International Journal of Innovative
Computing, Information and Control
Volume **20**, Number **3**, June **2024**

ICIC International ©2024 ISSN 1349-4198
pp. **693–707**

# TASK MIGRATION FOR CLOUDLET FEDERATION BASED ON IMPROVED PARTICLE SWARM OPTIMIZATION ALGORITHM

Hengzhou Ye, Junhao Guo and Xinxiao Li*

Guangxi Key Laboratory of Embedded Technology and Intelligent System
Guilin University of Technology
No. 319, Yanshan Street, Yanshan District, Guilin 541006, P. R. China
2002018@glut.edu.cn; 15236163819@163.com; *Corresponding author: renpet@glut.edu.cn

Received July 2023; revised November 2023

Abstract. *The CloudLet Federation (CLF) collaboratively provides edge computing services to mobile devices by sharing cloudlet resources from multiple CloudLet Providers (CLP). The task migration strategy in the scenario of CLF needs to pay attention to the new problems such as resource pricing and resource utilization of CLF when resources are shared between CLPs. By exploring the delay model of task offloading, and the pricing models of tasks and cloudlet resources leasing in CLF, we construct an optimization model of task migration based on the total profit of CLF. An improved particle swarm optimization algorithm, marked as GRPSO, is proposed to solve the model. GRPSO uses chaotic strategy to initialize the population, dynamically adjusts the inertia weight and learning factor according to the number of iterations and particle diversity, and promotes the population transition by combining the opposite-based learning strategy and the cross-mutation idea of genetic algorithm. The effectiveness of the proposed scheme is verified by simulation experiments.*

**Keywords:** Edge computing, Cloudlet federation, Task migration, Particle swarm optimization, GRPSO

1. **Introduction.** The Mobile Edge Computing (MEC) paradigm manages and processes non-resource intensive applications by deploying cloudlet servers at the network edge, making computing resources closer to end users, to solve the shortage of mobile device resources and high latency caused by cloud data centers [1-3]. Although MEC makes resources closer to the target users, because of the heterogeneity of 5G network users' resource demands, the resources of a single CLP are difficult to meet users' diversified resource demands. Thus, researchers propose a horizontal edge federation [4,5], whose core idea is to meet the latency requirements of an increasing number of IoT devices by integrating the resources of edge nodes. When the cloudlet acts as the edge device, this horizontal edge federation forms the CLF architecture [6], which promotes better resource collaboration among CLPs, reduces the deployment cost of cloudlet and improves the service quality of cloudlet with a wide, fast and reliable geographical distribution.

Task migration is an important study content of MEC. The task migration for CLF requires more considerations. 1) The task request is usually associated with a CLP, which may be migrated to the associated CLP or the other CLP in the CLF. This means that there are more options for task migration. 2) Considering the heterogeneity of the cloudlet resources and task requests of CLPs, the price factor should be considered when renting

DOI: 10.24507/ijicic.20.03.693

the cloudlet resource between CLPs. 3) The resource utilization or profit of the federation is also a noteworthy optimization target.

This paper studies the task migration strategy in the CLF scenario, and the main contributions are as follows.

1) Aiming at task migration in the CLF scenario, the delay model, the task quotation model, and the pricing model of the cloudlet resource shared in the federation are constructed. Furthermore, the task migration decision model with the total profit of CLF as the optimization goal is constructed.
2) An improved particle swarm optimization algorithm (GRPSO) is designed to solve the model. Compared with the classical PSO, GRPSO uses chaotic strategy to initialize the population, adopts the inertia weight and learning factor adaptive adjustment strategy, and combines the opposite-based learning strategy and the cross-mutation idea of genetic algorithm.
3) The results of the simulation experiment show that the GRPSO algorithm is better than the existing methods in the total profit of the CLF and the average delay of tasks.

The rest of this paper is organized as follows. The relevant work about task migration for edge federation or CLP is discussed in Section 2. Section 3 explores the task delay model, the task quotation model, the resource pricing model in CLF and the profit model of CLF, and clarifies the optimization goal of the task migration for CLF. In Section 4, an improved particle swarm optimization algorithm is designed to solve the task migration model for CLF. In Section 5, the simulation experiments carried out are described and the experimental results are analyzed. Finally, the conclusion and future work are given.

2. **Related Work.** The high cost of deployment and management of edge resources is an important factor that restricts the market landing of edge computing. To reduce the cost of deployment and management of edge resources, Cao et al. [6] proposed the edge federation framework. When edge resources exist in the form of cloudlets, Huang et al. [7] and Ye et al. [8] studied the edge federation as a CLF. Some academic studies have been carried out against the edge federation scenario or the CLF scenario. Latif et al. [9] and Liao et al. [10] focused on the federated learning problem in CLF. Nayyer et al. [11] proposed a CLF caching model to alleviate the challenge of resource scarcity. Li et al. [12] are concerned with the distribution of cloudlet storage resources. Huang et al. [7] discussed the pricing problem of leasing resources within CLF, based on the economic model. Ye et al. [8] also focused on the pricing problem of leasing resources within CLF, which adopts a double auction strategy.

Resource location and task migration for CLF have also received attention in recent years. Baghban et al. [4] modeled the resource location and task migration problem of CLF as a mixed integer linear programming problem with the optimization goal of delay minimization, and proposed a multidimensional fractional knapsack algorithm based on federated learning to solve this problem. Ma et al. [13] proposed a cloudlet-assisted mobile edge computing framework based on vertical CLF, aiming to balance system delay and cost. Cao et al. [6] explicitly proposed the concept of CLF integrated service provisioning model, which considers vertical and horizontal two-dimensional integration and defines it as an integer linear programming problem with a varied-dimensional shrinkage method to solve the large-scale optimization problem. Awada and Zhang [14] proposed a collaborative architecture of vertical and horizontal CLF to provide the intelligent resource scheduling and optimization solution in the joint edge computing system. These studies do not consider the pricing of resource sharing within CLF, and also ignore the differences in task types.

Ye et al. [15] considered the pricing problem of internal resource sharing in CLF, which supports two types of tasks: delay sensitive and delay tolerant. However, it is based on distributed task scheduling strategy. Because the task scheduling of CLPs is not coordinated, there may be concurrency conflict. This paper is based on the idea of centralized task scheduling, different from scenarios such as cloud federation [16-18], multi-cloudlet [5], and fog federation [19,20]. In this paper, a CLF is considered to be an independent cooperative relationship, that is, each EIP has its own user group, EIPs share cloudlet resources with each other but have to pay resource leasing fees, and there is time expenditure in task migration between EIPs.

