Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 9319136, 9 pages
https://doi.org/10.1155/2022/9319136

Research Article

Computation Offloading Strategy for IoT Using Improved Particle Swarm Algorithm in Edge Computing

Aichuan Li,[1] **Lin Li**,[1] **and Shujuan Yi**[2]

[1]*College of Information and Electrical Engineering, Heilongjiang Bayi Agricultural University, Daqing, Heilongjiang 163319, China*
[2]*College of Engineering, Heilongjiang Bayi Agricultural University, Daqing, Heilongjiang 163319, China*

Correspondence should be addressed to Shujuan Yi; yishujuan@byau.edu.cn

Received 17 December 2021; Revised 12 January 2022; Accepted 18 January 2022; Published 10 February 2022

Academic Editor: Xin Ning

To address the problems of high energy consumption and time delay of the offloading strategies in traditional edge computing, a computation offloading strategy for the Internet of Things (IoT) using the improved Particle Swarm Optimization (PSO) in edge computing is proposed. First, a system model and an optimization objective function are constructed based on the communication model for the uplink transmission and the multiuser personalized computation task load model while considering constraints from multiple aspects. Then, the PSO is used to update the position of particles by encoding them and calculating the fitness values to find the optimal solution of the task offloading strategies, which greatly reduces the energy consumption during the task allocation process in the system. Finally, the simulations are conducted to compare the proposed method with two other algorithms in terms of the average time delay and energy consumption under different numbers of user mobile devices and data transmission rates. The simulation results showed that the average time delay and energy consumption of the proposed method are the smallest in different cases. And, the average delay and energy consumption are 0.205 s and 0.2 J, respectively, when the number of users' mobile devices is 80, which are better than the other two comparison algorithms. Therefore, the proposed method can reduce the task execution delay with less energy consumption.

1. Introduction

In recent years, with the rapid development of mobile Internet and wireless communication technology, the human society has entered the 5G era [1]. Mobile smart devices are widely used in many fields, which provide great convenience to all walks of life [2, 3]. However, it is difficult for existing technologies or individual hardware devices to provide shorter-delay and larger-bandwidth network communication to these devices [5, 6]. In addition, the characteristics of mobile smart terminal devices that consume large amounts of computing energy put forward higher demands on the performance of batteries [7, 8]. Therefore, the problems associated with the massive computation and energy consumption of smart devices have been the focus in recent researches [9, 10].

The mobile edge computing provides new approach to solve the aforementioned problems, which is also the most forward-looking solution [11, 12]. It allows terminal users to offload missions to servers for implementation, which can reduce the energy consumption and time delay [13]. In [14], a constrained multiobjective optimization task offloading model with corresponding algorithms was proposed aiming at the problem of allocating tasks from edge clients to edge servers as well as other edge clients in mobile edge computing, which takes the minimum energy consumption and processing delay as the objective function. However, this approach does not consider the balance between task delay and the network energy consumption. Ren and Wang proposed a method for quality-of-service (QoS) prediction in mobile edge computing environment [15]. It identifies the similar users for different users through clustering analysis

and applies the data provided by similar users to predict the QoS when the edge server accessed by the user is switched. But the energy minimization problem is not taken into consideration in this method. An ARIMA-BP-based selective offloading (ABSO) strategy for mobile edge computing was given by Zhao and Zhou, which can minimize the energy consumption of mobile devices while satisfying the latency requirements [16]. However, this method does not take into account the user experience and QoS. In [17], a novel mobility and dependency-aware QoS monitoring method for mobile edge environments was proposed to address the problem of bias between monitoring results and real results due to the dependency between user mobility and the QoS value. But this approach does not yield significant improvements in terms of the communication time delay. As the sudden task offloading of mobile users when they are located in hotspots may lead to overloading of several edge servers, a load balancing algorithm for mobile devices in edge cloud computing environments based on genetic algorithms was given by Lim and Lee [18]. Nevertheless, this algorithm does not consider the minimization of the task latency and the network energy consumption. In [19], the reliability-aware offloading and the minimization of energy consumption and delay were studied. A task-merging strategy based on mobile program component graph with the fundamental constraint of valuable offloading was proposed. Meanwhile, a fast algorithm was developed for the hybrid problem to minimize the computation complexity in program partitioning. However, the safety interruption problem is not taken into consideration when the user devices are energy-constrained. To solve the problem of limited computational resources in edge computing architectures, Shi et al. studied the problem of cross-server computation offloading and the collaboration between multiple edge servers for multitask mobile edge computing and proposed a greedy approximation algorithm, which can greatly reduce the overall energy consumption [20]. However, this method does not give a specific approach to improve the QoS of users.

Based on the above analysis, to address the problems of high energy consumption and time delay of the offloading strategies in traditional edge computing, a computation offloading strategy for IoT using improved PSO in edge computing is proposed. There are two main general ideas in this proposed method. First, the overall model of mobile edge computing system and the corresponding objective function are constructed by establishing the communication model and the computation task load model. In addition, the individual particles are encoded by introducing the improved particle swarm algorithm to improve the algorithm's performance in finding the optimal solution, compared with the traditional computation offloading strategies for IoT.

2. Model of the System and Optimization Object

2.1. Model of the System. The analysis of the task offloading system model of mobile edge computing in multiuser scenario is performed in this section.

