Research on Task Scheduling Strategy under Mobile Cloud Computing Based on ICSO

Xuan Chen[1], Hongfeng Zheng[2*]

[1] Design and Art Branch, Zhejiang Industry Polytechnic College, China
[2] School of Mechanical and Electrical Engineering, Zhejiang Industry Polytechnic College, China
chenxuan1979@sina.com, sxzhf1966@163.com

Abstract

With the gradual application of mobile terminals such as cell phones in production and life, mobile cloud computing has become an important part of the internet. Different from traditional cloud computing task scheduling methods, mobile cloud computing task scheduling needs to consider not only task time minimization but also the lowest possible mobile device energy consumption. We propose an improved chicken swarm optimization (ICSO) algorithm applied to the task scheduling strategy under mobile cloud computing. First, we establish a multiobjective optimization strategy with minimum completion time and minimum energy consumption. Second, for the shortcomings of the chicken swarm optimization algorithm that easily fall into local optimums leading to algorithm stagnation, we use reverse learning initialization for the chicken flock population to expand the space of understanding and an adaptive strategy for learning factors and following coefficients. To illustrate the effectiveness of our algorithm in scheduling, we chose the number of mobile devices as 50, 100, and 150 and compared the improved chicken swarm optimization algorithm, ant colony algorithm, particle swarm algorithm, and chicken swarm optimization algorithm. The results illustrate that our proposed algorithm can reduce the task completion time, control the energy consumption of mobile devices well, and save energy.

Keywords: Mobile cloud computing, Time, Energy, Task scheduling

1 Introduction

With the rapid development of wireless networks and the widespread use of mobile devices, mobile internet is gradually playing an increasingly important role in people's lives and work. Users' needs and requirements for mobile terminal devices have also started to rise, and various large-scale applications have begun to appear, but mobile devices still have certain limitations due to the influence of the size and battery of mobile terminal devices and the hardware system itself. The emergence of the mobile cloud computing concept [1] solves such problems by reducing the requirements of applications for mobile terminal devices on the one hand and improving the performance of devices in handling large applications on the other. However, mobile cloud computing task scheduling is affected by the computational storage capacity of the device, the network connection status and stability, and the device power [2-3], so mobile cloud computing task scheduling is a fully NP problem. The metaheuristic algorithm has achieved good results in solving cloud computing task scheduling. We use the metaheuristic algorithm to solve the mobile cloud computing task scheduling problem by designing a fitness function applicable to the mobile cloud environment. Chicken Swarm Optimization is a new metaheuristic algorithm for population intelligence that was proposed by Chinese scholars Meng et al. in 2014 [4]. The algorithm takes the group behavior of roosters, hens, and chicks in the animal kingdom as the research object. There are three main bodies in the algorithm, the algorithm structure is simple, and the algorithm steps are easy to implement and are widely used in various engineering fields. We propose a new algorithm, improved chicken swarm optimization (ICSO), in which we introduce a backward learning strategy in the initialization of the population to improve the population diversity, enrich the number of solutions, optimize the learning factor and the following coefficient in the algorithm, avoid the algorithm falling into a local optimum, and improve the quality of the algorithm solution. The quality of the algorithm solution is improved. We use ICSO in task scheduling for mobile cloud computing, and simulations illustrate that the ICSO algorithm reduces task completion time by 32%, 30%, and 20% compared to ACO, PSO, and CSO, respectively, and reduces mobile device energy consumption by 22%, 21%, and 18%, respectively.

This paper is structured as follows: In Section 2, we describe the results of current scholars' research in terms of the task completion time of mobile cloud computing and the energy consumption of mobile devices, and find the direction of this paper's research from these results. In Section 3, we propose a new task scheduling model in a mobile environment that implements a scheduling model with the goal of reducing task completion time and reducing energy consumption of mobile devices. In Section 4, we explain the principle of the chicken flock optimization algorithm and propose optimization measures in terms of population, algorithm parameters, and coefficients for the shortcomings of this algorithm and explain how to apply it in mobile cloud task scheduling. In Section 5, we test the effectiveness of the optimized algorithm and verify it in terms of both reduced task time and reduced device energy consumption. Finally, we conclude the paper in Section 6.

*Corresponding Author: Hongfeng Zheng; E-mail: sxzhf1966@163.com
DOI: 10.53106/160792642022122307004

2 Related Work

The current research results are divided into two categories [5]. One is research on task scheduling only under a single mobile device [6-7], which usually considers only task offloading between a single mobile device and a cloud data center. The other category is the task allocation problem between multiple mobile devices [8-10], which considers multiple mobile devices (generally fewer mobile devices) to form an autonomous system with shared resources and equal mutual assistance.

Numerous research results have been published in academia for the task scheduling problem oriented to a single mobile computing device. C. Wang in [11] showed a dynamic optimization algorithm that minimizes the combined resource consumption of the CPU and bandwidth weighted sum as the optimization objective. The experimental results show that the algorithm can give the optimal task migration scheme. T. Soyata in [12] showed a Cloudlet application framework, MOCHA, which improves the QoS of task scheduling by dividing the set of different tasks. P. Balakrishnan in [13] believed that the task scheduling of mobile cloud computing needs to consider not only the energy consumption of mobile devices but also the energy consumption of the CPU in mobile devices at any time. S. Saha in [14] showed a migration decision algorithm for migrating tasks from mobile to cloud execution, which finally achieves the optimization of completion time. M. R. Ra in [15] showed an environment for incremental greedy policies for mobile cloud computing. Experiments illustrated the good results of this environment in terms of task latency reduction.

In the process of multimobile task device task distribution, as mobile devices need to consume more resources among themselves, reducing the energy consumption and completion time of the device becomes the main direction of optimization. In terms of energy consumption, Y. B. Li in [16] showed an energy-aware dynamic task scheduling algorithm with a dynamic voltage scaling technique, and experimental results showed that the algorithm can significantly reduce the energy consumption of mobile terminals. M. Nir in [17] showed the use of centralized agent nodes in mobile cloud computing to optimize the energy consumption of all mobile devices, which can reduce the energy consumption of mobile devices using minimal search. The reliability of this approach was verified in simulation experiments. A. Ali in [18] showed an energy-efficient dynamic decision-based method that can improve the decision-making ability of mobile devices during task offloading. In simulation experiments, this method was shown to reduce energy consumption. Chowdhury in [19] used mobile cloud computing in wireless sensors to reduce the energy consumption between nodes, and simulation experiments illustrated that using mobile cloud computing technology is indeed able to node process task energy consumption. P. Akki in [20] used the powerful performance of neural networks to reduce the energy consumption of mobile devices, and simulation experiments illustrated a 30.3% reduction in energy consumption. In terms of completion time, S. Ramu in [21] used the Capuchin search algorithm for task scheduling under mobile cloud computing, and simulation experiments showed significant advantages over existing state-of-the-art methods in terms of completion time, execution time, and resource utilization. M. G. Chen in [22] presented a robust computation offloading strategy with failure recovery in an intermittently connected cloudlet system, and simulation experiments illustrated the great advantage of this strategy in terms of task time reduction. B. Saemi in [23] showed the flow of the water cycle algorithm, and simulation experiments indicated that the algorithm reduces task completion time by 23%, 28%, and 21% compared to the GA, ACO, and PSO algorithms, respectively. V. Sundararaj in [24] showed a hybrid queueing ant colony-artificial bee colony optimization (Ant-Bee)-based algorithm for optimizing task assignment in MCC environments, and simulation experiments illustrated that this algorithm outperforms other algorithms in terms of power consumption, average task completion time, and dropout rate of mobile devices. H. Peng in [25] applied the whale optimization algorithm to the dynamic voltage-frequency scaling technique to better schedule the execution location of tasks. Simulation experiments showed that the algorithm can reduce the task completion time by 29% and the energy consumption of mobile devices by 13% compared to the WOA.