## 3. System Model.

In this section, we keep our focus on the cloudlet federation framework, communication and computation models.

### 3.1. CLF model.

As shown in Figure 1, a CLF constitutes the edge layer of edge computing, which is composed of cloudlet clusters provided by K CLPs, and the $i$th CLP is represented by $clp_i$. Each CLP deploys several cloudlet servers, and $C_{i,j}$ represents the $j$th cloudlet server deployed by $clp_i$. A cloudlet server provides services in the form of Virtual Machine (VM) instances, and each of VM performs only one task at a time. We consider two types of VMs that execute Delay Tolerant (DT) tasks and Delay Sensitive (DS) tasks, respectively. The VMs of the same type are isomorphic, and the distance between cloudlet servers of the same CLP is ignored. Therefore, when scheduling a task, we only need to decide which CLP to schedule the task to, not the specific cloudlet server.

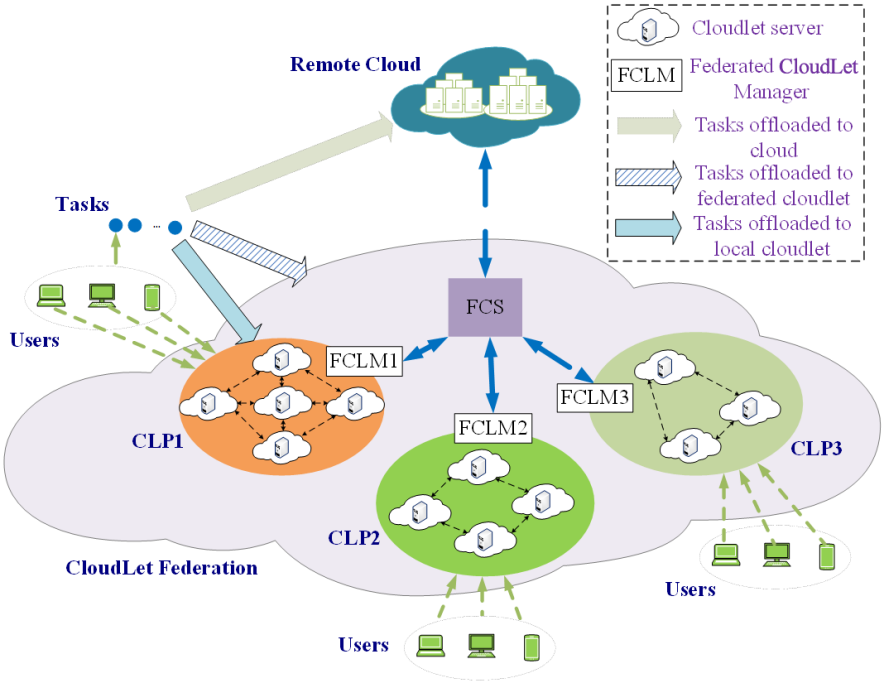



FIGURE 1. CLF architecture

The CLP interacts with the Federation Central Scheduler (FCS) through the Federated CloudLet Manager (FCLM), so that the FCS can get the resource information required for task scheduling decisions in real time. FCSs are responsible for centrally scheduling all tasks, with three possible outcomes: to the cloudlet server of the CLP associated with the task, to the cloudlet server of the federation, or to the remote cloud data center. The decision variable $x_{ijk} = 1$ means that the $j$th task associated with $clp_i$ is scheduled to the cloudlet server of $clp_k$ (for simplicity, $clp_0$ denotes the cloud data center).

3.2. **Delay model.** When the task is not executed locally, the delay usually includes the delay of the sending, the delay of the transmission, and the delay of execution. According to the cloudlet federation scenario, the schedule of the $j$th task associated with the $clp_i$, which is represented by $r_{ij}$, may have three scheduling results.

3.2.1. *It is executed by the cloudlet server of the $clp_i$.* The time delay of $r_{ij}$, which is represented by $t_1(r_{ij})$, can be determined by Equation (1), that is,

$$t_1(r_{ij}) = \frac{data_{ij}}{R_{ij}} + \frac{d1_{ij}}{V1_{ij}} + \frac{RI_{ij}}{CPU_i \cdot NCPU_i} \tag{1}$$

where $data_{ij}$ is the input data size of $r_{ij}$, $R_{ij}$ is the data transmission rate when the user uploads $r_{ij}$, and $d1_{ij}$ is the distance between the device corresponding to $r_{ij}$ and $clp_i$, $V1_{ij}$ is the channel propagation rate between the device corresponding to $r_{ij}$ and $clp_i$, $RI_{ij}$ is the number of Million Instructions (MI) that $r_{ij}$ needs to be executed. $CPU_i$ is the number of Million Instructions Per Second (MIPs) that can be executed by a single CPU of a VM in the cloudlet of $clp_i$, and $NCPU_i$ is the number of CPUs contained in a VM in the cloudlet of $clp_i$.

3.2.2. *It is executed by the cloudlet server of $clp_k$ ($k \neq i \&\& k \neq 0$).* That is, it is executed by the other CLP of the federation. The transmission delay consists of two stages: from the user to $clp_i$, and from $clp_i$ to $clp_k$. Considering the cost of identity authentication and information exchange between $clp_i$ and $clp_k$ (denoted as $\tau_{ik}$), the delay $t_2(r_{ij})$ of $r_{ij}$ can be determined by Equation (2), that is,

$$t_2(r_{ij}) = \frac{data_{ij}}{R_{ij}} + \frac{d1_{ij}}{V1_{ij}} + \frac{d2_{ik}}{V2_{ik}} + \tau_{ik} + \frac{RI_{ij}}{CPU_k \cdot NCPU_k} \tag{2}$$

where $d2_{ik}$ is the distance between $clp_i$ and $clp_k$, and $V2_{ik}$ is the channel propagation rate between $clp_i$ and $clp_k$.

