processing time and transmission time will also increase. When the number of devices does not exceed 200, the two strategies of PSAO and GIP are not much different; but when the number of devices exceeds 200 or more, the total system cost of PSAO suddenly increases, which is much higher than the EIPSO uninstall strategy. It can be seen from the figure that the total system cost of the ALL-Local uninstall strategy and the MEC-R uninstall strategy is much higher than the total system cost of the EIPSO uninstall strategy.

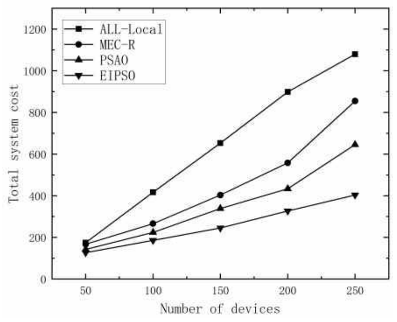

Fig. 3. The impact of the number of equipment on the total cost of the system.

Figure 4 shows the comparison of the total system cost of the ALL-Local algorithm, MEC-Random algorithm, PSAO algorithm, and EIPSO algorithm offloading strategies under different tasks. Take the number of tasks K=50 and the number of MEC servers N=10, as shown in Figure 4. As the number of tasks increases, the total system cost gradually increases. The ALL-Local uninstall strategy is to completely uninstall tasks locally. The increase in the amount of tasks will directly lead to an increase in time and energy consumption. Therefore, the ALL-Local uninstall strategy increases the total system cost much faster than the other three uninstall strategies. When the task volume of PSAO offloading strategy is less than 15Mb, the total cost of the system differs very little from the EIPSO offloading strategy; but when the task volume exceeds 15Mb, the total cost of the system increases greatly, and the offloading effect is not ideal. In the EIPSO uninstall strategy, the total system cost is better than the other three uninstall strategies, and it shows the best effect when the task volume gradually increases.

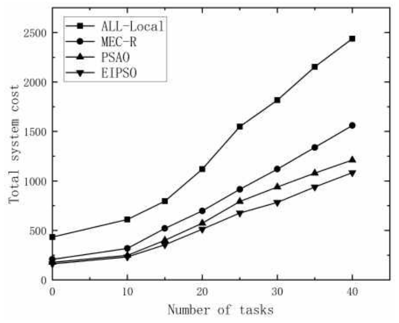

Fig. 4. The impact of the amount of tasks on the total cost of the system.

Figure 5 shows the effect of increasing the number of iterations on the total cost of the system. Take the number of tasks K=60. As the number of iterations increases, the total system cost of the MEC-Random algorithm, PSAO algorithm, and EIPSO algorithm gradually decreases. The ALL-Local algorithm is completely uninstalled locally, and the number of iterations has no effect on the experimental results of the algorithm. influences. When the number of iterations is [10, 100], the total cost of the system has a large downward trend, and the number of iterations exceeds 50 times, and the change trend of the total cost of the system is small and tends to be stable. Therefore, when the number of iterations is 50, better experimental results can be obtained.

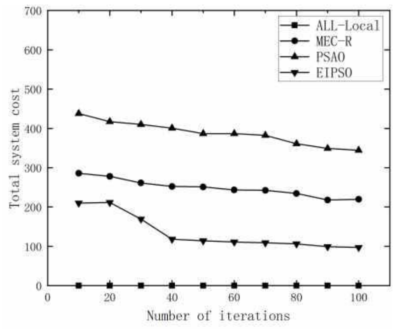

Fig. 5. The effect of the number of iterations on the total cost of the system.

In summary, through simulation analysis, it can be concluded that, compared with the ALL-Local uninstall strategy, MEC-Random uninstall strategy, and PSAO uninstall strategy, the EIPSO uninstall strategy proposed in this paper can all obtain the smallest total system cost.

V. Conclusion

In this paper, the user computing task offloading problem in the communication cell in the 5G reality scene is modeled as a multi-user and multi-MEC experimental optimization problem. In order to solve the problem of reducing the time delay through energy consumption of local mobile devices, the penalty function is used to balance the delay and energy consumption. , And proposed an unloading strategy EIPSO based on PSO algorithm optimization. After analyzing the results, it can be seen that the EIPSO offloading strategy proposed in this paper allows the local equipment and the MEC server to perform collaborative calculations. The original total system cost is reduced by 16%. With the increase in the number of devices and tasks, the total system cost is lower than the other three types of offloading. Strategies to meet the quality of service requirements of delay-sensitive applications. Future work will consider joint cloud computing, and continue computing offloading cloud servers and multiple edge servers.

References

[1] Xu X , Li D , Dai Z , et al. A Heuristic Offloading Method for Deep Learning Edge Services in 5G Networks[J]. IEEE Access, 2019, PP(99):1-1.

[2] Y. Qi, L. Tian, Y. Zhou and J. Yuan, "Mobile Edge Computing-Assisted Admission Control in Vehicular Networks: The Convergence of Communication and Computation," in IEEE Vehicular Technology Magazine, vol. 14, no. 1, pp. 37-44, March 2019,

[3] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin and X. Shen, "Cost-efficient workload scheduling in Cloud Assisted Mobile Edge Computing," 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS), Vilanova i la Geltrú, Spain, 2017, pp. 1-10,

[4] Maofei Deng, Hui Tian and Bo Fan, "Fine-granularity based application offloading policy in cloud-enhanced small cell networks," 2016 IEEE International Conference on Communications Workshops (ICC), Kuala Lumpur, Malaysia, 2016, pp. 638-643,

[5] J. Liu, Y. Mao, J. Zhang and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, 2016, pp. 1451-1455, doi: 10.1109/ISIT.2016.7541539.

[6] Y. Mao, J. Zhang and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," in IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3590-3605, Dec. 2016, doi: 10.1109/JSAC.2016.2611964.

[7] J. Liu and Q. Zhang, "Offloading Schemes in Mobile Edge Computing for Ultra-Reliable Low Latency Communications," in IEEE Access, vol. 6, pp. 12825-12837, 2018, doi: 10.1109/ACCESS.2018.2800032.

[8] X. Chen, L. Jiao, W. Li and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," in IEEE/ACM Transactions on Networking, vol. 24, no. 5, pp. 2795-2808, October 2016,

[9] M. Chen, M. Dong and B. Liang, "Multi-user Mobile Cloud Offloading Game with Computing Access Point," 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), Pisa, Italy, 2016, pp. 64-69,

[10] Goudarzi M , Zamani M , Haghighat A T . A Fast Hybrid Multi-site Computation Offloading for Mobile Cloud Computing[J]. Journal of Network and Computer Applications, 2016, 80(FEB.):219-231.

[11] 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," in IEEE Transactions on Vehicular Technology, vol. 66, no. 8, pp. 7432-7445, Aug. 2017.