Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 7, July 2019, pp. 1929-1938
DOI: 10.23940/ijpe.19.07.p20.19291938

Chicken Swarm Optimization in Task Scheduling in Cloud Computing

Liru Han*

Zhejiang University of Water Resources and Electric Power, Hangzhou, 310018, China

Abstract

In order to solve the problem of low efficiency in resource scheduling in cloud computing, an improved chicken swarm optimization (CSO) is proposed for task scheduling. Firstly, the concept of opposition-based learning is introduced to initialize the chicken population and improve the global search ability. Secondly, the concepts of the weight value and learning factor in particle swarm optimization (PSO) are introduced to improve the positions of chickens, and the individual positions of chickens are optimized. Thirdly, the overall individual positions of the CSO are optimized by the difference algorithm. Finally, the possible cross-boundary of individual positions in the algorithm is prevented as a whole by boundary processing. In the simulation experiment, the optimized CSO is compared with the basic CSO, PSO, and ant colony optimization (ACO) in terms of completion time, cost, energy consumption, and load balancing, and good results are achieved.

Keywords: chicken swarm algorithm; opposition-based learning; learning factor; difference algorithm

(Submitted on March 12, 2019; Revised on May 15, 2019; Accepted on June 15, 2019)

© 2019 Totem Publisher, Inc. All rights reserved.

1. Introduction

Scheduling resources efficiently is the focus of cloud computing research at present [1]. Many researchers have applied various intelligent algorithms such as the genetic algorithm [2-3], particle swarm optimization (PSO) [4-5], and ant colony optimization (ACO) [6-7] to resource scheduling and achieved good results. In 2014, Meng put forward chicken swarm optimization (CSO) [8], which is a random optimization algorithm based on population. The advantage of this algorithm is that it is easy to implement, and the disadvantages are that it has slow convergence speed, low accuracy, and can easily fall into a local optimum and not get the optimum solution. Some scholars have studied the application of CSO. Reference [9] proposed to apply the improved CSO to the optimization of indexes of micro-grid. A good result was achieved, and the deficiency was that it was only compared with the PSO in the experimental part and not with the basic CSO, making the optimization result slightly less convincing. References [10-11] applied the improved CSO to the scheduling of workshop problem, thereby enhancing the scheduling efficiency overall and achieving good results. However, they lacked comparison with classical scheduling algorithms. Reference [12] proposed using CSO to optimize a multi-layer perception classifier based on deep neural network and obtained good results. Reference [13] proposed to use CSO to optimize the PSO and carried out relevant tests, which showed that the combined algorithm has good performance. Reference [14] proposed an improved hybrid meta-heuristic method (IRRO-CSO) based on improved raven roosting optimization (IRRO) and CSO. Simulation results showed that this algorithm has good scheduling effects. Reference [15] presented a new method called DDR (duplication and direct redistribution), and the DDR algorithm was shown to be easier to implement and can achieve very similar or better performance under different outer join workload. Thus, it can be considered as a new option for current data analysis applications.

Based on the above researches, this paper uses the improved chicken swarm optimization (ICSO) for the allocation of task resources in resource scheduling in cloud computing. It improves the algorithm's local and global search ability by initializing the algorithm population, updating the chicken position, and selecting the optimum and boundary processing with the difference algorithm.