In other aspects of mobile cloud computing such as mechanisms, P. Nawrocki in [26] showed a new security-aware task allocation model strategy in mobile cloud computing. This approach used machine learning methods to predict resource utilization and select the best security service for task execution, such as neural networks. S. P. Wen in [27] showed a volume set kernel operation method in neural networks that provides a reference for related computations under mobile cloud computing.

3 Design of Mobile Cloud Computing Task Scheduling Model

In the mobile cloud computing environment, the nodes executing task assignment consist of a large number of terminal devices with mobile characteristics, and the computational storage energy, stability, and device energy consumption of these devices become constraints for task scheduling, which seriously affects the task scheduling efficiency. We construct a multiobjective optimization model based on the task scheduling model for mobile cloud computing described in [28], which is based on the completion time and energy consumption of mobile devices in task scheduling and can reasonably schedule tasks for each mobile device through a metaheuristic algorithm to minimize task completion time and reduce device energy consumption. A set of task resources is used to represent a subtask that can be processed in parallel after the job submitted by the user is divided, and N set T is used to represent $T=\{T_1,T_2,...,T_N\}$. Each subtask can be executed in parallel and has no correlation. The M mobile devices involved in the execution of the task are selected and denoted by the set D as $D=\{D_1,D_2,...,D_M\}$. Therefore, the essence of task allocation in mobile cloud computing is how to complete the execution of N tasks in M mobile devices in the shortest time and with the least energy consumption.

(1) Task completion time function

Mobile resources for executing tasks in mobile cloud computing are dynamic and heterogeneous, and the same task may take different times to complete on different mobile devices because each device may have a different computing

power itself. In general, for D_j task T_i assigned to a particular device, the execution time $te_{i,j}$ is determined from length L_i of the task T_i and the CPU processing power C_j (MIPS) of that device, so the execution time is expressed as follows.

$$te_{i,j} = \frac{L_i}{C_j \times (1 - u_{cpu})}, \tag{1}$$

where u_{cpu} denotes the CPU utilization rate of the mobile device. Since mobile devices themselves have a certain amount of energy consumption, especially the CPU, even when it is idle, they will continue to generate energy consumption, which affects the time to perform tasks on mobile devices. Especially when the CPU utilization rate of the mobile device is large, the task execution time will be greatly increased, so setting u_{cpu} is mainly considered when using the CPU availability of the mobile device to calculate the task time.

The task completion time is related not only to the computing power of each mobile device but also to the network transmission capability of the mobile device. Different network bandwidths result in different data transmission times for the task mapping to the mobile device and the result return from the device. For task T_i assigned to mobile device D_j, the task mapping time and return time are ts_{ij} and tr_{ij}, respectively, which are mainly determined by input data sizes d_i^{in} and d_i^{out} and the network bandwidth of device B_j and are thus expressed as follows.

$$ts_{ij} = \frac{d_i^{in}}{B_j}. \tag{2}$$

$$tr_{ij} = \frac{d_i^{out}}{B_j}. \tag{3}$$

Thus, the time t_{ij} for D_j mobile device a to complete task T_i is expressed as follows.

$$t_{ij} = te_{ij} + ts_{ij} + tr_{ij}. \tag{4}$$

Therefore, the final completion time t_T for all tasks in the set of tasks $T = \{T_1, T_2, ..., T_N\}$ is the time used to complete the longest task among all tasks, i.e., t_T is expressed as follows.

$$t_T = \max_{ij} t_{ij}. \tag{5}$$

(2) Equipment energy consumption function

In mobile cloud computing, mobile devices are the key devices for task execution, but because they need to consume power, we must consider the power consumption of these mobile devices in the process of considering energy consumption under mobile cloud computing. Under the condition of its own power, the remaining power can be used to perform mobile cloud computing tasks. Reference [29] indicated that the power consumption of hardware is related to the usage rate of the hardware itself, and [30] stated that the energy consumption of a computer and the CPU are linearly related, expressed as follows.

$$E_{cpu} = \alpha_{cpu} \times \mu_{cpu} + \gamma_{cpu}, \tag{6}$$

where E_{cpu} denotes CPU power consumption, μ_{cpu} denotes CPU utilization, and α_{cpu} and γ_{cpu} are fixed coefficients.

In the mobile cloud environment, the main power-consuming hardware modules of the device are the CPU and memory modules, so the energy consumption generated by the mobile device D_j performing task T_i is as follows.

$$er_{ij} = \mu_{cpu} \times c_{cpu} \times te_{ij} + u_{mem} \times c_{mem} \times te_{ij}, \tag{7}$$

where μ_{cpu} and u_{mem} denote CPU usage and memory usage, respectively. c_{cpu} and c_{mem} denote the power consumption coefficients of CPU and memory modules, respectively, and te_{ij} is the execution time of task T_i on device D_j.

Data transfer between mobile devices and proxy servers also consumes energy, which is due to the different connection methods used by mobile devices, and therefore the power consumption of the devices varies. Since the energy consumption of data transfer is proportional to the size of the data transferred, the energy consumption caused by data transfer between devices in a mobile environment ed_{ij} is as follows.

$$ed_{ij} = d_i^{in} \times ts_{ij} \times c_n + d_i^{out} \times tr_{ij} \times c_n, \tag{8}$$

where d_i^{in} and d_i^{out} denote the input data size and output data size of task T_i, respectively; c_n is the power consumption factor of the network transmission module; ts_{ij} is the task mapping time; and tr_{ij} is the result return time.