Assume that the total amount of user mobile devices in a defined region is U; the user mobile device is set to u, where $u \in U\{1, 2, 3, \ldots, U\}$. Every mobile device has its corresponding mission that needs to be performed and the focus of all mobile devices is different (some user mobile devices focus on the energy consumption while some focus on the time delay, etc.). A base station is located in the defined area and there are server devices of mobile edge computing placed. The individual user mobile devices in U within the area can communicate with the server device of mobile edge computing through a radio access network and offload their computation missions to a server device of mobile edge computing for execution. The total number of channels in the base station is set to be V, and each user mobile device can connect to the server devices of mobile edge computing through only one channel.

In this paper, it is assumed that user mobile devices will not change the channel again when it is selected, and the computational resources of server devices of mobile edge computing are capped but central server has unlimited available computational resources. Moreover, one server device of mobile edge computing can only meet A user mobile devices for computing during the same time period. When the number of computation tasks goes beyond the limit of server device, the excess computation tasks will be offloaded to the remote central cloud server for execution through the backbone network. The basic architecture of mobile edge computing is shown in Figure 1.

As Figure 1 shows, the computation mission of every mobile device requires three steps for decision making in the process of offloading:

Decision 1. For the user mobile device side, clarify that the computation task is performed locally or offloaded to the server devices of mobile edge computing. It will get into Decision 2 if the computation mission is selected to be offloaded to the server devices of mobile edge computing.

Decision 2. On the mobile edge computing server side, it needs to ensure whether the computational resources are sufficient or not. If the computational resources are not sufficient, it gets into Decision 3.

Decision 3. When computational resources in mobile edge computing server are not sufficient, a decision needs to be made whether it should continue to wait in queue of the current server or to offload the mission to remote central server for execution.

2.2. Communication Model. In the system of mobile edge computing, it is assumed that each user mobile device in the specific area communicates through the uplink transmission channels that are orthogonal to each other. Namely, all the user mobile devices do not interfere with each other on the same frequency during the task offloading. Denoting the transmission power of mobile device u as P_u, then the uplink rate R_u of this mobile device is shown in the following equation:

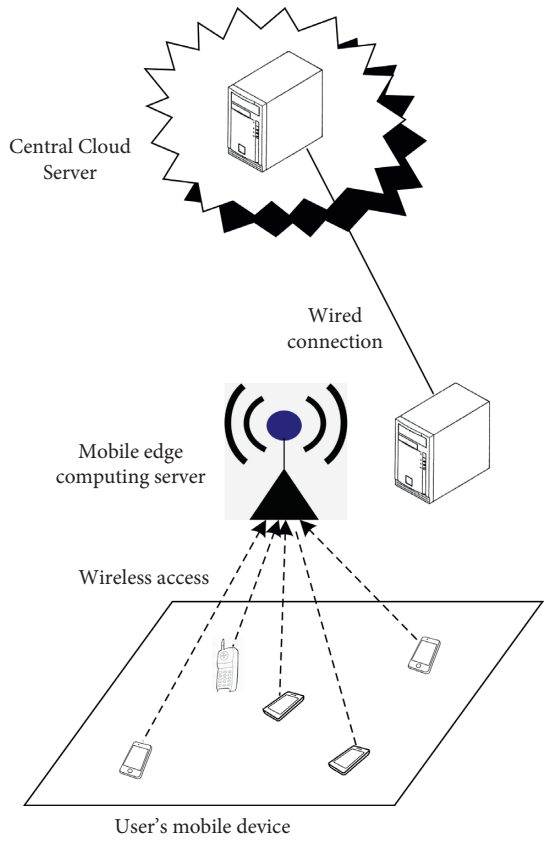

Figure 1: Basic architecture of mobile edge computing.

$$R_u = S_u \log_2\left(1 + \frac{P_u z_u}{\delta_u^2}\right), \quad \forall u \in U, \tag{1}$$

where S_u represents the uplink bandwidth of the user mobile device u and $B = \sum_{u=1}^{U} S_u$ represents the total uplink bandwidth of the system. z_u denotes the channel gain coefficient of the uplink between the user mobile device u and the base station. δ_u^2 is the noise power between the uplink of the user mobile device u and the base station.

2.3. Computation Task Model. This section demonstrates the process of simplifying the tasks by constructing a computation task model. In general, a computation task includes three main elements:

(1) The volume of data required for the computation task: the data mentioned here mainly consist of the program code, input parameters, and so on. If the computation task should be offloaded to the server device of mobile edge computing to carry out, these data need to be uploaded to the server via the mobile device's transmitter unit module.

(2) Computing capacity required for this task: it is usually denoted by the number of CPU cycles.

(3) The result data of this computation task: the result data need to be downloaded from the server device of mobile edge computing to the mobile device if the task is offloaded.

Thus, a computation task can be represented by a ternary equation containing these three elements, which can be written as

$$Q_v \triangleq (D_v, C_v, R_v), \tag{2}$$

where Q_v denotes the computation task. D_v denotes the volume of data required for the computation task. C_v indicates the computing capacity for the mission. R_v denotes the result data of this computation task.

The user mobile device can obtain these three elements (D_v, C_v, and R_v) of the computation task by the program call graph technology. Meanwhile, in order to fill the personalized needs of each user, various application types and corresponding data are provided to analyze and calculate the delay, and energy consumption in the situation of the computation task is completed locally and on cloud, respectively. Thus, the delay and energy consumption load model for multiusers can be constructed.