* Corresponding author.
E-mail address: 623885818@qq.com

2. Description of Resource Scheduling of Cloud Computing

Task resource allocation for cloud computing is an NP-complete problem. This paper solves the resource allocation problem from the perspective of users' needs and resource owners. It proposes a resource scheduling model for cloud computing by introducing time, cost, and energy consumption constraints, which can meet the QoS requirements of users and effectively realize the load balancing of resources. Assume that there are n tasks sharing m resources, and each task S_i is composed of $k(i)$ parallel and interdependent subtasks. Assume that these subtasks have the same amount of computation, and each computing resource R_j has a fixed price P_j and energy consumption M_j. When multiple subtasks are assigned to R_j, the subtasks will share the computing power of R_j equally. The core problem of resource allocation is to solve an $n \times m$ matrix a, wherein rows denote tasks, columns denote resources, $a_{i,j}$ denotes the number of subtasks assigned by task S_i to resource R_j, and a_i denotes row i of matrix a, which satisfies the equation $\sum_{a_{ij} \in a_i} a_{ij} = k(i)$. Based on the same principle, three matrices of time, cost, and energy consumption can be obtained, designated as T, E, and U respectively. Element t_{ij} in matrix T denotes the time it takes resource R_j to complete $a_{i,j}$ subtasks assigned to it by task S_i. Since subtasks are completed in parallel, task S_i is completed in $\max\{t_{ij} \in t_i\}$. e_{ij} in matrix E denotes the cost for resource R_j to complete the $a_{i,j}$ subtasks assigned to it by task S_i, so the cost of task S_i is $\sum_{j=1}^{m} e_{ij}$. u_{ij} in matrix U denotes the energy consumption of resource R_j on the $a_{i,j}$ subtasks assigned to it by task S_i, so the energy consumption of task S_i is $\sum_{j=1}^{m} u_{ij}$. In general, weight values are set in the task-related indexes to denote the trade-off relationship between the three variables, which are denoted by ω_t, ω_e, and ω_u. Therefore, the total cost of task S_i can be obtained as $\omega_t \times \max_{t_{ij} \in t_i}\{t_{ij}\} + \omega_e \times \sum_{j=1}^{m} e_{ij} + \omega_u \times \sum_{j=1}^{m} u_{ij}$, which can be denoted by the following equation $u_i(a_i)$:

$$u_i(a_i) = \omega_t \times \max_{t_{ij} \in t_i}\{t_{ij}\} + \omega_e \times \sum_{j=1}^{m} e_{ij} + \omega_u \times \sum_{j=1}^{m} u_{ij} \tag{1}$$

Therefore, the goal of each task is to maximize its utility, i.e., minimize the cost, time, and energy consumption of task S_i, so the total utility of the task is:

$$u(a) = \min(u_i(a_i)) \tag{2}$$

Therefore, the best value of the utility of task resource scheduling in cloud computing is the best value of the total task utilities.

3. Chicken Swarm Optimization

Chicken swarm is a new bionics method that was developed by Xianbing Meng, Yu Liu, Xiaozhi Gao, and Hengzhen Zhang in 2014. This algorithm simulates the hierarchical level and behavior of chicken swarm (including cocks, hens, and chicks), which can effectively use swarm intelligence to solve optimization problems. Chicken swarm optimization (CSO) has superior performance in terms of accuracy and robustness [16]. CSO describes the optimization problem as the process of a chicken swarm searching for food, and it mainly simulates the hierarchical classification and foraging methods existing in a chicken swarm. The whole chicken swarm is divided into several groups, where each group is composed of a rooster, several hens, and chicks. Due to different hierarchical systems, there is a certain competition among different chicken swarms. The detailed process is as follows:

(1) There are several groups in the whole chicken swarm, and each sub-swarm is composed of a rooster, several hens, and chicks;

(2) In CSO, several chickens with the best fitness value are called roosters and are distributed into different groups. Chickens with the worst fitness values are taken as chicks, and chickens with medium fitness values are taken as hens. The

hens can choose which group to belong to at will, and the relationship between the hens and the chicks is also randomly established;

(3) The hierarchical classification, dominant relationship, and hen-chicken relationship in chicken swarms remain unchanged once they are established and do not change until they are updated;

(4) In each group, individuals search for food around the roosters in their group, while preventing other individuals from snatching their own food. However, chicks can steal food from other individuals at random. Each chick follows the hens in search of food. Some individuals in the swarm have a competitive advantage of a good dominant position and can get food in priority.

The individuals in the chicken swarm move according to their own rules until the best position is found. Therefore, individual positions in the chicken flock are regarded as solutions of the optimization problem, and finding the individual with the best position will be the optimal solution of the optimization problem. Assume that there are N individuals in the whole chicken swarm and each individual's position is denoted by $x_{i,j}(t)$, the value of which denotes the position of the i^{th} individual in the j^{th} dimension in the t^{th} iteration. Among the three different types of chickens in the whole chicken swarm, the individual position update in the chicken swarm varies for the different types. The roosters are the individuals with the best fitness values in the whole swarm and can search for food in a wider space.