The energy consumption e_{ij} of mobile resource device D_j to complete task T_i is expressed as follows.

$$e_{ij} = er_{ij} + ed_{ij}. \tag{9}$$

Therefore, for the set of tasks $T=\{T_1,T_2,...,T_N\}$, the final energy consumption of all tasks is the sum of the energy consumption of all tasks, i.e., e_T. The expression is as follows.

$$e_T=\sum_{ij}e_{ij}\ . \tag{10}$$

4 Improved Chicken Swarm Optimization Algorithm for Mobile Cloud Computing Task Scheduling

4.1 Chicken Swarm Algorithm

The backpropagation neural network is one of the most widely used neural network models and mainly uses backpropagation for multilayer feedback training network models. It can achieve mapping capability from input to output, and it is widely used for pattern recognition, data prediction, and fault identification. However, the BP neural network has certain limitations: it lacks simple and effective parameters, resulting in the lack of stability of the BP algorithm. In addition, BP neural networks have local minimization and slow convergence, and the global optimum must be found by resetting the initial parameter values, which increases the algorithm running time.

The chicken swarm optimization algorithm is a new algorithmic idea designed to simulate group behavior generated by chickens in the animal kingdom during the process of obtaining food. Like most metaheuristic algorithms, it has the characteristics of a swarm intelligence optimization algorithm in that it mainly uses the collaborative idea of local solution between individuals and overall solution between groups and populations to obtain the optimal solution of the algorithm. Therefore, the algorithm has the characteristics of fast convergence and strong merit-seeking ability. The algorithm establishes the existence of different hierarchical classifications and ways of finding food in the chicken flock. It decomposes the whole swarm into several subflocks, and in each small flock there is a rooster and several hens and chicks. Different subflocks have different competitive relationships with each other due to different hierarchical classifications. The specific process is as follows.

(1) In the whole chicken swarm optimization algorithm, we divide into several different subgroups, but each subgroup contains a rooster and many hens and chicks.

(2) We call the individual with the best fitness value in each subgroup a rooster, the individual with average fitness value a hen, and the individual with the worst fitness value a chick. The hen is free to choose any subgroup, and there is no fixed relationship between the two, i.e., the hen and the chicks can lead the chicks, and the chicks can leave the current hen at any time.

(3) The hierarchical classification of the Chicken swarm, the dominance relationship and the hen-chick relationship remain unchanged until after the update;

(4) In each subgroup, individuals searched for food around the rooster in the group while preventing other individuals from taking their own food. Chicks were able to steal food from other individuals randomly, each chick followed the hen in searching for food, and some individuals in the Chicken swarm had the competitive advantage of good dominance and were able to obtain food first.

By analyzing the above rules, we have a clear understanding of the three types of individual chickens in the flock algorithm. In the algorithm, the rooster is the most important individual whose task is to find food, and the final position of the rooster is the optimal solution of the algorithm. As the rooster in the subgroup, the influence of the hen and the chick is significant, so we need to deal with the relationship between the hen and the chick to make the rooster obtain the best position in the subgroup. Therefore, we set all individuals in the subgroup algorithm as N, and the position of the chicken individual $x_{i,j}(t)$ denotes the position of the i th individual in the j -dimensional space in the t th iteration. The corresponding location of the rooster is updated as follows:

$$x_{i,j}(t+1)=x_{i,j}(t)\times(1+Randn(0,\sigma^2))\ , \tag{11}$$

$$\sigma^2=\begin{cases}1 & if\ \ f_i\le f_k\\ \exp(\dfrac{f_k-f_i}{|f_i|+\varepsilon}) & otherwise\end{cases}, \tag{12}$$

where $Randn(0,\sigma^2)$ is the mean value of 0, σ^2 is a Gaussian distribution, ε is a small constant, and k represents another individual among all roosters.

The formula for updating the position of individual roosters in the swarm optimization algorithm is as follows.

$$x_{i,j}(t+1)=x_{i,j}(t)+c_1\times rand\times(x_{r1,j}(t)-x_{i,j}(t))+c_2\times rand\times(x_{r2,j}(t)-x_{i,j}(t))\ . \tag{13}$$

$$c_1=\exp((f_i-f_{r1})/abs(f_i)+\varepsilon)\ . \tag{14}$$

$$c_2=\exp(f_{r2}-f_i)\ . \tag{15}$$

In Eq. (13), c_1, c_2 denotes the learning factor in the hen position update, $rand$ denotes any random number with a random value. r_1 denotes the corresponding rooster in the flock of the i th hen, r_2 denotes any randomly selected individual among the roosters and hens in the flock, and $r_1\neq r_2$, in Eq. (14-15), $\exp()$ denotes the learning factor function, $abs()$ denotes the absolute value function, ε is a fixed parameter to prevent the denominator from being 0, f_i denotes the individual fitness value, f_{r1} denotes the fitness value of the roosters in this subgroup, and f_{r2} denotes the fitness value of any rooster and hen in all groups.

The corresponding position of the chicks in the subgroup is expressed as follows:

$$x_{i,j}(t+1)=x_{i,j}(t)+F\times(x_{m,j}(t)-x_{i,j}(t))\ . \tag{16}$$

In Equation (16), we use $x_{m,j}(t)$ to denote the m th hen corresponding to the i th chick, F to denote the following coefficient, and $x_{m,j}(t)-x_{i,j}(t)$ to denote the hen leading the chick in search of food.

In this paper, the purpose of the model we construct is to reduce the task execution time and the energy consumption generated by the mobile device corresponding to the task, with the two objectives of minimum task completion time and minimum mobile device energy consumption as the goals of the optimization of the algorithm in this paper. Since the completion time adaptation function and the system equipment energy consumption adaptation function have different values and are both nonlinear functions, the normalization method is used to adjust them so that the adjusted function values are in the range of [0,1]. Let $f(x)$ denote the task completion time function t_T or the total energy consumption function e_T of the system equipment, $f^{'}(x)$ denote the function after the normalization operation of $f(x)$, and let $f_{\max}(x)$ and $f_{\min}(x)$ denote the maximum and minimum values of the function $f(x)$, respectively. Therefore, the expression of the $f^{'}(x)$ function is as follows.

$$f^{'}(x)=\frac{f(x)-f_{\min}(x)}{f_{\max}(x)-f_{\min}(x)}. \tag{17}$$

$$fitness_{ICSO}=\alpha\times f_t^{'}(i)+\beta\times f_e^{'}(i). \tag{18}$$

In Equation (18), $fitness_{ICSO}$ denotes the fitness function of the locust algorithm, and $f_t^{'}(i)$ and $f_e^{'}(i)$ are the normalized task completion time function and the system equipment energy consumption function, respectively. α and β are the weights of the two functions, and their sum is 1. Therefore, solving to obtain $\min fitness_{GoA}$ is the goal of the algorithm in this paper.

4.2 Improved Chicken Swarm Optimization Algorithm

Most metaheuristic algorithms suffer from the drawbacks of being locally trapped in the optimum and having slow convergence, which leads to stagnation of the algorithm. The chicken swarm optimization algorithm also suffers from these problems, and we propose corresponding solution strategies to avoid the shortcomings of the chicken flock optimization algorithm from three aspects: the population of the swarm, the learning factor, and the following coefficient.