2.4. Personalized Computation Task Load Model. The personalized computation task load model for multiusers is performed in this section. From the above analysis, it is needed to clarify that the computation task of each user mobile device is selected to be executed locally or not at the beginning. The analysis is carried out for these two cases as follows.

The case where the computation task of user mobile device u is selected to be executed locally. We assume that the computing capacity of user u, which means the amount of CPU periods, is J_{u1} and the amount of CPU periods required to execute the computation task is j_{u1}. Thus, the time t_{u1} required for local calculation of the mission is

$$t_{u1} = \frac{j_{u1}}{J_{u1}}. \tag{3}$$

Hence, the energy consumption E_{u1} for the local execution of the computation task can be formulated as

$$E_{u1} = \lambda j_{u1} (J_{u1})^2, \tag{4}$$

where λ is the energy consumption coefficient and its value is related to the chip structure of the user mobile device. λ is a constant and is usually set as $\lambda = 10^{-26}$.

Therefore, the total overhead W_{u1} when the computation task of the user mobile device u is executed locally can be written as

$$W_{u1} = \alpha_{u1} E_{u1} + \alpha_{u2} t_{u1}, \tag{5}$$

where α_{u1} andα_{u2} represent the trade-off coefficients of the energy consumption and the time delay for task execution, respectively, when the user is making the offloading decision. And they need to satisfy the constraints shown in the following equation.

$$\begin{cases} \alpha_{u1}, \alpha_{u2} \in [0, 1], \\ \alpha_{u1} + \alpha_{u2} = 1. \end{cases} \tag{6}$$

It can be seen in equation (5) that the total overhead consists of two components, which are the energy consumption and the time delay incurred during local task execution. When α_{u1} is larger, it indicates that the residual energy of user mobile device is less and more attention will be paid to the energy consumption of the user mobile device when making offloading decisions. While α_{u2} is larger, it indicates that the computation task of the user mobile device is delay sensitive and more attention should be paid to the task execution delay when making offloading decisions. In addition, users can also dynamically adjust and balance the different concerns according to their specific situations.

The computing task of the u-th user is unloaded to the edge server for execution Set the processing delay of the mission at the server device of mobile edge computing as $t_{u2}(P_u, J_{u2})$, which can be formulated as

$$t_{u2}(P_u, J_{u2}) = t_{u2(1)}(P_u) + t_{u2(2)}(J_{u2}), \tag{7}$$

where $t_{u2(1)}(P_u)$ denotes the time delay caused by uploading the input data of computation tasks to the mobile edge computing server via uplink. $t_{u2(2)}(J_{u2})$ denotes the time delay caused by executing the computation task at the mobile edge computing server. $t_{u2(1)}(P_u)$ can be calculated as

$$t_{u2(1)}(P_u) = \frac{s_u}{W_u \log_2(1 + \sigma_u P_u)}, \tag{8}$$

where $\sigma_u = z_u/\delta_u^2$.

When a mission is uploaded to a server device of mobile edge computing, computational resources J_{u2} will be allocated. And the time delay $t_{u2(2)}(J_{u2})$ of the task execution can be calculated as

$$t_{u2(2)}(J_{u2}) = \frac{j_u}{J_{u2}}. \tag{9}$$

The energy consumption when transferring the computation task of user mobile device u to the mobile edge computing server can be written as

$$E_{u2} = \frac{P_u}{\beta} t_{u2(1)}(P_u) = \frac{P_u}{\beta} \frac{s_u}{W_u \log_2(1 + \sigma_u P_u)}, \tag{10}$$

where β is the efficiency of the device transmission power amplifier.

The total overhead W_{u2} when the computation task of user mobile device u should be offloaded to a server device of mobile edge computing to carry out can be formulated as

$$W_{u2} = \alpha_{u1} E_{u2} + \alpha_{u2} t_{u2}(P_u, J_{u2}). \tag{11}$$

As depicted in equation (11), the total overhead consists of 2 contents. They are the energy consumption incurred by task execution and the time delay of the remote server device.

Through the analysis of the abovementioned task offloading process, it can be seen that the impact of task offloading on the energy consumption and delay of the user's mobile device is the main consideration. As the computing capacity of the mobile edge computing server is much larger than that of the user mobile devices, the energy consumption incurred by executing the computation tasks on the edge server can be neglected.

In addition, considering that the volume of the result data is generally small when the computation task is executed on the mobile edge computing server, the energy consumption and time delay required when returning execution results to the user mobile device can be ignored.

2.5. Optimization Objectives. Based on the aforementioned analysis, the total overhead W_u of the user mobile device u during task offloading can be written as

$$W_u = (1 - x_u) W_{u1} + x_u W_{u2}. \tag{12}$$

Here, the objective function is formulated as the minimization of the total overhead of the user mobile devices during the task offloading process. The optimal offloading strategy $X_O = \{x_1, x_2, x_3, \ldots, x_u\}$, the optimal uplink allocation power $P_O = \{P_1, P_2, P_3, \ldots, P_u\}$, and the optimal mobile edge computing resource allocation strategy $M_O = \{m_1, m_2, m_3, \ldots, m_u\}$ are obtained with the minimum W_u. Hence, the optimization objective function for the computation task offloading of user mobile devices in a specific area can be formulated as

$$\min_{X,P,M} W = \sum_{u=1}^{U} (1 - x_u) W_{u1} + x_u W_{u2}. \tag{13}$$

The constraints are shown in the following equation.