The roosters' position update is as follows:

$$x_{i,j}(t+1) = x_{i,j}(t) \times (1 + Rand(0, \sigma^2)) \tag{3}$$

$$\sigma^2 = \begin{cases} 1, & \text{if } f_i \le f_k \\ \exp\left(\dfrac{f_k - f_i}{|f_i| + \varepsilon}\right), & \text{otherwise} \end{cases} \tag{4}$$

In this equation, the average value of $Rand(0, \sigma^2)$ is 0, σ^2 is a Gaussian distribution, ε is a small constant, and k denotes another individual in all roosters. Equation (5) is for the particle position in the standard PSO algorithm. In this paper, the hen corresponds to the particle, and the position of the hen is calculated by using the position formula of particle swarm.

The hens' position update is as follows:

$$x_{i,j}(t+1) = x_{i,j}(t) + c_1 \times rand \times (x_{r1,j}(t) - x_{i,j}(t)) + c_2 \times rand \times (x_{r2,j}(t) - x_{i,j}(t)) \tag{5}$$

$$c_1 = \exp((f_i - f_{r1})/abs(f_i) + \varepsilon) \tag{6}$$

$$c_2 = \exp(f_{r2} - f_i) \tag{7}$$

In this equation, $rand$ is 0.5, r_1 is the rooster individual of the group in which the i^{th} hen itself is located, r_2 is an arbitrary individual randomly selected from the roosters and hens in the whole chicken swarm, and $r_1 \neq r_2$.

The chicks' position update is as follows:

$$x_{i,j}(t+1) = x_{i,j}(t) + F \times (x_{m,j}(t) - x_{i,j}(t)) \tag{8}$$

In this equation, m denotes the hen corresponding to the i^{th} chick and F denotes the following coefficient, indicating that the chicks follow the hens to find food.

4. Scheduling of Improved Chicken Swarm Optimization in Cloud Computing

4.1. Initialization Selection

The population described in CSO is not initialized. As a result, the optimal solution of the algorithm cannot be guaranteed to

be evenly distributed in the space, thus limiting the efficiency of the algorithm and reducing the performance of the algorithm. Opposition-based learning [17] is an optimization strategy used in machine learning, where in each iteration of the algorithm, all the inverse solutions of these current solutions are obtained. A solution conducive to evolution is selected between the current solution and the inverse solution, reducing the blindness of the algorithm. In this paper, the opposition-based learning strategy [18] is adopted to expand the search space for population initialization and improve the global search ability of the algorithm. The process of the algorithm is as follows:

Step 1 Randomly select the CSO population $NP_1 = \{x_1(t), x_2(t), \cdots, x_N(t)\}$.

$$x_i(t) = (x_{i,1}(t), x_{i,2}(t), \cdots, x_{i,j}(t), \cdots, x_{i,D}(t)), \quad i \in N \tag{9}$$

Step 2 Find the reverse population $NP_{op} = \{\tilde{x}_1(t), \tilde{x}_2(t), \cdots, \tilde{x}_p(t)\}$ corresponding to NP_1, and find each body according to reference [16].

$$\tilde{x}_i(t) = (\tilde{x}_{i,1}(t), \tilde{x}_{i,2}(t), \cdots, \tilde{x}_{i,j}(t), \cdots, \tilde{x}_{i,D}(t)) \tag{10}$$

Step 3 Select the individual x_{best} with the best fitness value from $NP_1 \bigcup NP_{op}$. Calculate $x_{mean} = (x_1 + x_2 + \cdots + x_{2N})/2N$, which is obtained using Equation (11):

$$x_{opbest} = \begin{cases} x_{best}, & f(x_{best}) < f(x_{mean}) \\ x_{mean}, & \text{otherwise} \end{cases} \tag{11}$$

Through the method of opposition-based learning, the algorithm can search for the optimal value in a larger search space and guide individuals to evolve towards the optimal value, thus improving the convergence speed of the whole algorithm.