3.2.3. *It is executed by the server of $clp_0$.* $r_{ij}$ is first transmitted to $clp_i$, and then transmitted to the remote cloud platform. The delay $t_3(r_{ij})$ of $r_{ij}$ can be determined by Equation (3), that is,

$$t_3(r_{ij}) = \frac{data_{ij}}{R_{ij}} + \frac{d1_{ij}}{V1_{ij}} + \frac{d3_i}{V3_i} + \frac{RI_{ij}}{CPU_0 \cdot NCPU_0} \tag{3}$$

where $d3_i$ is the distance between $clp_i$ and the cloud data center, and $V3_i$ is the channel propagation rate between $clp_i$ and the cloud data center.

To sum up, the delay $t(r_{ij})$ of $r_{ij}$ is obtained to satisfy the following equation.

$$t(r_{ij}) = \begin{cases} t_1(r_{ij}) \cdot x_{ijk}, & k = i \\ t_2(r_{ij}) \cdot x_{ijk}, & k \neq i \wedge k \neq 0 \\ t_3(r_{ij}) \cdot x_{ijk}, & k = 0 \end{cases} \tag{4}$$

3.3. **Pricing model.** Consider the two types of tasks, DT and DS. For any type of tasks, the response delay may affect the payment willingness of the user, that is, the quotation of the user for task A (denoted as $Q_A$ which is the function of the response delay in the execution of task A). This paper adopts the quotation strategy of [15]. For the DT task A, when its response delay is $t$, the quotation of the user for task A satisfies the following equations.

$$Q_A(t) = \begin{cases} P_A^{\max}, & t \leq t_A^{ept} \\ k_{dt} \cdot (t_A^{\max} - t) + P_A^{low}, & t_A^{ept} < t < t_A^{\max} \\ P_A^{low}, & t \geq t_A^{\max} \end{cases} \tag{5}$$

where $P_A^{\max}$ and $P_A^{low}$ represent the highest and lowest price that the user is willing to pay for task A, respectively. $t_A^{ept}$ and $t_A^{\max}$ are the user's expected delay and maximum tolerable delay for task A, respectively. $k_{dt}$ is the bid attenuation coefficient, satisfying the following equation.

$$k_{dt} = \frac{P_A^{\max} - P_A^{low}}{t_A^{\max} - t_A^{ept}} \tag{6}$$

For the DS task A, when its response delay is $t$, the quotation can be determined by Equation (7), that is,

$$Q_A(t) = \begin{cases} P_A^{\max}, & t \le t_A^{ept} \\ P_A^{\max} \cdot \left(\alpha \left(t - t_A^{ept}\right)\right)^{\beta}, & t_A^{ept} < t < t_A^{\max} \\ P_A^{low}, & t \ge t_A^{\max} \end{cases} \tag{7}$$

where $\alpha$ and $\beta$ are adjustable parameters whose values should ensure the continuity of the curve described in Equation (7).

When task A associated with $clp_i$ is scheduled to $clp_k$, it means that $clp_i$ needs to rent the resource of $clp_k$. Giving that user's quotation for task A is denoted as $Q_A$, the resource cost of $clp_k$ required to execute task A is $cost_A^k$, the total amount of cloudlet resources owned by $clp_k$ is $Total_k$, and the amount of available resources is $Res_k$, the quotation of $clp_k$ for executing task A (denoted as $Rent_A^k$), satisfies the following equation.

$$Rent_A^k = cost_A^k + \frac{Total_k - Res_k}{Total_k}\left(Q_A - cost_A^k\right) \tag{8}$$

3.4. **CLP profit model.** The total profit of a CLP is sum of the profit from executing the tasks locally, the profit from migrating the tasks to other CLPs, and the profit from renting out resources to other CLPs. These tasks that are scheduled to the cloud data center are not considered relevant to the cloudlet. The profit of $clp_i$ (denoted as $profit(clp_i)$) satisfies the following equation

$$\begin{aligned} profit(clp_i) = & \sum_{k=i} \left(Quote(r_{ij}) \cdot x_{ijk} - RI_{ij} \cdot cost_i\right) \\ & + \sum_{k \neq i \wedge k \neq 0} \left(Quote(r_{ij}) \cdot x_{ijk} - Rent_k(r_{ij}) \cdot x_{ijk}\right) \\ & + \sum_{k \neq i} \left(Rent_i(r_{kj}) \cdot x_{kij} - RI_{kj} \cdot cost_i\right) \end{aligned} \tag{9}$$

where the summation means traversing all possible $j$ and $k$, $Quote(r_{ij})$ is the user's quotation for $r_{ij}$, $Rent_k(r_{ij})$ is the rental price of the resource rented by $clp_k$ for executing $r_{ij}$, and $cost_i$ is the unit resource cost of $clp_i$.

3.5. **Optimization objective.** The optimization goal of task scheduling is to maximize the profit of cloudlet federation. Due to the differences in cloudlet resources and user task requests of different CLPs, the total profit of cloudlet federation is defined by Equation (10), that is,

$$Profit = \sum_{i=1}^{K} \left(\frac{res_i}{\sum_{j=1}^{K} res_j} + \gamma \frac{MI_i}{\sum_{j=1}^{K} MI_j}\right) profit(clp_i) \tag{10}$$

where $res_i$ is the total number of cloudlet resources owned by $clp_i$, $MI_i$ is the sum of millions of instructions needed to be executed for the tasks associated with $clp_i$, and $\gamma \in [0, 1]$ is the weight.

4. **Improved Particle Swarm Optimization Algorithm.** The task migration model established for CLF is suitable to be solved by some evolutionary algorithm. Particle Swarm Optimization (PSO) is a widely used evolutionary algorithm because of its fast search speed, few parameters and easy implementation. However, PSO also has problems such as lack of dynamic adjustment of speed, easy to fall into local optimum, and poor selection of parameters. An improved PSO algorithm is designed to solve the task migration model for CLP.

4.1. **Classical PSO algorithm.** The PSO algorithm uses massless and volumeless particles to represent individuals. It provides them with behavior rules to search for optimal or approximately optimal solutions in the search space. Each particle of PSO has its velocity and position, and their updating rules satisfy the following equations, respectively.

$$\begin{aligned} V_i(t+1) &= \omega \times V_i(t) + c_1 \times rand() \times (pbest_i(t) - X_i(t)) \\ &\quad + c_2 \times rand() \times (gbest(t) - X_i(t)) \end{aligned} \tag{11}$$

$$X(t+1) = X_i(t) + V_i(t+1) \tag{12}$$

where $\omega$ is the inertia weight factor of the particle, and $rand()$ is a random number distributed between $[0, 1]$. $X_i(t)$, $V_i(t)$ and $pbest_i(t)$ represent the position, velocity and historical best position of the $i$th particle at the $t$ iteration, respectively. $gbest(t)$ means the global best position in the $t$ iteration. The parameters $c_1$ and $c_2$ are learning factors, and $t$ indicates the current number of iterations.