(1) Population initialization

According to the description of the chicken swarm optimization algorithm, the initial solutions of the chicken swarm individuals are generated randomly, and the optimal solutions generated by the algorithm cannot be uniformly distributed in the space due to the randomness of the individuals easily, so that the solution performance of the algorithm decreases as the number of iterations keeps increasing and therefore reduces the performance of the algorithm. Inverse learning [31] is a machine learning strategy suitable for population optimization that usually obtains the reverse solutions of these current solutions in each iteration of the algorithm and selects the solutions from each current and reverse solution that are favorable to the evolution of the algorithm, further reducing the blindness in the search of the algorithm while expanding the search space of the solution by increasing the number of individuals of the reverse and current solutions. This improves the chicken flock optimization algorithm with regard to the quality of the solution and expands the scope of the global optimal solution generation. The algorithm process is as follows:

Step 1: Randomly generate the initial population. Randomly generate the initial population A of the chicken swarm algorithm, where the solutions of the corresponding individuals are obtained according to Equation (19):

$$x_i^j=x_{\min}^j+rand\times(x_{\max}^j-x_{\min}^j), \tag{19}$$

where W denotes the number of populations, D denotes the dimension, x_i^j denotes the i th individual in the j dimension, F ranges from [1, W], H ranges from [1, D], and $x_{\min}^j$ and $x_{\max}^j$ denote the upper and lower spatial boundaries, respectively.

Step 2: Solve for the inverse solution for each individual. In population NP_1, we find the inverse solution for each individual in this population according to Equation (20) to obtain the inverse population $NP_{op}=\{x_1^{'}(t),x_2^{'}(t),.....,x_W^{'}(t)\}$.

$$x_i^{'j}=x_{\min}^j+x_{\max}^j-x_i^j. \tag{20}$$

Step 3: Select the best individual in the population. First, we obtain a reverse population by the operations in steps 1 and 2. Second, we select the individual with the best fitness value, individual x_{best}, among the original population and the reverse population. At the same time, we mix all individuals of the two populations to find the average value of these individuals x_{mean}. We select the fitness value of the flock individual c and fitness value of the flock individual x_{mean} among the two. Finally, we select the optimal value of the fitness value of individual x_{best} and the fitness value of individual d to obtain the optimal flock individual x_{opbest} according to Equation (21):

$$x_{opbest}=\begin{cases}x_{best}, f(x_{best})>f(x_{mean})\\ x_{mean}, otherwise\end{cases}. \tag{21}$$

By introducing backward learning in the results of population initialization, we found that the chicken individuals

have more solutions after initialization and at the same time have a larger search space. This increases the number of other chicken individuals toward the optimal rooster individual position and improves the performance of the algorithm overall, thus providing a larger space of solutions to obtain the optimal solution.

(2) Learning factor optimization

In the chicken swarm optimization algorithm, the role of the learning factor reflects the relationship between the chicks following the hen in the subgroup. Therefore, the magnitude of the learning factor value reflects the extent to which the chicks learn from the hen and the rooster. By using Eqs. (14-15), we find that when the value of the learning factor is low, we allow the chick to search for a better solution in the current subgroup, and conversely, when the value of the learning factor is high, the chick can search in a larger range and thus can obtain the possibility of a better solution. However, we find from the formula that such a setting method of the learning factor tends to make the algorithm fall into a local optimum and thus cause the algorithm to stall. To avoid this situation from affecting the solution accuracy of the algorithm, we reset the two learning factors, and in the process of setting them, we relate them to the current number of iterations and the total number of iterations. Through the minimum and maximum factor values, we are able to dynamically adjust the learning factor values to guide the chick to a better solution.

$$c_1 = \begin{cases} c_1 + \dfrac{t_{\max} - t_{current}}{t_{\max}}, c_1 > c_{\min} \\ c_{\min} \qquad\qquad\qquad c_1 \le c_{\min} \end{cases}. \tag{22}$$

$$c_2 = \begin{cases} c_2 + \dfrac{t_{\min} + t_{current}}{t_{\min}}, c_2 < c_{\max} \\ c_{\max} \qquad\qquad\qquad c_2 \ge c_{\max} \end{cases}. \tag{23}$$

In Eqs. (22-23), the learning factor takes values in the range [$c_{\min}$, $c_{\max}$], while $t_{current}$ denotes the number of iterations currently being performed and takes values in the range [$t_{\min}$, $t_{\max}$].

(3) Follow factor optimization

In the chick's position update, the following coefficient is a fixed value; such a setting is not conducive to the chick's position update in each iteration, which makes the algorithm fall into a local optimum to some extent. To avoid this situation, the coefficient is set in this paper as follows:

$$F = (1 - \frac{t_{current}}{t_{\max}}) \times \frac{f_i}{f_{obj}}. \tag{24}$$

In Equation (24), we find that the following coefficient is related to the current number of iterations ($t_{current}$), total number of iterations ($t_{\max}$), fitness value of the i th individual (f_i), and fitness (f_{obj}) value of the optimal individual of the population. In Eq. (24), the value of the following coefficient F becomes gradually smaller as the number of iterations gradually becomes larger to ensure that the hen does not lead the chick all the time and to avoid the result of algorithm stopping due to falling into local optimum:

4.3 Algorithm Steps

The task flow of mobile cloud computing based on ICSO is shown in Figure 1.

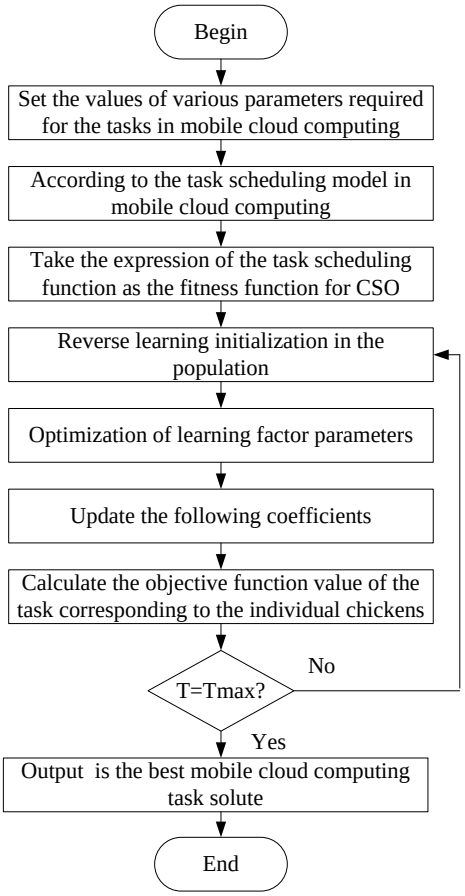

Figure 1. Flowchart

5. Simulation Experiments

In this paper, we use the resource-to-VM mapping mechanism in the C1oudSim resource pool to simulate the task assignment process from proxy servers to mobile devices in mobile cloud computing in order to realize the simulation effect of the task assignment algorithm proposed in this paper. The ant colony algorithm (ACO), particle swarm algorithm (PSO), and CSO are selected as the comparison algorithms. The hardware environment is as follows: CPU as Core I7, memory as 8 GB, hard disk capacity of 1 T, and software environment of Win10. The main parameters of the mobile devices, tasks, and other mobile devices in this experiment for mobile cloud computing are listed in Table 1. Table 2 lists the values of the main parameters for the four algorithms.