4.2. Position Update of Chicks

In CSO, the update of the chicks' position is only movement following the hens' position and does not move in the direction of the best-adapting individual in the algorithm, the rooster, which to a certain extent will easily cause the individual to fall into a local optimum and thus reduce the overall efficiency of the algorithm. Chicks in CSO have some similar characteristics with the particles in PSO. The particles obtaining the local position and the global optimum position in the PSO are equivalent to the chicks obtaining the local position next to their hens and the optimum position in the whole group guided by the rooster. Therefore, this paper uses the concept of learning factor in PSO [19] to update Equation (8):

$$x_{i,j}(t+1) = \omega \times x_{i,j}(t) + \lambda_1 \times (x_{m,j}(t) - x_{i,j}(t)) + \lambda_2 \times (x_{r,j}(t) - x_{i,j}(t)) \tag{12}$$

In this equation, m denotes the hen corresponding to the chick and r denotes the rooster corresponding to the chick. λ_1 and λ_2 are learning factors, where λ_1 denotes the degree to which the chick learns from the hen and λ_2 denotes the degree to which the chick learns from the rooster. ω denotes the weight.

4.2.1. Setting of Weight Values

The setting of the weight values is related to the population diversity in the later stage, so this paper adopts an expression of nonlinear dynamic inertia weight [17]:

$$\omega = 1 - \left(\omega_{\max} \times e^{\left(-\left(\frac{t-1}{T_{\max}}\right)\right)} + \frac{k}{2} \times \omega_{\min} \times \lg(t) - \left| \frac{T_{\max} - t}{p \times T_{\max}} \right| \right) \tag{13}$$

In this equation, $\omega_{\max}$ and $\omega_{\min}$ respectively denote the maximum and minimum values of inertial weights and $T_{\max}$ is the maximum iteration number. k and p are control factors, which adjust weights so that the chick individuals can perform global search in the group where they are located in the early stage of the algorithm, thereby improving the

accuracy of the algorithm. The local search is strengthened in the later stage to improve the overall accuracy of the solution of the overall algorithm.

4.2.2. Setting of Learning Factors

The values of factors λ_1 and λ_2, to a certain extent, can determine the degree to which chicks learn from hens and roosters. When the values of λ_1 and λ_2 are small, chicks are allowed to find better solutions in the current group. When the values of λ_1 and λ_2 are large, chicks have a larger search space so that they can search for solutions in the rest of the space. Setting the values of λ_1 and λ_2 can easily lead to the algorithm falling into a state of convergence or divergence prematurely, so it is necessary to limit the values of λ_1 and λ_2. Set $\lambda_{\min}$ and $\lambda_{\max}$ as the maximum and minimum values of the learning factors, $t_{current}$ as the current iteration number, and $t_{\max}$ and $t_{\min}$ as the maximum values, so that the chicks can learn better.

$$\lambda_1 = \begin{cases} \lambda_1 \times \dfrac{t_{\max} - t_{current}}{t_{\max}}, & \lambda_1 > \lambda_{\min} \\ \lambda_{\min}, & \lambda_1 \le \lambda_{\min} \end{cases} \tag{14}$$

$$\lambda_2 = \begin{cases} \lambda_2 \times \dfrac{t_{\max} + t_{current}}{t_{\max}}, & \lambda_2 < \lambda_{\max} \\ \lambda_{\max}, & \lambda_2 \ge \lambda_{\max} \end{cases} \tag{15}$$

4.3. Introducing Difference Algorithm to Select the Optimum Individual

In view of the selection of the individual with the best position in CSO, this paper adopts the difference algorithm. It is a heuristic random search algorithm to solve individuals in a population. Its process consists of variation, interaction, and selection.

(1) Variation: The value of the i^{th} individual in the j^{th} dimension in the g^{th} generation population is set as $x_{i,j}^{g}$ to perform variation according to Equation (16):

$$V_{i,j}^{g+1} = x_{i,j}^{g} + F \times (x_{r1,j}^{g} - x_{r2,j}^{g}) \tag{16}$$

In Equation (16), $V_{i,j}^{g+1}$ is an individual after variation. F is 0.5, and it mainly controls the scaling degree of the difference vector.

(2) Interaction: Through the selection of probability p (p is 0.5), the i^{th} individual $x_{i,j}^{g}$ in the j^{th} dimension interacts with the mutated individual $V_{i,j}^{g+1}$ to obtain a new individual.

$$\kappa_{i,j}^{g} = \begin{cases} V_{i,j}^{g+1}, & p \in [0,1] \\ x_{i,j}^{g}, & \text{otherwise} \end{cases} \tag{17}$$

(3) Selection: The "greedy" selection strategy is used in the selection of individuals. The process compares the values of two individual fitness functions f to generate new individuals with the function that has a larger value, i.e., the new individual $\kappa_{i,j}^{g}$ produced after variation and interaction is compared with the individual $x_{i,j}^{g}$ in the previous generation, and if it is smaller, $x_{i,j}^{g}$ will remain unchanged; otherwise, it will directly enter the next generation.