The basic idea of PSO can be described as follows. First, the population and parameters are initialized, then the loop starts until the end condition is met, and finally the particle with the best position is decoded. Calculating the fitness value of the particle, so as to obtain its historical best position and global best position, as well as updating the speed and position of the particle, is the core operation of the loop body.

PSO has the problems of early convergence and low quality of optimal solution. Therefore, the classical PSO is improved as follows. Chaos theory is used to initialize the population to improve the quality of the initial population. By dynamically adjusting parameters such as inertia weight and learning factor, the convergence process is accelerated in the early stage and the population diversity is maintained in the later stage. The Opposition-Based Learning (OBL) mechanism is used to search the corresponding reverse solutions, and the better solutions are retained through comparison and evaluation; therefore, the search scope is expanded. The crossover and mutation operations of genetic algorithm are used to increase the population diversity.

4.2. **Sin chaos initialization population.** Different from random behavior, chaotic behavior has better dynamic and statistical characteristics, which can ensure that chaotic variables pass through all states without repetition within a certain range [21]. Therefore, using chaos theory to initialize the population can better ensure the diversity of the population and reduce the probability of early convergence. The method of initializing the population using sin chaos [22] can be expressed as

$$\begin{cases} X_{n+1} = \sin \dfrac{2}{X_n}, \quad n = 0, 1, \ldots, N \\ -1 \le X_n \le 1, \quad X_n \ne 0 \\ X_0 \ne 0 \end{cases} \tag{13}$$

where $X_n$ represents the position of the $n$th particle in the population.

4.3. **Adaptive parameter adjustment.** Inertia weight $\omega$ affects particle velocity and is closely related to search efficiency and early convergence. At the early part of the iteration, $\omega$ should be larger to improve the search efficiency and explore more areas, while at the later stage, $\omega$ should be smaller to start a more detailed local search. Meanwhile, when the population diversity is insufficient, $\omega$ should be larger to avoid early convergence. Therefore, $\omega$ is dynamically adjusted according to the difference of iteration number and particle fitness, satisfying the following equations.

$$\omega = \omega_{\min} + \eta \left(1 - \frac{|\min(F(X_i(t-1)), F(X_i(t)))|}{|\max(F(X_i(t-1)), F(X_i(t)))|}\right) \tag{14}$$

$$\eta = (\omega_{\max} - \omega_{\min}) \left(1 - \frac{t}{t_{\max}}\right) \tag{15}$$

where $F(X_i(t))$ is the fitness of particle $i$ during the iteration of round $t$, $\omega_{\max}$ and $\omega_{\min}$ represent the maximum and minimum weights respectively, and $t_{\max}$ denotes the maximum number of iterations.

Learning factors $c_1$ and $c_2$ also seriously affect the performance of PSO. [23] points out that in the early stage of PSO iteration, $c_1$ should be appropriately increased and $c_2$ should be decreased, and in the later stage, it is just the opposite. Thus, the update strategies of $c_1$ and $c_2$ should satisfy the following equations.

$$c_1(t) = c_{1\min} + \left(\frac{t_{\max} - t}{t_{\max}}\right) + (c_{1\max} - c_{1\min}) \tag{16}$$

$$c_2(t) = c_{2\min} + \left(\frac{t_{\max} - t}{t_{\max}}\right) + (c_{2\max} - c_{2\min}) \tag{17}$$

where $c_{1\max}$ and $c_{1\min}$ indicate the maximum and minimum values of $c_1$ respectively, and $c_{2\max}$ and $c_{2\min}$ indicate the maximum and minimum values of $c_2$ respectively.

4.4. **Opposition-based learning strategy.** Based on the current goal, OBL [24] uses the OBL mechanism to search for the corresponding opposite solution, and retains the better solution through comparison and evaluation. By introducing OBL into PSO, $gbest(t)$ can be optimized, so as to expand the search scope and improve the search efficiency. The OBL strategy to search for the opposite solution should satisfy the following equation.

$$X'_{best}(t) = ub + r \oplus (lb - X_{best}(t)) \tag{18}$$

where $X_{best}(t)$, $X'_{best}(t)$ are the optimal solution in the $t$ generation and its opposite solution, $ub$ and $lb$ respectively represent the upper and lower bounds of the particle position, and $r$ is the D-dimensional (that is, the spatial dimension of the particle) vector that obeys $(0, 1)$ uniform distribution. Once the opposite solution is found, the optimal solution is updated by the rule, which should satisfy the following equation.

$$X_{best}(t) = \begin{cases} X_{best}(t), & f(X_{best}(t)) > f(X'_{best}(t)) \\ X'_{best}(t), & f(X_{best}(t)) < f(X'_{best}(t)) \end{cases} \tag{19}$$

where $f(X)$ represents the fitness of particle $X$.

4.5. **Population updating by fusion genetic algorithm.** PSO updates the population through position transitions, while Genetic Algorithms (GA) update the population through natural selection, genetic variation and crossover. Integrating GA into PSO to carry out cross mutation for particles with low fitness can improve the global convergence ability [25].

Let the fitness of the $i$th particle in the population be $f_i$, $f_{\max}$ and $f_{\min}$ denote the maximum and minimum fitness values, respectively. The crossover probability $p_c$ and genetic probability $p_m$ are calculated using Equations (20) and (21), that is,

$$p_c = \frac{f_{\max} - f_i}{f_{\max} - f_{\min}} \tag{20}$$

$$p_m = \frac{f_i - f_{\min}}{f_{\max} - f_{\min}} \tag{21}$$

Set the threshold $p_l$. The two particles $X_i$ and $X_{i+1}$ of $p_c > p_l$ are crossed to obtain new particles $\hat{X}_i$ and $\hat{X}_{i+1}$. First, use Equation (22) the roulette strategy to select the parent particles, perform the particle crossing operation, and then return the new particles generated by the crossing into the population.