$$\begin{cases} 1: x_u \in \{0, 1\}, \forall u \in U, \\ 2: 0 < P_u \le P_{\max}, \forall u \in U_2, \\ 3: \sum_{u \in U_2} J_2 \le J_{\max}, \\ 4: J_{u2} > 0, \forall u \in U_2, \\ 5: \sum_{u \in U} x_u S_u \le B, \end{cases} \tag{14}$$

where x_u is the offloading decision of users. When $x_u = 1$, user mobile device u will offload the mission to the server device of mobile edge computing for execution. When $x_u = 0$, user mobile device u chooses to execute the task locally. $P_{\max}$ represents the maximum transmission power, $J_{\max}$ represents the maximum computation resource, and U_2 represents the set of user mobile devices that offload the mission to the server device of mobile edge computing to carry out.

Constraint 1 shows the constraints about user's offload decisions. Constraint 2 requires that the uplink power of the user mobile device in offloading process cannot exceed its maximum transmission power. Constraint 3 shows that computational resources allocated to the user mobile device in offloading process must not exceed the maximum computational resources available in the mobile edge computing server. Constraint 4 indicates that computational resources allocated to the user mobile device in offloading process cannot be less than zero. Constraint 5 represents the bandwidth limit of the system. The number of user mobile devices that are allowed to upload data at the same time in the specific area cannot be more than U.

3. Computational Resource Allocation Strategy Using Improved PSO

It is clear that the numerous objective function constraints make it much more difficult to obtain the optimal computation offloading strategy. Therefore, an improved PSO is introduced to find an optimal solution of task offloading strategy.

3.1. Particle Coding. The particles in the particle swarm denote some mobile edge computing servers where the current computation task is to be offloaded. And the offloading decision vector $X = \{x_1, x_2, x_3, \ldots, x_u\}$ corresponding to each particle represents the optimal execution point for this computation task.

The size of the particle swarm is G, and total amount of computation tasks currently selected for local execution is L, and the total number of mobile edge computing servers in a given area is Y. The set of all tasks is denoted as $R = \{r_1, r_2, r_3, \ldots, r_v\}$. In order to formulate different tasks mathematically, we assume that the mobile device of user u has r_i number of computation tasks that need to be executed. r_i is related to three variables and can be denoted as $r_i = r_i(I_i, \rho_i, E_c)$, where I_i indicates input data volume, ρ_i is the computational density, E_c indicates energy consumption for the computation task execution. The cycle frequency of all servers can be represented as set $T = \{t_1, t_2, t_3, \ldots, t_n\}$, which contains the cycle frequency of each server in the system. The maximum transmission rate can be calculated by channel gain matrix Z, which is shown as

$$Z = \begin{bmatrix} Z_{1,1} & Z_{1,2} & \cdots & Z_{1,n} \\ Z_{2,1} & Z_{2,2} & \cdots & Z_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ Z_{v,1} & Z_{v,2} & \cdots & Z_{v,n} \end{bmatrix}, \tag{15}$$

where the diagonal element $Z_{i,i}$ in the channel gain matrix Z represents a channel gain of 0, and $Z_{i,i}\,(1 \le i \le v, 1 \le i \le n, i \ne j)$ represents the channel gain required for the transmitting the computation task of the user mobile device i to server j.

All individual particles in the particle population are integer coded. Each individual particle in the encoding process can take any integer from $[1, n]$. For example, a task set contains 5 computation tasks: $R = \{r_1, r_2, r_3, r_4, r_5\}$, and the particles are encoded into $[1, 2, 0, 1, 2]$, which characterizes the position where different computation tasks are executed. The element value of tasks r_1 and r_4 is 1, which indicates that these missions will be offloaded to a server whose number is 1 for execution. The value of tasks r_2 andr_5 is 2, which indicates that they will be offloaded to a server whose number is 2 for execution. The value of task r_3 is 0, denoting that this task will be executed locally. The elements x_i in the offloading decision vector take values from $[0, n]$. $x_i = k\,(1 \le k \le n)$ means that the computation task will be offloaded to the kth server to carry out.

The moving velocity of the particles in the particle swarm is denoted as $A = \{a_1, a_2, a_3, \ldots, a_u\}$, indicating the velocity of current computation missions sent to server. The moving velocity of the particles is encoded in the same dimension as the size of task sets and initialized as an integer. Rounding operations are also adopted during the update process. For example, there is a task set containing 5 computation tasks: $R = \{r_1, r_2, r_3, r_4, r_5\}$ and the particles are encoded into $[4, 1, 2, 0, 3]$, which characterizes the execution servers for different computation tasks. The element value of task r_2 is 1, which means that the mission is carried out by a server where original server is shifted down by 1 position. The element value of task r_5 is 3, which indicates that this mission is carried out by a server where original server is shifted down by 3 positions. And the element value of task r_4 is 0, denoting that this task is executed by the original server.

In the iterative computation of the PSO, the optimal position of every individual particle is denoted as $D_O = \{d_{o1}, d_{o2}, d_{o3}, \ldots, d_{on}\}$, and the optimal position that occurs among all particles is denoted as $F_O = \{f_{o1}, f_{o2}, f_{o3}, \ldots, f_{on}\}$. The set D indicates that the total overhead of the system is minimized when each particle is allocated to the task in its corresponding way. The set F represents that the total overhead of the system is minimized when all the particles are allocated to the tasks in such way.