4.4. Boundary Processing

In CSO, it is easy for the chicken position to go out of boundaries. Although the chicks only follows the hens and stay around them, it cannot be guaranteed that when the hens move close to the boundaries, the chicks will not also cross the

boundaries in the same way. This will easily lead to the algorithm falling into a local optimum in the search process, resulting in reduced efficiency. Therefore, the method of boundary variation is adopted to shift the chicks' position out of the boundaries to a certain position in the feasible solution. The random number r is set in (0.2, 1.0). If a chick exceeds the upper boundary U of the feasible solution, the position of the chick is adjusted to Equation (18), and if the position of the chicken exceeds the lower boundary L of the feasible solution, the position is adjusted according to Equation (19).

$$x_{i,j} = U - r(U - L) \tag{18}$$

$$x_{i,j} = L + r(U - L) \tag{19}$$

4.5. Process of the Algorithm

Step 1 Correlate the individuals in ICSO with the resource scheduling schemes in cloud computing one by one;

Step 2 Initialize, set the maximum number of iterations and the initial values of CSO related parameters, set the values of ω_{max} and $\omega_{\min}$ to 0.9 and 0.1, $\lambda_{\max}$ and $\lambda_{\min}$ to 0.8 and 0.1, and r to 0.5, respectively;

Step 3 Use Equations (9)-(11) to carry out opposition-based learning on the population and optimize the initial population;

Step 4 Update the position of the chicks with Equations (13)-(15) and handle the boundaries that may be exceeded with Equations (18)-(19) to improve the quality of the individual positions in the whole algorithm;

Step 5 Select the optimal individual positions of the CSO with Equations (15)-(17);

Step 6 When the number of iterations is less than the maximum number of iterations, go to Step 4 to continue; otherwise, go to Step 7;

Step 7 The individual with the best position is the optimal individual, which corresponds to the resource scheduling scheme in cloud computing.

5. Simulation Experiment

This experiment compares the convergence of the ICSO, CSO, PSO, and GA algorithms within the task completion time. The length of task is set to [1000, 6000], the population size is set to 80, and the maximum number of iterations is 500. In the experiment, the number of virtual machines used is 100, and the number of tasks is 500. The optimization of the four algorithms is recorded in detail in each experiment, and average value is taken after repeating ten times of the experiment, with the results shown in Figure 1. As the number of iterations increases, a comparison of the convergence of the ICSO and CSO algorithms is presented in terms of task completion time. The effect of resource scheduling is compared with the Cloudsim simulation platform. The algorithm parameters involved in the experiment are shown in Table 1.

This experiment compares the convergence of ICSO and CSO in task completion time. The task length is set to [1000, 6000], the population size is set to 80, the maximum number of iterations is 500, the number of virtual machines used in the experiment is 100, and the number of tasks is 500. Each experiment records the optimization results of ICSO and CSO in detail, and the average value is obtained by repeating ten experiments. The results are shown in Figure 1. As the number of iterations of the algorithms increases, the convergence of ICSO and CSO in task completion time is compared.

Figure 1 shows that the ICSO algorithm has the shortest completion time compared to the other three algorithms, which also indicates that the ICSO algorithm has better convergence. The main reason is that the ICSO algorithm improves the performance of algorithm in terms of the swarm initialization, chick position updating, and introduction of the differential algorithm and boundary processing.

5.1. Comparison of Load Balancing

Set 10000 tasks to be distributed among the five resource points {s1, s2, s3, s4, s5}, with a processing capacity of {100, 200, 300, 400, 500}. The comparison of the effects of the four algorithms is shown in Figure 2. Due to the different processing

capacities of the cloud computing resource points, their loads are different. The load of ICSO is more balanced, which shows that the effect of cloud computing resource allocation is improved. The CSO, IPSO, and ACO loads generate different values, resulting in less tasks and poor processing capacity for resource points with stronger processing capacity.