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{22}$$

where $p_i$ represents the probability that particle $i$ is selected. The crossover operation satisfying the following equations.

$$\hat{X}_i = rX_i + (1-r)X_{i+1} \tag{23}$$

$$\hat{X}_{i+1} = rX_{i+1} + (1-r)X_i \tag{24}$$

The mutation operation is performed on particles of $p_m > p_l$, satisfying the following equation.

$$\dot{X}_i = X_i^{\min} + r\left(X_i^{\max} - X_i^{\min}\right) \tag{25}$$

where $X_i^{\max}$ and $X_i^{\min}$ are the positions with the maximum and minimum fitness respectively, and $\dot{X}_i$ is the new particle generated.

4.6. **Improved PSO algorithm.** The improved PSO algorithm, denoted as GRPSO, is shown in Algorithm 1. Compared with the classical PSO, the chaotic strategy is used to initialize the population (Line3), the OBL mechanism (Line8) is introduced, the cross-variable strategy of GA is integrated (Line10-15), and the adaptive adjustment strategy of inertia weights and learning factors is used (Line17).

5. **Experiment Analysis.** In order to test the superiority of the proposed improved particle swarm optimization (GRPSO), three comparison algorithms are selected: the classical PSO, the particle swarm optimization algorithm (GA-PSO) [26] combined with genetic algorithm, and task offloading algorithm based on improved particle swarm optimization algorithm (TPSO) [27]. The effects of the number of tasks, the number of iterations, and the number of CLPs are considered.

5.1. **Cloudlet scenario and parameter setting.** The simulation experimental platform is IntelliJ IDEA2019, and the simulation experimental parameters are set as follows: The cloudlet scenario consists of several CLPs and a remote cloud. Each CLP deploys four servers as a cloudlet, and each server creates 30-50 virtual machine instances for task execution. The remote cloud has 2000 virtual machines. Two types of virtual machines are involved, which are used to execute DT and DS tasks, respectively. Referring to the instance resources provided by Amazon, the DT virtual machine is equipped with 4 virtual CPUs (vCPU) and 16GB memory and costs \$0.424; The DS virtual machine is equipped with 2 vCPUs and 8GB memory and costs \$0.212. The processing speed of the virtual CPU is 20000MIPS. The task type is randomly selected. The distance between the user and the associated CLP is 10-50 meters, the distance between the user and the cloud platform is 2000-5000 meters, and the distance between the CLPs is 200-500 meters.

**Algorithm 1. GRPSO**

**Input:** population size $N$; maximum iterations $maxGen$; threshold $p_l$
**Output:** The task scheduling resolution

```
for each particle
    Parameter initialization;
    Initialize the position and velocity of particles by chaotic strategy;
    Calculate fitness, pbest and gbest;
end for
while t ≤ maxGen do
  for i ← 1 to N
      Search opposite solution by OBL and update gbest;
      Update the position and velocity of particles;
      Calculate p_c and p_m using Equation (20) and Equation (21);
      if p_c > p_l then
          Generate new particles using Equation (23) and Equation (24);
      if p_m > p_l then
          Generate new particles using Equation (25);
      Update population by the roulette strategy;
  end for
      Adjust parameters dynamically using Equations (14)-(17);
  t = t + 1;
end while
return gbest;
```

The size of DS task is 1000-2000MI, and the expected delay is 50-100ms. The size of DT task is 500-1000MI, and the expected delay is 200-400ms. $\gamma = 0.5$. Quotation descent gradient $\beta = 3$. The delay sensitive factor $\alpha$ is 0.6-0.8.

The parameters associated with GRPSO are set as follows: $c_{1\max} = 2.5$, $c_{1\min} = 0.5$, $c_{2\max} = 2.5$, $c_{2\min} = 0.5$, $\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$, $p_l = 0.8$. The population size is 100.

5.2. **Effect of the number of tasks.** Figure 2 shows the total system profit obtained by the four algorithms when the number of CLPs is 10, the number of iterations is 150, and the number of tasks varies between 100-1000. As can be seen from Figure 2, the total profits obtained by PSO, TPSO, GA-PSO and GRPSO increase with the increase of the number of tasks. When the number of tasks is small, the difference among the total profits obtained by the four algorithms is relatively small. However, when the number of tasks reaches 500, the total profits gained by GRPSO, TPSO and GA-PSO are significantly higher than that of PSO. This is because the global optimization ability of the algorithm is positively correlated with the total system revenue. The traditional PSO is easy to fall into local optimum because of its poor search accuracy. TPSO and GA-PSO introduce simulated annealing algorithm and genetic algorithm respectively to improve the traditional PSO, and have the ability to jump out of the local optimum, so as to search for better scheduling strategy and obtain better profits. GRPSO can obtain a more uniform distribution of particles when initializing the population. At the same time, the search efficiency and accuracy of GRPSO can be effectively improved by improving the particle quality of the population during the iteration process. When the number of tasks reaches 1000, the total profit value obtained by GRPSO is 38%, 20% and 11% higher than that of PSO, TPSO and GA-PSO, respectively.

Figure 3 shows the effect of the number of tasks on the average delay, and results show that the average delay increases with the increase of the number of tasks. This is because,