Table 1. Main parameters of mobile cloud computing

	Parameter	Value
Mobile devices	CPU processing power (MIPS)	2000-10000 (Step 500)
	Network transfer capability	100-1000 (Step100)
	CPU utilization rate	0.1-0.9 (Step 0.1)
	CPU power consumption factor	3.0
Task	Task length	10000-30000 (Step 2000)
	Transfer data	100-1000 (Step 100)
Other	Memory power consumption factor	2.5
	Network transfer power consumption factor	4G: 5 WiFi:1

Table 2. Main parameters of contrast algorithm

Algorithm	Main parameters
ICSO	$c_{\min}$ value is 0.1, $c_{\max}$ value is 1, ε value is 0.01, $rand$ value is 0.5, F value is 1
CSO	ε value is 0.01, $rand$ value is 0.5, F value is 1
ACO	The value of pheromone of individual ant colony is 0.005, the value of pheromone volatility coefficient in the path is 0.01, and the probability of path selection is 0.5
PSO	The value of inertia weights of individual particles is 0.5, the array of two random learning factors is 0.5, and the value of random number weights is set to 0.5

5.1 Algorithm Performance Comparison

In this subsection, we verify the performance effect of the ICSO algorithm. To better test the performance of the ICSO algorithm, we tested ICSO with ACO, PSO, and CSO in 2-dimensional, 5-dimensional, 10-dimensional and 30-dimensional arrangements under nine benchmark test functions, as shown in Table 3. The number of iterations is set to 1000, the data index results are shown in Table 4 to Table 9, and the test index is the maximum value, minimum value average and standard value.

Table 3. Test function

No	Test function
F1	$\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$
F2	$20\exp(-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))$
F3	$\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$
F4	$\sum_{i=1}^{n}([x_i+0.5])^2$
F5	$\frac{1}{1000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$
F6	$\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$

Table 4. F1 test function result

Algorithm	Dimension	Minimum Value	Maximum value	Mean	Standard deviation
ACO	2	1.151	98.901	40.892	44.350
	5	4.979	98.870	77.038	28.641
	10	28.874	99.636	90.603	12.575
	30	87.062	99.834	96.583	3.806
PSO	2	2.333E-06	1.116	1.007	1.017
	5	1.051	5.485	1.855	1.884
	10	5.196	19.512	11.652	3.667
	30	15.841	39.906	26.712	5.985
CSO	2	9.173E-08	2.349E-02	1.893E-03	4.220E-03
	5	2.922E-03	7.346E-02	2.728E-02	1.487E-02
	10	1.823E-02	9.975E-02	6.390E-02	2.171E-02
	30	6.550E-02	1.697E+01	1.333E+01	2.146E-02
ICSO	2	7.408E-11	4.882E-05	3.069E-06	8.231E-06
	5	2.448E-10	3.814E-05	1.846E-05	6.841E-05
	10	3.383E-10	1.600E-03	4.980E-05	2.352E-04
	30	2.187E-11	1.363E-03	5.199E-05	2.174E-04

Table 5. F2 test function result

Algorithm	Dimension	Minimum Value	Maximum value	Mean	Standard deviation
ACO	2	1.172	79962.931	74916.871	22733.531
	5	19.845	16782.762	23591.073	55016.221
	10	260.879	32373.592	45288.364	9965.661
	30	505.033	77681.35	14473.293	27452.502
PSO	2	4.536E-07	7.7018	0.249	2.102
	5	1.638	1731.353	165.484	361.100
	10	216.660	67022.206	8255.293	1345.592
	30	1352.401	29054.250	847467.368	6346.300
CSO	2	7.135E-10	2.314E+02	7.926E-03	2.812E-03
	5	8.137E-02	6.919E+02	3.912E+02	2.349E+01
	10	7.374E+02	1.716E+03	2.338E+02	3.318E+02
	30	2.312E+03	8.916E+04	2.816E+03	2.713E+03
ICSO	2	1.601	1.989	1.655	1.093
	5	4.598	4.9743	4.891	1.106
	10	9.498	10.069	9.875	1.095
	30	29.493	29.848	28.704	1.046

Table 6. F3 test function result

Algorithm	Dimension	Minimum Value	Maximum value	Mean	Standard deviation
ACO	2	1.033	1663.389	1698.460	4631.145
	5	1.316	3057.926	4916.712	9140.773
	10	611.799	5651.691	16125.421	17752.751
	30	26202.126	1326.816	6921.981	35142.120
PSO	2	3.242E-10	1.004	1.091	1.781
	5	1.000	13.333	2.124	3.120
	10	13.1117	560.9040	143.552	115.685

	30	1363.988	7473.098	3643.119	1311.625
CSO	2	5.704E-12	10.926E-04	7.431E-05	2.951E-04
	5	9.987E-04	3.929E-01	5.114E-02	5.675E-02
	10	10.370E-02	2.588E+01	5.122E-02	2.791E-02
	30	1.314E+01	1.214E+02	4.299E+01	2.124E+01
ICSO	2	8.001E-10	2.372E-03	1.578E-04	4.2013E-04
	5	5.947E-12	4.362E-02	1.590E-03	6.378E-03
	10	6.447E-07	1.770E+01	1.180E-02	2.834E-02
	30	2.840E-06	4.999E+00	4.621E-01	8.813E-01

Table 7. F4 test function result

Algorithm	Dimension	Minimum Value	Maximum value	Mean	Standard deviation
ACO	2	2.777	55.279	25.327	13.159
	5	33.047	133.306	81.638	18.575
	10	88.077	239.859	173.640	35.095
	30	484.405	645.319	544.770	38.159
PSO	2	7.323E-10	2.001	1.159	1.350
	5	4.005	23.881	9.273	5.629
	10	14.863	63.029	35.954	12.067
	30	120.847	264.788	201.047	28.585
CSO	2	1.603E-11	1.992E+01	3.064E-02	5.525E-02
	5	1.062E-06	2.009E+01	5.128E+00	3.697E+00
	10	8.033E-01	5.856E+01	3.281E+01	2.020E+01
	30	2.578E-01	3.215E+02	2.238E+02	7.525E+01
ICSO	2	4.552E-15	9.092E-05	4.813E-06	2.362E-05
	5	3.062E-06	2.815E+00	4.827E-02	3.566E-01
	10	2.065E-13	8.2853E-01	4.273E-02	2.438E-01
	30	9.343E-11	10.239E+00	4.920E-01	2.563E+00