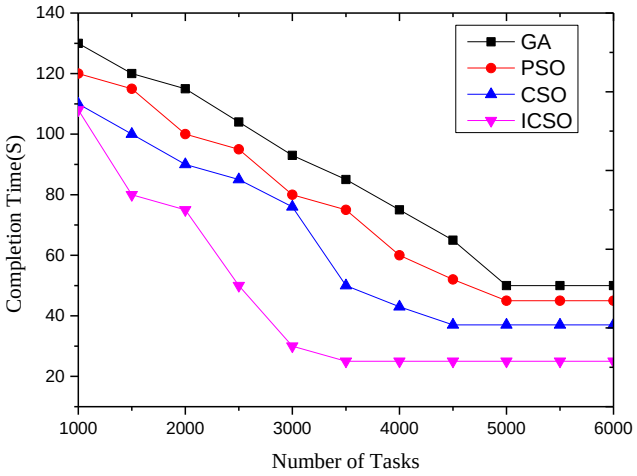

Figure 1. Comparison of the convergence of four algorithms

Table 1. Relevant parameters in algorithm comparison

Parameter	Value	Description
τ	0.0005	Pheromone
ρ_i	0.01	Pheromone evaporation rate
p	0.5	Path selection probability
w	0.5	Inertia weight
$c1$	0.5	Learning factor of the particle swarm
$c2$	0.5	Learning factor of the particle swarm
F	0.5	Following number
rand	0.5	Random number
ω_{max}	0.9	Maximum weight
ω_{min}	0.1	Minimum weight
λ_{max}	0.8	Maximum value of learning factors in ICSO
λ_{min}	0.1	Minimum value of learning factors in ICSO
r	0.5	Radom factor of boundaries
Max	100	Number of iterations

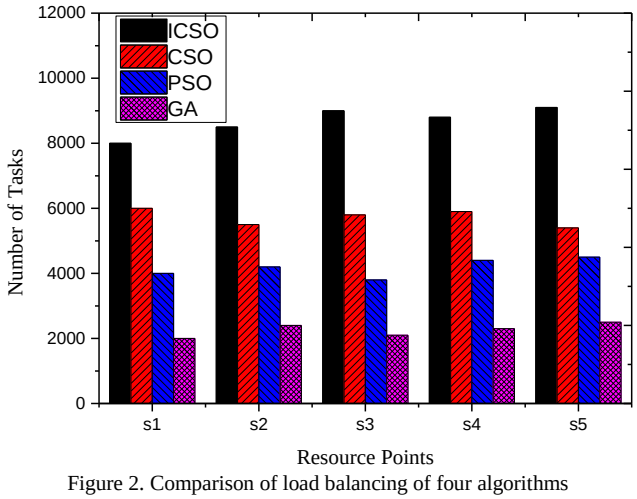

Figure 2. Comparison of load balancing of four algorithms

5.2. Comparison of Task Time

Figure 3 shows the completion time of the four algorithms with 5000 tasks. It is found from the figure that the overall time spent by ICSO is better than that spent by CSO, PSO, and GA. As the number of tasks increases gradually, the curve of the ICSO algorithm as a whole has a balanced rise, mainly because it is initialized at the beginning of the algorithm and avoids the time spent randomly selecting individual chickens at the beginning of the algorithm, thereby making the overall time of the algorithm more balanced. In particular, the introduction of the difference algorithm reduces the time for finding the optimal solution of individuals, so the overall task completion time of ICSO is satisfactory. Compared with CSO, nearly 9.13% is saved, 14.28% is saved compared with PSO, and 18.71% is saved compared with GA on average.