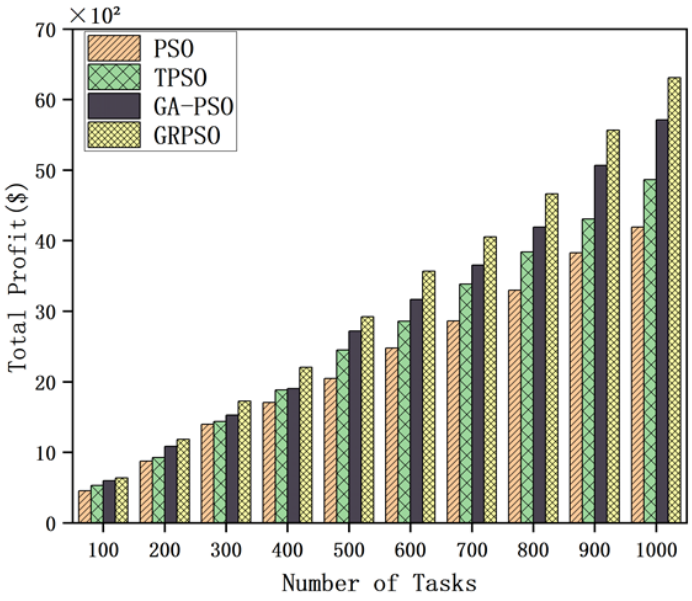



FIGURE 2. Effect of the number of tasks on the total profit

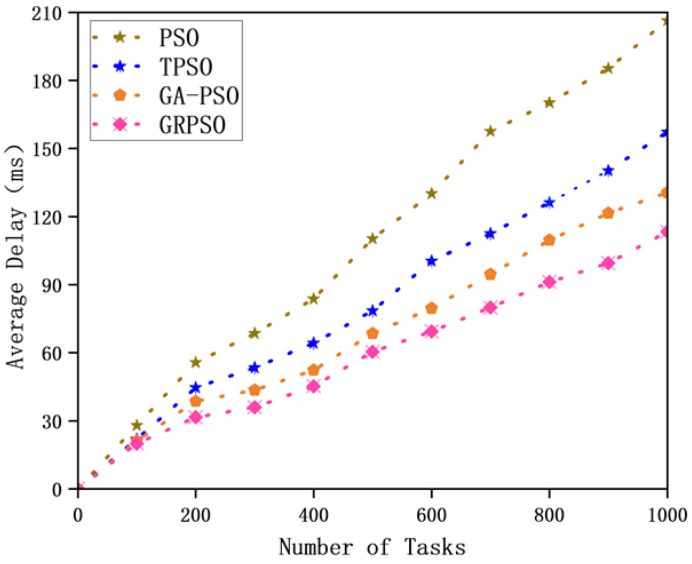



FIGURE 3. Effect of the number of tasks on the average delay

with the increasing number of tasks, some tasks have to be scheduled to the CLPs of the federation or even to the remote cloud, which pushes up the average delay. The results in Figure 3 show that GRPSO has obvious advantages over the other three algorithms. When the number of tasks reaches 1000, the average response delay obtained by GRPSO is reduced by 45%, 28% and 15% compared with PSO, TPSO and GA-PSO, respectively.

5.3. **Effect of the number of iterations.** In order to explore the effect of the number of iterations on the performance of the algorithm, the number of iterations is 10-200, the number of tasks is 500, and the number of CLPs is 10.

Figure 4 compares the changes of the optimal fitness (corresponding to the total profit) obtained by the four algorithms with the number of iterations. By observing Figure 4, it can be found that GRPSO can obtain obvious advantage when the number of iterations is low, which is largely due to the chaotic population initialization strategy and

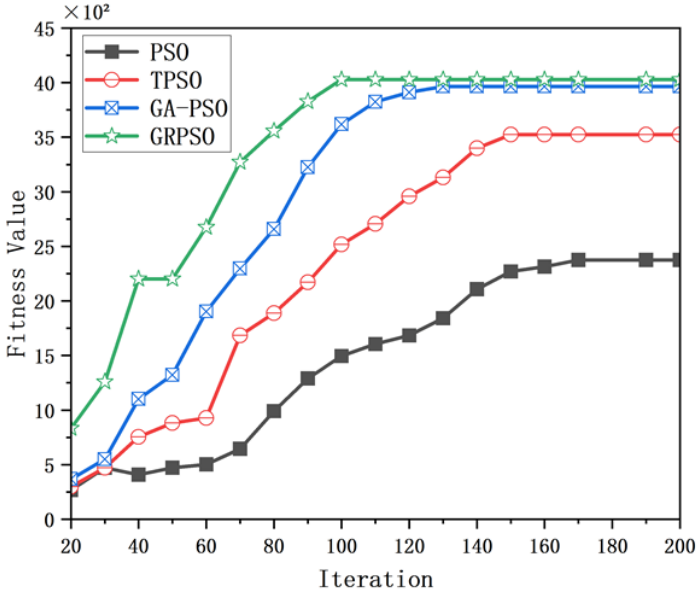



FIGURE 4. Effect of the number of iterations on fitness

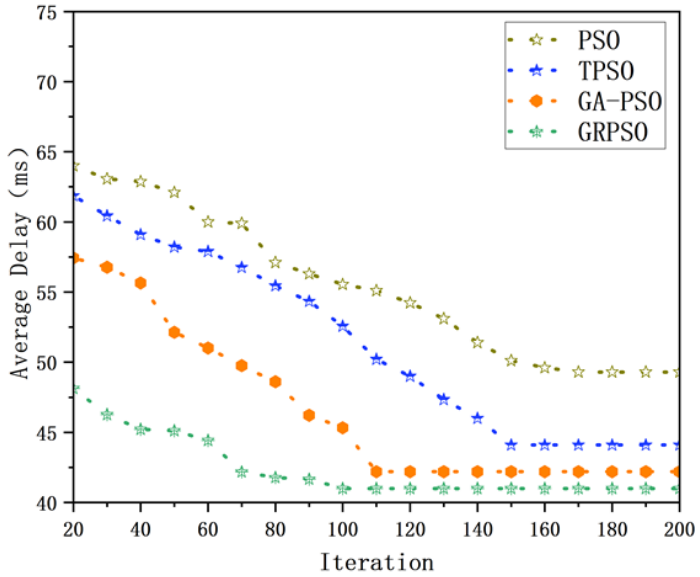



FIGURE 5. Effect of the number of iterations on the average delay

reverse learning strategy. GRPSO gradually enters the convergence state after about 100 iterations, while GA-PSO, TPSO and PSO require about 120, 150 and 170 iterations, respectively. When converging, the optimal fitness value obtained by GRPSO is better than the other three algorithms.

Figure 5 compares the changes of the average delay of the four algorithms with the number of iterations. The number of iterations required when the average delays of the four algorithms reaches the optimal is shown in Figure 5. The convergence in Figure 5 is consistent with the number of iterations required by the convergence of the optimal fitness of the four algorithms shown in Figure 4. The results in Figure 5 show that after the average delay converges (when the number of iterations reaches 150), the average task response delay obtained by GRPSO is reduced by 19%, 10% and 2% compared with PSO, TPSO and GA-PSO, respectively.