Table 8. F5 test function result

Algorithm	Dimension	Minimum Value	Maximum value	Mean	Standard deviation
ACO	2	2.034E-07	5.310E-04	3.503E-05	8.763E-05
	5	7.609E-05	2.735E-02	6.880E-03	6.454E-03
	10	3.513E-03	1.347E+01	3.342E-02	3.094E-02
	30	1.378E-02	8.948E+01	2.487E+01	1.944E+01
PSO	2	1.983E-10	6.383E-42	4.172E-43	1.052E-43
	5	2.277E-32	7.883E-26	3.344E-27	2.025E-26
	10	9.473E-30	2.525E-22	6.799E-24	3.216E-23
	30	4.240E-28	3.511E-21	3.215E-22	6.730E-22
CSO	2	6.535E-11	1.188E-06	9.397E-07	2.203E-08
	5	8.429E-09	6.725E-04	9.290E-06	1.286E-04
	10	4.559E-05	6.795E+04	1.561E+04	1.781E+05
	30	1. 277E-33	1.173E-69	2.411E-71	1.659E-70
ICSO	2	3.876E-55	8.115E-47	1.856E-44	1.151E-45
	5	2.011E-52	6.021E-36	1.379E-40	8.523E-38
	10	5.665E-47	1.724E-35	4.873E-37	2.542E-37
	30	2.840E-06	4.999E+00	4.621E-01	8.813E-01

Table 9. F6 test function result

Algorithm	Dimension	Minimum Value	Maximum value	Mean	Standard deviation
ACO	2	5.432E-09	1.893E-05	1.957E-06	3.729E-06
	5	5.473E-08	3.095E-04	3.213E-05	5.294E-05
	10	1.622E-07	1.366E-03	1.702E-04	3.209E-04
	30	8.818E-06	3.191E-02	1.703E-03	4.590E-03
PSO	2	1. 671E-07	3.787E-11	3.663E-12	7.017E-12
	5	1.205E-13	5.942E-10	5.904E-11	1.115E-10
	10	1.182E-12	1.648E-09	2.508E-10	4.093E-10
	30	4.588E-12	2.867E-08	2.393E-09	5.060E-09
CSO	2	4.919E-07	1.555E-02	1.262E-03	2.466E-03
	5	1.125E-03	7.455E-02	2.771E-02	1.632E-02
	10	3.538E-02	1.011E+01	6.949E-02	1.609E-02
	30	1.036E+01	1.731E+01	1.367E+01	1.688E-02
ICSO	2	2. 367E-10	3.523E-16	2.404E-17	6.835E-16
	5	3.180E-13	5.060E-02	3.298E-03	5.992E-03
	10	8.375E-06	5.8473E-01	8.733E-02	2.017E-01
	30	4.884E-02	7.1964E-01	5.310E-01	2.520E-01

From the results in Table 4 to Table 9, it is found that the ICSO algorithm has better results in terms of the minimum, maximum mean, and standard values of the three test functions, and the PSO numerical results have obvious advantages compared to ACO. Compared with the CSO algorithm, ICSO also has certain advantages, especially when the dimension is 10,30. The results of this paper's algorithm in four indices illustrate that the performance of the chicken swarm optimization algorithm significantly improves after the initialization of the population, learning factor optimization, and subsequent coefficient optimization. This lays a foundation for the subsequent scheduling of mobile cloud computing tasks.

5.2 Task Completion Time

Figure 2 to Figure 4 compare the completion times of the four algorithms for different numbers of tasks when the number of mobile devices is 50, 100, and 150, respectively. In Figure 1, the difference between the four algorithms in terms of time required to complete the task with different numbers of mobile devices is insignificant. When the number of mobile devices is 100, there is a certain difference in the completion time of the four algorithms. When the number of mobile devices is 150, the difference in completion time between the four algorithms is more obvious, and ICSO has a better advantage over the other three algorithms in terms of completion time with reductions of 49.7%, 47.3%, and 30.9% compared to ACO, PSO, and CSO, respectively. This shows a significant improvement in the algorithm performance of ICSO after population initialization, learning factor, and subsequent coefficient optimization.