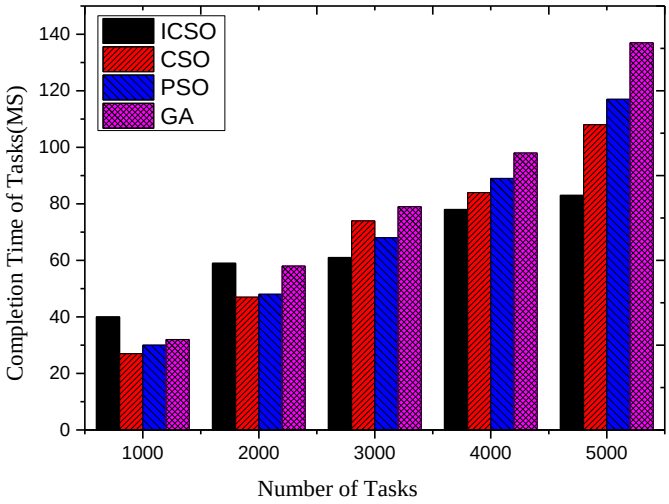

Figure 3. Comparison of task time of four algorithms

5.3. Comparison of Energy Consumption

Figure 4 shows the comparison of energy consumption of the four algorithms with 5000 tasks. At the beginning of the tasks, the energy consumption of the four algorithms is almost the same. As the number of tasks increases gradually, the energy consumption of ICSO increases gradually, mainly because the initialization of the algorithm in the early stage and the update of the individual position in the algorithm leads to a gradual increase in the energy consumption of the algorithm. However, the boundary processing in ICSO makes the energy consumption of the algorithm not easily wasted. Therefore, in the later stage, ICSO is gradually stable and the energy consumption is gradually reduced. In contrast, CSO, PSO, and GA have lower energy consumption at the beginning stage, but without improvement, the energy consumption of these algorithms increases as the number of tasks increases. Compared with CSO, nearly 8.25% is saved, 15.81% is saved compared with PSO, and 16.34% is saved compared with GA on average.

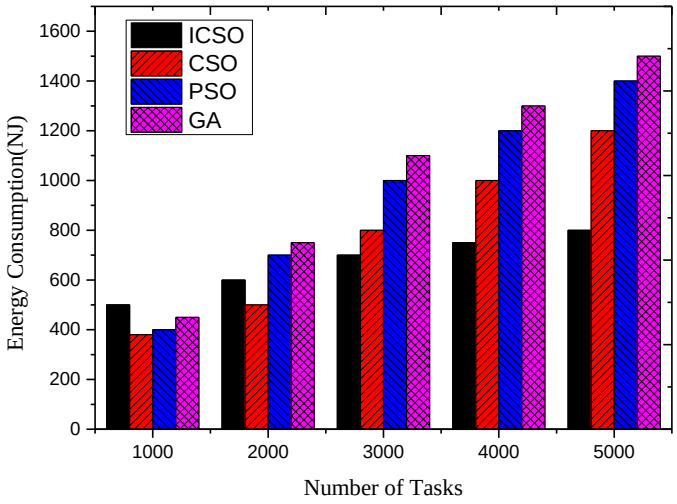

Figure 4. Comparison of energy consumption of four algorithms

5.4. Comparison of Unit Cost of Tasks

Figure 5 shows the cost comparison of the four algorithms with 5000 tasks. With the increasing number of tasks, the costs of the four algorithms all gradually increase without any significant differences. When the number of tasks reaches the halfway point, ICSO is the highest. Then, the cost consumption of ICSO gradually tends to be stable, mainly because the update of the algorithm's individual position makes the algorithm more complex, thus increasing the cost of the algorithm. In addition, the boundary processing makes the cost of the algorithm gradually increase, but the generation of the individual optimal solution is accelerated by the difference algorithm, so the cost in the later stage does not increase. The cost consumption of CSO, PSO, and GA increases constantly with the increase in the number of tasks, which shows the advantages of ICSO.

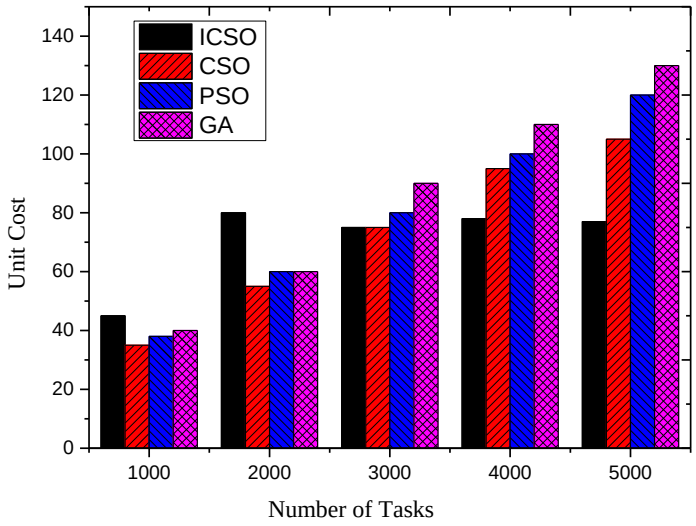

Figure 5. Comparison of unit cost of four algorithms

6. Conclusions

The effective allocation of resources has always been an important research direction of cloud computing. This paper uses the bionic algorithm CSO for resource allocation, and it has achieved some effects and improvements in CSO algorithm initialization, chick position updating, individual selection, and boundary processing. Simulation results show that the performance of the proposed algorithm has been significantly improved. At the same time, task allocation under cloud computing has achieved good results. The next step is to study the virtual machine load as a research object in order to further improve the algorithm performance.