3.2. The Fitness Function. The fitness value of the particles represents the total overhead of the system when missions are sent to different servers, which can be calculated as

$$h(X) = \sum_{v=1}^{n} \sum_{u=1}^{U} C_u + \zeta \cdot \sum_{v=1}^{n} \sum_{u=1}^{U} (E_u - E_{\max}), \tag{16}$$

where C_u denotes the total time delay for executing an offloaded task. ζ is the penalty coefficient. E_u is the transmission energy consumption of the uth computation task. $E_{\max}$ is the maximum energy consumption.

3.3. Computation Offloading Process. In u-dimensional discrete binary particle swarm problem, the velocity and position of the ith particle are generally denoted by n-dimensional vector A and D, which are $X_i = \{x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iu}\}$ and $Y_i = \{y_{i1}, y_{i2}, y_{i3}, \ldots, y_{iu}\}$, respectively. Therefore, in the task offloading optimization problem, the positions of particles represent the decision variable for the execution position of each computation task, which can be written as

$$\{x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iu}\} = \{A_1, A_2, \ldots, A_j, \ldots, A_u\}. \tag{17}$$

The overall flow of mission offloading strategy for IoT on edge computing using improved PSO is shown in Figure 2.

The detailed process of the computation task offloading based on the discrete binary PSO is shown as follows:

(1) Initialization for the particle swarm: all particles in the swarm are divided into decision variables in a random way. The values of the objective function of different particles and the optimal positions of the individual particle D_O and the particle swarm F_O are calculated according to equation (13).

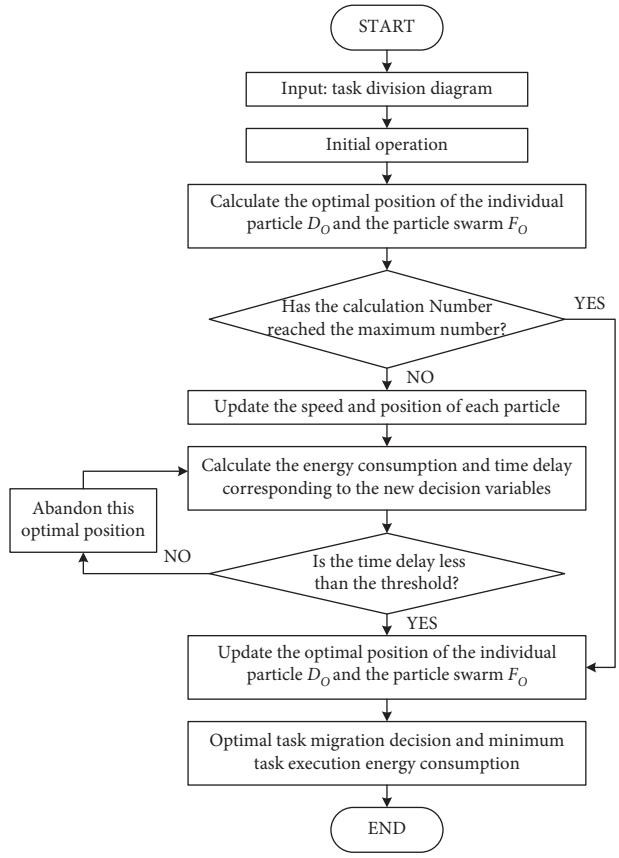

Figure 2: The overall flow of computation task offloading strategy algorithm.

(2) Set the maximum number of iterations; update the velocity and positions of all the particles and calculate the corresponding energy consumption E_u and the time delay C_u related to the new decision variables. If the time consumed to execute the task is larger than the time delay threshold, the decision variable cannot meet the requirement, and the particle position should be discarded. Otherwise, the time consumed to execute the task is smaller than the threshold and the requirement is satisfied. Then the optimal positions of the individual particle D_O and the particle swarm F_O should be updated according to the calculation results of the objective function.

(3) The relatively optimal decision variables and energy consumption are obtained when the iteration is complete.

4. Experiments and Analysis

4.1. Simulation Environment and Parameter Settings. When there are multiple user mobile devices in a specific area, the server should be located near the base station and all user mobile devices should be within the coverage area. As shown in Table 1, basic simulation parameters of channel are set according to the 3GPP standard.

4.2. Analysis of Simulation Results. First, the convergence performance of the proposed computation offloading algorithm for IoT using improved particle swarm algorithm in edge computing is analyzed. The number of users in the defined region is set to 80, the number of servers is 15, and the size of each computation mission is 5 Mb. The simulation results of the time delay for whole missions under different numbers of iterations are shown in Figure 3.

In Figure 3, the proposed method converges quickly after 20 iterations, and the total time delay of the system remains almost the same when the number of iterations is more than 20, indicating that the global optimal solution is obtained. Hence, the proposed method has strong global optimization capability and searching performance in the early stage of computation and can also perform fast global search in the later stage. The proposed method reduces the total system time delay from about 0.275 s to about 0.205 s, which is a 25.45% decrease, and greatly improves the overall performance of the computation offloading strategy.