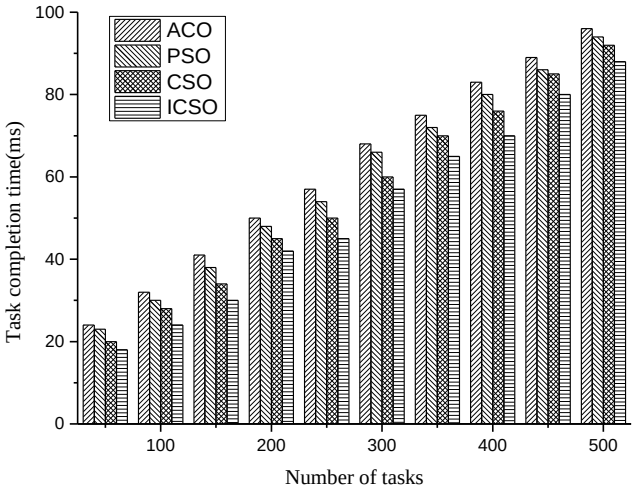

Figure 2. Number of tasks is 50

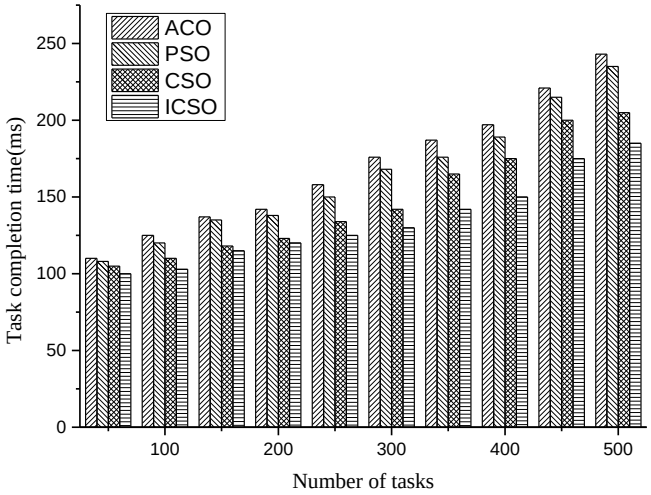

Figure 3. Number of tasks is 100

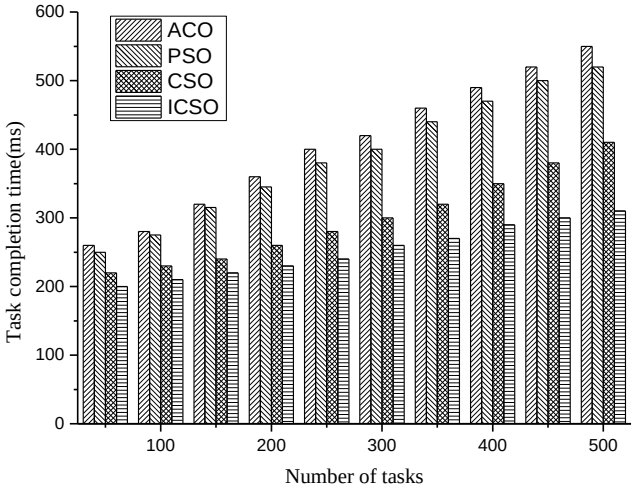

Figure 4. Number of tasks is 150

5.3 Task Completion Energy Consumption

Figure 5 to Figure 7 show the energy consumption of the four algorithms for different numbers of tasks when the number of mobile devices is 50, 100, and 150, respectively. The overall energy consumption of the four algorithms with 50 mobile devices is shown in Figure 4. The energy consumption of the four algorithms with different numbers of tasks is not much different. The overall energy consumption of the four algorithms with 100 mobile devices is shown in Figure 5. The values of energy consumption of the four algorithms vary greatly with the number of tasks completed. However, the overall effect of CSO is significantly lower than that of ICSO, which shows that our improvement measures for the algorithms are effective. From the comparison results, the algorithms in this paper reduce the energy consumption by 20%, 19%, and 12% on average compared to ACO, PSO, and CSO. The overall energy consumption of the four algorithms for 100 mobile devices is shown in Figure 6. The deviation of the energy consumption values of the four algorithms for completing different numbers of tasks is relatively large, and the energy consumption corresponding to the four algorithms increases with an increase in the number of tasks. The energy consumption values of the ACO and PSO algorithms are relatively large, while the energy consumption values of CSO and ICSO are relatively small, but CSO is significantly lower than ICSO in terms of overall effectiveness. The algorithms in this paper have an overall reduction of 40%, 35%, and 17% compared to ACO, PSO, and CSO.

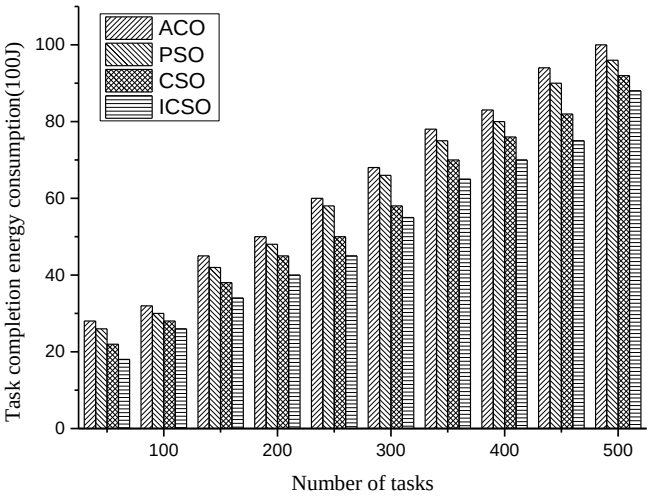

Figure 5. Number of tasks is 50

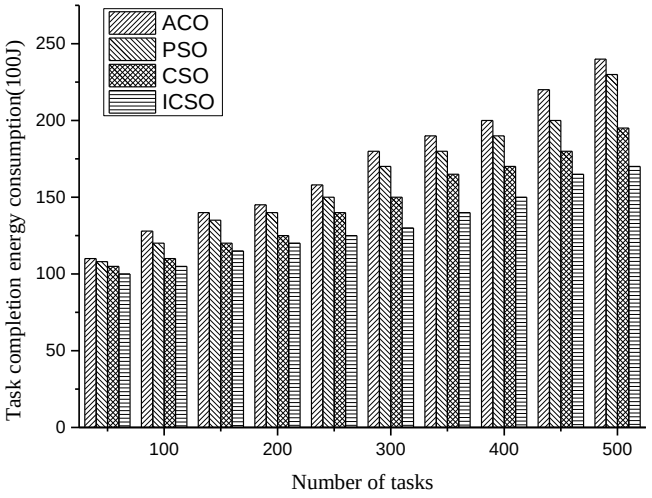

Figure 6. Number of tasks is 100

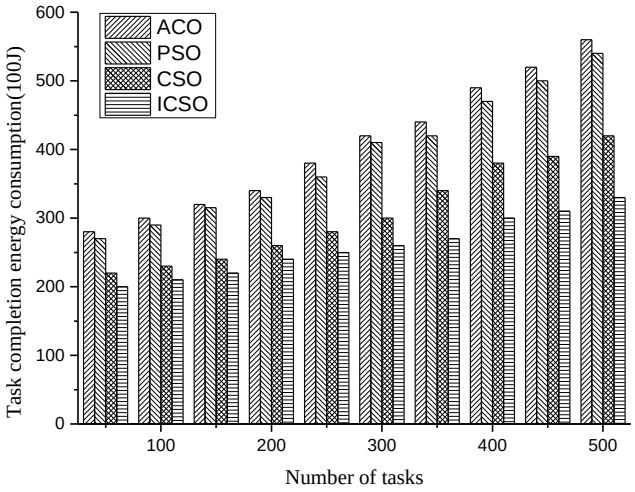

Figure 7. Number of tasks is 150

6. Conclusion

In this paper, we researched how to reduce task completion time and reduce the energy consumption of mobile devices. We constructed a multiobjective task scheduling scheme that minimizes the task completion time and energy consumption. To ensure smooth implementation of this scheme, we used the chicken flock optimization algorithm to complete the scheme. To counter the shortcomings of the chicken flock optimization algorithm in terms of convergence speed and low solution accuracy, we proposed three improvement ideas in terms of how to initialize the population, how to improve the learning factor, and how to optimize the following coefficient. To verify the effect of this improvement idea, we verified that the performance of the improved chicken flock optimization algorithm is good by benchmarking functions during the simulation, especially under the conditions of different numbers of mobile devices and different numbers of tasks. We compared the results of the four algorithms ICSO, ACO, PSO, and CSO in terms of reducing the completion time and reducing the energy consumption of the devices and verified that our proposed improvement strategy is effective. In future research, we will continue to focus on research related to task scheduling for mobile cloud computing.

References

[1] K. Kumar, Y. H. Lu, Cloud computing for mobile users: Can offloading computation save energy? *Computer*, Vol. 43, No. 4, pp. 51-56, April, 2010.

[2] A. Aliyu, A. H. Abdullah, O. Kaiwartya, S. H. H. Madni, U. M. Joda, A. Ado, M. Tayyab, Mobile cloud computing: taxonomy and challenges, *Journal of Computer Networks and Communications*, Vol. 2020, pp. 1-23, July, 2020.

[3] N. Parajuli, A. Alsadoon, P. W. C. Prasad, R. S. Ali, O. H. Alsadoon, A recent review and a taxonomy for multimedia application in Mobile cloud computing based energy efficient transmission, *Multimedia Tools and Applications*, Vol. 79, No. 41-42, pp. 31567-31594, November, 2020.

[4] X. B. Meng, Y. Liu, X. Z. Gao, H. Z. Zhang, A new bio-inspired algorithm: chicken swarm optimization, *International conference in swarm intelligence*, Hefei, China, 2014, pp. 86-94.