5.4. **Effect of the number of CLPs.** Figure 6 and Figure 7 respectively show the total profit and average delay obtained by the four algorithms when the number of tasks is 500, the number of iterations is 150, and the number of CLPs increases from 3 to 21 with step size 3. As can be seen from Figure 6, when the number of CLPs increases from 3 to 9, the total profits obtained by the four algorithms increase with the increase of the number of CLPs. However, when the number of CLPs continues to increase, the total profits remain unchanged on the whole. This is because there is a serious surplus of cloudlet resources at this time, and there is no need to schedule tasks to the remote cloud, the price of leasing resources between CLPs will also decline, and the acquisition of resource leasing will also decline. On the whole, the total profit that GRPSO can obtain is significantly higher than the other three algorithms.

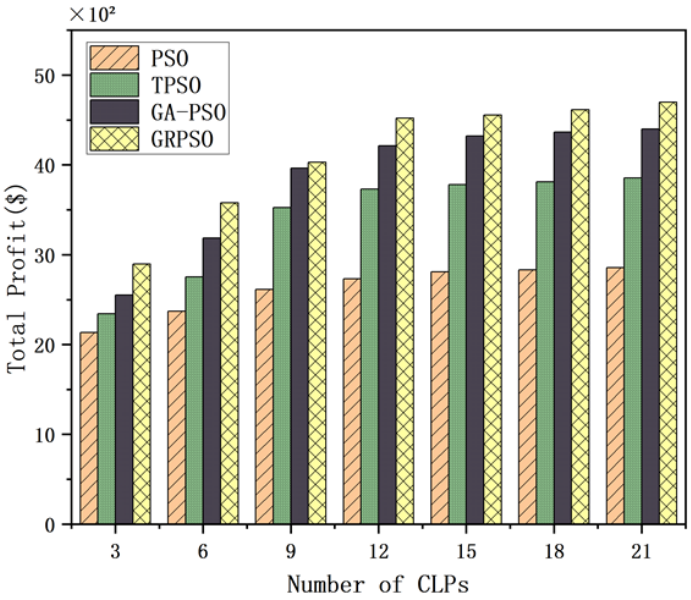



FIGURE 6. Effect of the number of CLPs on total profit

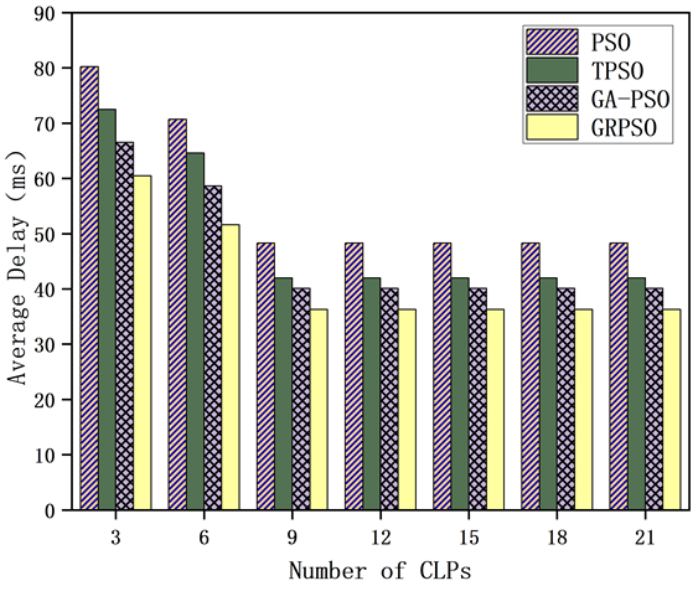



FIGURE 7. Effect of the number of CLPs on the average delay

We evaluated the impact of the number of different CLPs in the federation on the average delay. As shown in Figure 7, when the number of CLPs increases from 3 to 6, the average task delay has a significant downward trend, and then, with the increase in the number of CLPs, this downward trend gradually moderates. This is because, with the increase in the number of CLPs, the federation also has more resources for processing the task of the user, so there are more choices when executing tasks so that CLPs can obtain higher profits and further reduce task latency. However, considering the combined optimization of CLP profit and task delay, this downward trend will only last for a while. We can also see that the average task delay of the GRPSO algorithm is significantly better than the other three algorithms.

6. **Conclusion.** In this paper, in order to solve the task migration and scheduling problems in CLF scenario, a task scheduling algorithm based on the improved particle swarm optimization algorithm is proposed. By changing the population initialization strategy, the superiority and diversity of the initial population are further improved. At the same time, the inertia weight factor is dynamically adjusted during the particle iteration process to accelerate the particle convergence speed. By using the crossover and mutation ideas of genetic algorithm and combining the reverse learning strategy, the search field of the algorithm can be expanded, and the possibility of leaving the local optimum can be further improved. In this way, the federation can get the task scheduling strategy that maximizes the system profit more efficiently. Finally, after simulation, compared with PSO, TPSO and GA-PSO algorithms, the GRPSO algorithm proposed in this paper has good performance in the average task delay and the total profit of the federation system.

In the future, the authentication process of sharing resources between CLPs in cloudlet federation will be further discussed, and the task migration strategy under dynamic changing environment will be studied by combining the reinforcement learning algorithm.

**Acknowledgment.** This research was supported by the Guangxi Key Research and Development Program [grant number 2023AB01252], the National Natural Science Foundation of China [grant number 62262011], and the Foundation of Guilin University of Technology [grant number GUTQDJJ2002018].

## REFERENCES

[1] M. Chen and V. C. M. Leung, Reprint of: From cloud-based communications to cognition-based communications: A computing perspective, *Computer Communications*, vol.131, pp.77-82, 2018.

[2] X. Kong, Y. Wu and H. Wang, Edge computing for Internet of Everything: A survey, *IEEE Internet of Things Journal*, vol.9, no.23, pp.23472-23485, 2022.

[3] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali et al., All one needs to know about fog computing and related edge computing paradigms: A complete survey, *Journal of Systems Architecture*, vol.98, pp.289-330, 2019.

[4] H. Baghban, C. Y. Huang and C. H. Hsu, Latency minimization model towards high efficiency edge-IoT service provisioning in horizontal edge federation, *Multimedia Tools and Applications*, vol.81, no.19, pp.26803-26820, 2022.

[5] H. Baghban, C. Y. Huang and C. H. Hsu, Resource provisioning towards OPEX optimization in horizontal edge federation, *Computer Communications*, vol.158, pp.39-50, 2020.