Then, a detailed analysis for the effect of different crossover rate and mutation rate on convergence is carried out. We set the total number of users in this region to be 80. Figure 4 depicts the relationship between the total overhead of user mobile devices and the number of iterations for three different fixed crossover and mutation probabilities and the adaptive crossover and mutation probabilities.

Table 1: Basic simulation parameters.

Parameter	value
Noise power in uplink bandwidth	−108 dBm
Radius of the area	1 km
User maximum transmit power	25 dBm
System bandwidth	15 MHz
User bandwidth	0.5 MHz
Task input data volume	300–1500 KB
CPU cycles required for task	0.1–0.8 GHz
Maximum time delay	3–5 s
User's computing power	0.2–1.2 GHz
MEC server's computing power	3 GHz
Population size	50
Maximum number of evolutions	100

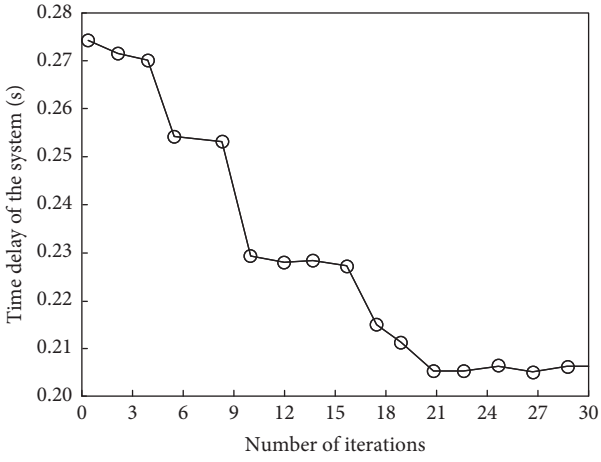

Figure 3: The time delay of the system under different numbers of iterations.

As shown in Figure 4, in the case of fixed crossover and mutation probabilities, whatever their values are will make the algorithm fall into the local optimal solution trap. In addition, fixed crossover and mutation probabilities will also prolong the searching process and consume more time. In contrast, the adaptive crossover and mutation probabilities can adjust dynamically as the fitness value changes in searching and not sink into the local optimal solution trap, which greatly reduce the time to targeting global optimal solution and improve calculation speed.

4.3. Comparative Analysis. A comparative analysis of the performance in delay and energy consumption of the proposed method with the algorithms proposed in [16] and [20] under the same conditions is provided. Simulations are conducted for the average time delay of different algorithms with different numbers of user mobile devices. The simulations results are illustrated in Figure 5.

In Figure 5, the average time delay of all three algorithms is proportional to the number of user's mobile devices. With the rise of the number of user's mobile devices, average delay and overall growth rate for the proposed method are the smallest. And the average time delay is only 0.205 s in the case of the total user's mobile devices number being 80. The

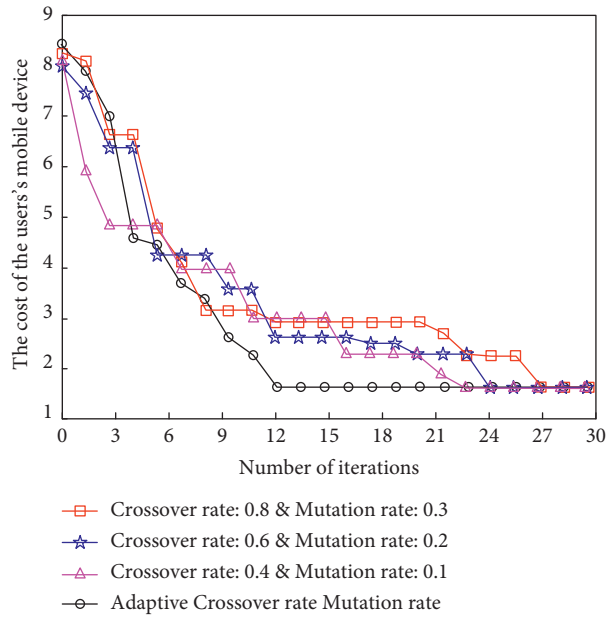

Figure 4: The effect of different crossover and mutation probabilities on algorithm convergence.

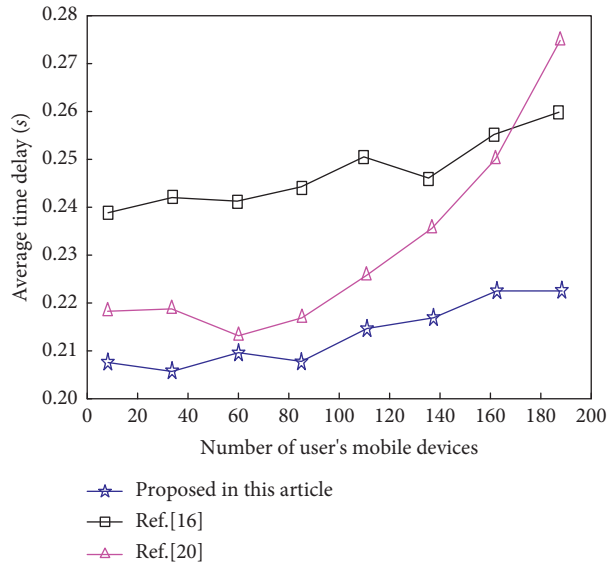

Figure 5: The average time delay of different algorithms with different numbers of user's mobile devices.