[5] N. Fernando, S. W. Loke, W. Rahayu, Mobile cloud computing: A survey, *Future generation computer systems*, Vol. 29, No. 1, pp. 84-106, January, 2013.

[6] H. S. Lee, J. W. Lee, Task offloading in heterogeneous mobile cloud computing: Modeling, analysis, and cloudlet deployment, *IEEE Access*, Vol. 6, pp. 14908-14925, March, 2018.

[7] G. A. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, J. Root, Cloudlet-based cyber-foraging for mobile systems in resource-constrained edge environments, *International Conference on Software Engineering*, Hyderabad, India, 2014, pp. 412-415.

[8] T. Verbelen, P. Simoens, F. D. Turck, B. Dhoedt, Adaptive deployment and configuration for mobile augmented reality in the cloudlet, *Journal of Network and Computer Applications*, Vol. 41, pp. 206-216, May, 2014.

[9] S. Bohez, T. Verbelen, P. Simoens, B. Dhoedt, Discrete-event simulation for efficient and stable resource allocation in collaborative mobile cloudlets, *Simulation Modelling Practice and Theory*, Vol. 50, pp. 109-129, January, 2015.

[10] C. M. S. Magurawalage, K. Yang, L. Hu, J. M. Zhang, Energy-efficient and network-aware offloading algorithm for mobile cloud computing, *Computer Networks*, Vol. 74, pp. 22-33, December, 2014.

[11] C. Wang, Z. Li, Parametric analysis for adaptive computation offloading, *Proceedings of the ACM SIGPLAN 2004 Conference on Programming Language Design and Implementation*, Washington, DC, USA, 2004, pp. 119-130.

[12] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, W. Heinzelman, Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture, *2012 IEEE symposium on computers and communications (ISCC)*, Cappadocia, Turkey, 2012, pp. 000059-000066.

[13] P. Balakrishnan, C. K. Tham, Energy-efficient mapping and scheduling of task interaction graphs for code offloading in mobile cloud computing, *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, Dresden, Germany, 2013, pp. 34-41.

[14] S. Saha, M. S. Hasan, Effective task migration to reduce execution time in mobile cloud computing, *2017 23rd International Conference on Automation and Computing (ICAC)*, Huddersfield, UK, 2017, pp. 1-5.

[15] M. R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, Odessa: enabling interactive perception applications on mobile devices, *Proceedings of the 9th international conference on Mobile systems, applications, and services*, Bethesda, Maryland, USA, 2011, pp. 43-56.

[16] Y. B. Li, M. Chen, W. Y. Dai, M. K. Qiu, Energy optimization with dynamic task scheduling mobile cloud computing, *IEEE Systems Journal*, Vol. 11, No. 1, pp. 96-105, March, 2017.

[17] M. Nir, A. Matrawy, M. St-Hilaire, An energy optimizing scheduler for mobile cloud computing environments, *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, 2014, pp. 404-409.

[18] A. Ali, M. M. Iqbal, H. Jamil, F. Qayyum, S. Jabbar, O.

Cheikhrouhou, M. Baz, F. Jamil, An efficient dynamic-decision based task scheduler for task offloading optimization and energy management in mobile cloud computing, *Sensors*, Vol. 21, No. 13, pp. Article No. 4527, July, 2021.

[19] M. Chowdhury, Time and energy-efficient hybrid job scheduling scheme for mobile cloud computing empowered wireless sensor networks, *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 37, No. 1, pp. 26-36, May, 2021.

[20] P. Akki, V. Vijayarajan, Energy efficient resource scheduling using optimization based neural network in mobile cloud computing, *Wireless Personal Communications*, Vol. 114, No. 2, pp. 1785-1804, September, 2020.

[21] S. Ramu, R. Ranganathan, R. Ramamoorthy, Capuchin search algorithm based task scheduling in cloud computing environment, *Yanbu Journal of Engineering and Science*, Vol. 19, No. 1, pp. 18-29, June, 2022.

[22] M. G. Chen, S. T. Guo, K. Liu, X. F. Liao, B. Xiao, Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing, *IEEE Transactions on Mobile Computing*, Vol. 20, No. 5, pp. 2025-2040, May, 2021.

[23] B. Saemi, M. Sadeghilalimi, A. A. R. Hosseinabadi, M. Mouhoub, S. Sadaoui, A New Optimization Approach for Task Scheduling Problem Using Water Cycle Algorithm in Mobile Cloud Computing, *2021 IEEE Congress on Evolutionary Computation (CEC)*, Kraków, Poland, 2021, pp. 530-539.

[24] V. Sundararaj, Optimal task assignment in mobile cloud computing by queue based ant-bee algorithm, *Wireless Personal Communications*, Vol. 104, No. 1, pp. 173-197, January, 2019.

[25] H. Peng, W. S. Wen, M. L. Tseng, L. L. Li, Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment, *Applied Soft Computing*, Vol. 80, pp. 534 -545, July, 2019.

[26] P. Nawrocki, J. Pajor, B. Sniezynski, J. Kolodziej, Modeling adaptive security-aware task allocation in mobile cloud computing, *Simulation Modelling Practice and Theory*, Vol. 166, Article No. 102491, April, 2022.

[27] S. P. Wen, J. D. Chen, Y. C. Wu, Z. Yan, Y. T. Cao, Y. Yang, T. W. Huang, CKFO: Convolutional kernel first operated algorithm with applications in memristor-based convolutional neural network, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 40, No. 8, pp. 1640-1647, August, 2021.

[28] L. Li, Z. L. Wang, X. H. Yang, Mobile resource reliability-based task allocation for mobile cloud, *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, Qinhuangdao, China, 2015, pp. 1746-1750.

[29] S. K. Choi, I. S. Cho, K. S. Chung, B. K. Song, H. C. Yu, Group-based resource selection algorithm supporting fault-tolerance in mobile grid, *Third International Conference on Semantics, Knowledge and Grid (SKG 2007)*, Xi'an, China, 2007, pp. 426-429.

[30] S. H. Jang, J. S. Lee, Mobile resource reliability-based job scheduling for mobile grid, *KSII Transactions on Internet and Information Systems (TIIS)*, Vol. 5, No. 1, pp. 83-104, January, 2011.

[31] S. Rahnamayan, H. R. Tizhoosh, M. M. A. Salama, Opposition-based differential evolution, *IEEE Transactions on Evolutionary computation*, Vol. 12, No. 1, pp. 64-79, February, 2008.

Biographies

Xuan Chen is an associate professor at Zhejiang Industry Polytechnic College. He received his master degree from University of Electronic Science and Technology of China. His research interests include cloud computing and algorithm design.

Hongfeng Zheng is a Professor at Zhejiang Industry Polytechnic College. He received his master's degree from Huazhong University of Science and Technology. His research interests include power electronics and applications, intelligent control, and algorithm research.