[6] X. Cao, G. Tang and D. Guo, Edge federation: Towards an integrated service provisioning model, *IEEE/ACM Transactions on Networking*, vol.28, no.3, pp.1116-1129, 2020.

[7] F. Huang, H. Ye and W. Hao, Cost-aware resource management based on market pricing mechanisms in edge federation environments, *The Journal of Supercomputing*, vol.79, no.6, pp.5939-5961, 2022.

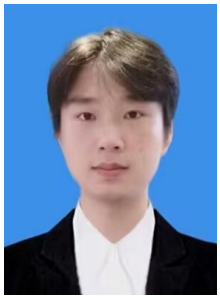
[8] H. Ye, Y. Chen and X. Li, Resource pricing model based on double auction for the cloudlet federation, *International Journal of Innovative Computing, Information and Control*, vol.17, no.1, pp.1-13, 2023.

[9] S. Latif, M. Z. Nayyer, I. Raza, S. A. Hussain, M. H. Jamal et al., Cloudlet federation based context-aware federated learning approach, *IEEE Access*, vol.10, pp.109153-109166, 2022.

[10] H. Liao, Z. Jia, Z. Zhou, Y. Wang, H. Zhang et al., Cloud-edge-end collaboration in air-ground integrated power IoT: A semidistributed learning approach, *IEEE Transactions on Industrial Informatics*, vol.18, no.11, pp.8047-8057, 2022.
[11] M. Z. Nayyer, I. Raza and S. A. Hussain, CFRO: Cloudlet federation for resource optimization, *IEEE Access*, vol.8, pp.106234-106246, 2020.
[12] W. Li, Q. Li, L. Chen, F. Wu and J. Ren, A storage resource collaboration model among edge nodes in edge federation service, *IEEE Transactions on Vehicular Technology*, vol.71, no.9, pp.9212-9224, 2022.
[13] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin et al., Cost-efficient workload scheduling in cloud assisted mobile edge computing, *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pp.1-10, 2017.
[14] U. Awada and J. Zhang, Edge federation: A dependency-aware multi-task dispatching and co-location in federated edge container-instances, *2020 IEEE International Conference on Edge Computing (EDGE)*, pp.91-98, 2020.
[15] H. Ye, J. Guo and X. Li, Delay-aware and profit-maximizing task migration for the cloudlet federation, *International Journal of Advanced Computer Science and Applications*, vol.13, no.10, pp.420-428, 2022.
[16] K. Liu, J. Peng, B. Yu, W. Liu, Z. Huang et al., An instance reservation framework for cost effective services in GEO-distributed data centers, *IEEE Transactions on Services Computing*, vol.14, no.2, pp.356-370, 2021.
[17] R. Lu, Y. Liang, Q. Ling, C. Li and W. Wu, Double auction and profit maximization mechanism for jobs with heterogeneous durations in cloud federations, *Journal of Cloud Computing*, vol.10, no.1, pp.1-22, 2021.
[18] S. Latif, M. Humayun, A. Sharif and S. Kadry, Resource discovery and scalability-aware routing in cloud federation using distributed meta-brokering paradigm, *International Journal of Web and Grid Services*, vol.18, no.1, pp.34-61, 2022.
[19] A. Hazra, M. Adhikari, T. Amgoth and S. N. Srirama, Stackelberg game for service deployment of IoT-enabled applications in 6G-aware fog networks, *IEEE Internet of Things Journal*, vol.8, no.7, pp.5185-5193, 2020.
[20] I. Sarkar, M. Adhikari, N. Kumar and S. Kumar, Dynamic task placement for deadline-aware IoT applications in federated fog networks, *IEEE Internet of Things Journal*, vol.9, no.2, pp.1469-1478, 2021.
[21] J. Ma, Y. Li, M. Hua and L. Xu, research of an adaptive chaotic artificial bee colony algorithm, *Journal of Electronics & Information Technology*, vol.47, no.5, pp.1060-1066, 2019.
[22] C. Li, Y. Zhao, Y. Li and S. Kong, An image encryption algorithm based on logistic chaotic mapping with sinusoidal feedback and its FPGA implementation, *Journal of Electronics & Information Technology*, vol.43, no.12, pp.3766-3774, 2021.
[23] J. A. Koupaei and M. Firouznia, A chaos-based constrained optimization algorithm, *Journal of Ambient Intelligence and Humanized Computing*, vol.12, pp.9953-9976, 2021.
[24] X. Zhu, P. Xia, Q. He, Z. Ni and L. Ni, Coke price prediction approach based on dense GRU and opposition-based learning salp swarm algorithm, *International Journal of Bio-Inspired Computation*, vol.21, no.2, pp.106-121, 2023.
[25] V. Anand and S. Pandey, New approach of GA-PSO? – Based clustering and routing in wireless sensor networks, *International Journal of Communication Systems*, vol.31, no.16, 2020.
[26] A. M. Manasrah and H. B. Ali, Workflow scheduling using hybrid GA-PSO algorithm in cloud computing, *Wireless Communications and Mobile Computing*, vol.2018, pp.1-16, 2018.
[27] Y. Miu, Y. Xu, W. Zhang, T. Liu and Z. Han, Improved particle swarm algorithm for task offloading in vehicular networks, *Application Research of Computers*, vol.38, no.7, pp.2050-2055, 2021.

## Author Biography

**Hengzhou Ye** received the B.S. and Ph.D. degrees from Guilin University of Technology and Guangxi University, Guangxi, China, in 2002 and 2019, respectively. He is currently a Full Professor with Guilin University of Technology. His research interests include mobile edge computing, multi-objective optimization, object detection and object tracking. He has published over 20 papers on well-known journals.



**Junhao Guo** received the B.S. and M.S. degrees from Luoyang Institute of Science Technology, Henan, China and Guilin University of Technology, Guangxi, China, in 2020 and 2023, respectively. His research interests include multi-objective optimization and mobile edge computing. He is currently a network optimization engineer at China United Network Communications Group Co., LTD.

**Xinxiao Li** received the B.S. and M.S. degrees from Guangxi University and Guilin University of Technology, Guangxi, China, in 2001 and 2007, respectively. She is currently a Lecturer in Guilin University of Technology. Her research interests include service composition, mobile edge computing, and affective computing. She has undertaken and participated in more than ten provincial key scientific research projects. She has published over 4 papers in journals and conferences.