proposed computational resource allocation strategy is studied for the communication model of uplink transmission and the multiuser personalized computation task load model, on which the system model and objective function are constructed. And the objective function is constrained in multiple aspects, which improves the computation efficiency of the system model to a certain extent. The algorithm proposed in [20] has a larger average time delay whose variation rate is low as the user amount grows. Because of that, the performance of time delay is sacrificed to save more energy. In addition, the balance between delay and network energy consumption is not considered. The delay of the

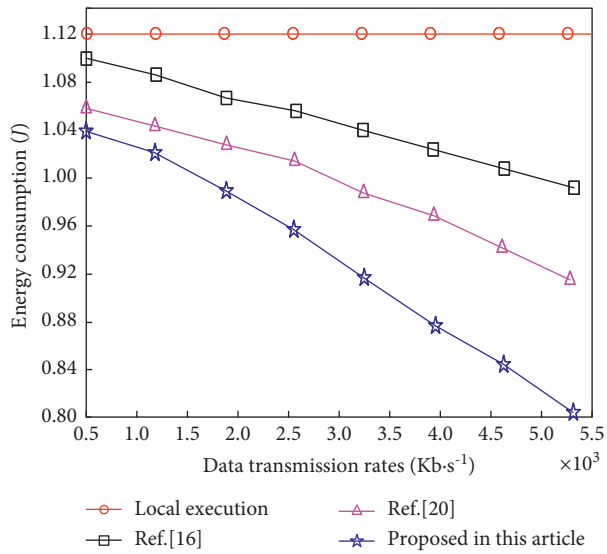

Figure 6: Comparison of energy consumption of different algorithms at different data transmission rates.

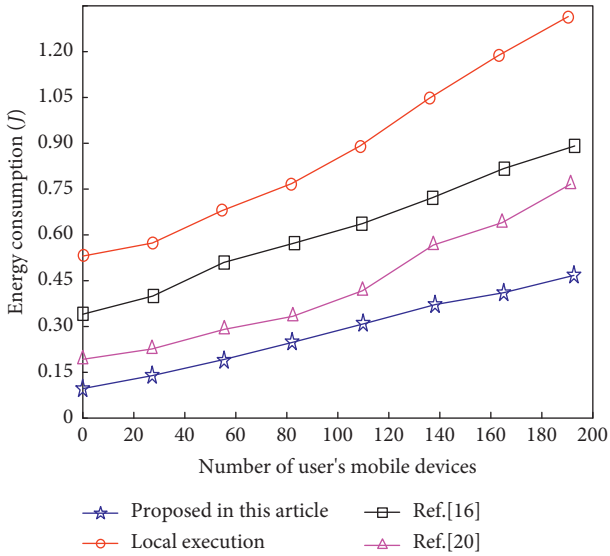

Figure 7: Comparison of energy consumption of different algorithms with different numbers of users' mobile devices.

algorithm proposed in [16] is smaller with fewer user mobile devices, but the growth rate of that is large with the rise of users because the tasks selected for local execution increase significantly when there are many users' mobile devices.

Simulation for the energy consumption of different algorithms at various transmission rates is conducted; the simulation curve is demonstrated in Figure 6.

In Figure 6, energy consumption of computation task executed locally is independent with data transmission rates, which is due to the fact that data do not need to be transmitted when the computation task is selected to execute locally. The energy consumption of all three algorithms descends as the data transmission rate increases, which indicates that the consumption of task offloading drops with the rise of transmission rate. In addition, the energy consumption in the proposed method decreases at the fastest rate and the energy consumption is the smallest for the same transmission rate. This is because the consumption and delay decrease as transmission rate increases when data are transmitted between users and servers. The proposed task offloading strategy can perform more fine-grained and frequent offloading operations due to the introduction of PSO.

Figure 7 shows the comparison for energy consumption in each algorithm with different numbers of users' mobile devices.

As depicted in Figure 7, the energy consumption in different algorithms grows with the rise of users, while energy consumption in the proposed method is the smallest with the lowest growth rate. The energy consumption is 0.2 J in the case of the number of users being 80. Moreover, the energy consumption of local execution is the largest. The better capability in proposed strategy is due to the introduction of PSO, which greatly improves searching efficiency and results in a lower proportion of locally executed tasks while considering the time delay, thus reducing the system energy consumption.

5. Conclusion

To address the problems in traditional computation offloading strategies for IoT environment, a computation offloading strategy for IoT using improved particle swarm algorithm in edge computing is proposed. The overall system model is constructed, and the searching efficiency of optimal solution in algorithm is improved by introducing the improved PSO. Based on the particle swarm algorithm, the optimal solution of task offloading strategy is obtained by encoding the particles and calculating the fitness value so as to update the positions of different particles, which minimizes the energy consumption produced by the system task allocation. The proposed strategy is analyzed by simulation experiments. It can be known from simulation that compared with the other two algorithms, the method shows the

least average time delay and energy consumption for task execution under different situations. Meanwhile, the growth rate of the average delay and consumption are smallest accompanied by the user's rise, and the energy consumption drops most rapidly accompanied by the rise of the transmission rate. Hence, the proposed method can achieve the best overall system performance.

The future work will focus on the overall capability of the proposed strategy and improved way considering the situations that users' mobile devices can move in a specific area or within different areas.

Data Availability

The data included in this paper are available upon request to the corresponding author without any restriction.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Natural Science Foundation Project of Heilongjiang Province (no. C2018050), Heilongjiang Agricultural Reclamation Administration's Projects (no. HKKY190201-02), and Heilongjiang Innovative Talent Project (no. CXRC2017014).

References

[1] M. McClellan, C. Cervello-Pastor, and S. Sallent, "Deep learning at the mobile edge: opportunities for 5G networks," *Applied Sciences-Basel*, vol. 10, no. 14, pp. 24–33, 2020.

[2] M. Habib ur Rehman, A. Batool, and K. Salah, "The rise of proximal mobile edge servers," *IT Professional*, vol. 21, no. 3, pp. 26–32, 2019.

[3] J. Lee, D. Kim, and J. Lee, "Mobile edge computing based immersive virtual reality streaming scheme," *Computing and Informatics*, vol. 38, no. 5, pp. 1131–1148, 2019.

[4] G. Liang, Q. Wang, and J. Xin, "Survey of mobile edge computing resource allocation," *Journal of Cyber Security*, vol. 6, no. 3, pp. 227–256, 2021.

[5] W. X. Shi, J. D. Zhang, and R. D. Zhang, "Share-based edge computing paradigm with mobile-to-wired offloading computing," *IEEE Communications Letters*, vol. 23, no. 11, pp. 1953–1957, 2019.

[6] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C. H. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," *Software: Practice and Experience*, vol. 50, no. 5, pp. 489–502, 2020.

[7] K. Zhang, X. Gui, and D. Ren, "Survey on computation offloading and content caching in mobile edge networks," *Journal of Software*, vol. 30, no. 8, pp. 2491–2516, 2019.

[8] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, no. 5, pp. 160–168, 2019.

[9] T. Wang, Y. C. Lu, and Z. H. Cao, "When sensor-cloud meets mobile edge computing," *Sensors*, vol. 19, no. 23, pp. 27–38, 2019.

[10] J. Lee, J. W. Kim, and J. Lee, "Mobile personal multi-access edge computing architecture composed of individual user devices," *Applied Sciences-Basel*, vol. 10, no. 13, pp. 57–66, 2020.

[11] G. Jia, G. Han, J. Du, and S. Chan, "A maximum cache value policy in hybrid memory-based edge computing for mobile devices," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4401–4410, 2019.

[12] D. Belli, S. Chessa, L. Foschini, and M. Girolami, "A probabilistic model for the deployment of human-enabled edge computing in massive sensing scenarios," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2421–2431, 2020.

[13] J. Ni, X. Lin, and X. S. Shen, "Toward edge-assisted Internet of Things: from security and efficiency perspectives," *IEEE NETWORK*, vol. 33, no. 2, pp. 50–57, 2019.

[14] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. Salinas Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Transactions on Mobile Computing*, vol. 20, no. 10, pp. 2992–3005, 2021.

[15] L. Ren and W. Wang, "Method for QoS prediction in mobile edge computing environment," *Journal of Chinese Computer Systems*, vol. 41, no. 6, pp. 1176–1181, 2020.

[16] M. Zhao and K. Zhou, "Selective offloading by exploiting ARIMA-BP for energy optimization in mobile edge computing networks," *Algorithms*, vol. 12, no. 2, pp. 23–31, 2019.

[17] P. Zhang, Y. Zhang, H. Dong, and H. Jin, "Mobility and dependence-aware QoS monitoring in mobile edge computing," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 1143–1157, 2021.

[18] J. Lim and D. Lee, "A load balancing algorithm for mobile devices in edge cloud computing environments," *Electronics*, vol. 9, no. 4, pp. 34–42, 2020.

[19] L. Dong, W. Wu, Q. Guo, M. N. Satpute, T. Znati, and D. Z. Du, "Reliability-aware offloading and allocation in multilevel edge computing system," *IEEE Transactions on Reliability*, vol. 70, no. 1, pp. 200–211, 2021.

[20] Y. P. Shi, Y. J. Xia, and Y. Gao, "Cross-server computation offloading for multi-task mobile edge computing," *Information*, vol. 11, no. 2, pp. 88–97, 2020.

[21] W. M. Zhang, Y. W. Zhang, and Q. L. Wu, "Mobility-enabled edge server selection for multi-user composite services," *Future Internet*, vol. 11, no. 9, pp. 87–98, 2019.

[22] J. N. Sun, Q. Gu, and T. Zheng, "Joint communication and computing resource allocation in vehicular edge computing," *International Journal of Distributed Sensor Networks*, vol. 15, no. 3, pp. 125–136, 2019.

[23] X. Kong, S. Tong, H. Gao et al., "Mobile edge cooperation optimization for wearable internet of things: a network representation-based framework," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5050–5058, 2021.

[24] S. Joo, H. Kang, and J. Kang, "CoSMoS: cooperative sky-ground mobile edge computing system," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 8373–8377, 2021.

[25] A. J. Ferrer, J. M. Marques, and J. Jorba, "Towards the decentralised cloud: survey on approaches and challenges for mobile, ad hoc, and edge computing," *ACM Computing Surveys*, vol. 51, no. 6, pp. 685–693, 2019.

[26] S. Y. Guo, X. Hu, and G. S. Dong, "Mobile edge computing resource allocation: a joint Stackelberg game and matching strategy," *International Journal of Distributed Sensor Networks*, vol. 15, no. 7, pp. 354–365, 2019.