Acknowledgements

This work is supported by the Basic Public Welfare Research Project of Zhejiang Province (No. LGF18F020014).

References

1. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, D. A. Patterson, et al., "A View of Cloud Computing," *Communications of the ACM*, Vol. 53, No. 4, pp. 50-58, 2010
2. S. Shahdi-Pashaki, E. Trymourin, and R. Tavakkolmoghaddam, "New Approach based on Group Technology for the Consolidation Problem in Cloud Computing-Mathematical Model and Genetic Algorithm," *Computational and Applied Mathematics*, Vol. 37, No. 1, pp. 693-718, 2018
3. D. Kẃo, A. Subasi, and J. Kevric, "Cloud Computing-based Parallel Genetic Algorithm for Gene Selection in Cancer Classification," *Neural Computing and Application*, Vol. 30, No. 5, pp. 1601-1610, 2018
4. M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A Survey of PSO-based Scheduling Algorithms in Cloud Computing," *Journal of Network and Systems Management*, Vol. 25, No. 1, pp. 122-158, 2017
5. N. Kumar and P. Patel, "Resource Management using ANN-PSO Techniques in Cloud Environment," in *Proceedings of the 2016 International Congresson Information and Communication Technology*, pp. 419-428, 2016
6. V. S. Kushwah and S. K. Goyal, "A Basic Simulation of ACO Algorithm under Cloud Computing for Fault Tolerant," in *Proceedings of the International Conference on Data Engineering and Communication Technology*, pp. 465-472, 2017
7. A. Ragmani, A. E. Omri, N. Abghour, K. Moussaid, and M. Rida, "A Performed Load Balancing Algorithm for Public Cloud

Computing using Ant Colony Optimization," *Recent Patents on Computer Science*, Vol. 11, No. 3, pp. 221-228, 2018
8. F. Kong and D. H. Wu, "An Improved Chicken Swarm Optimization Algorithm," *Journal of Southern Yangtze University* (*Natural Science Edition*), Vol. 14, No. 6, pp. 681-688, 2015
9. H. M. Hu, J. Y. Li, and J. G. Huang, "Economic Operation Optimization of Micro-Grid based on Chicken Swarm Optimization Algorithm," *High Voltage Apparatus*, Vol. 53, No. 1, pp. 119-125, 2017
10. S. P. Xu, D. H. Wu, and F. Kong, "Solving Flexible Job-Shop Scheduling Problem by Improved Chicken Swarm Optimization Algorithm," *Journal of System Simulation*, Vol. 29, No. 7, pp. 1497-1505, 2017
11. D. H. Wu and S. P. Xu, "Solving Multi-Objective Flexible Job Shop Scheduling Problem by the Chicken Swarm Optimization Algorithm based on Pareto Entropy," *Mini-Micro Systems*, Vol. 38, No. 12, pp. 2683-2688, 2017
12. D. Moldovan, V. R. Chifu, C. B. Pop, T. Cioara, I. Anghel, and I. Salomie, "Chicken Swarm Optimization and Deep Learning for Manufacturing Processes," in *Proceedings of 2018 17th RoEduNet Conference on Networking in Education and Research*, pp. 1-6, 2018
13. J. Grobler and A. P. Engelbrecht, "Arithmetic and Parent-Centric Headless Chicken Crossover Operators for Dynamic Particle Swarm Optimization Algorithms," *Soft Computing*, Vol. 22, No. 18, pp. 5965-5976, 2018
14. S. Torabi and F. Safi-Esfahani, "A Hybrid Algorithm based on Chicken Swarm and Improved Raven Roosting Optimization," *Soft Computing*, pp. 1-43, 2018
15. L. Cheng, I. Tachmazidis, S. Kotoulas, and G. Antoniou, "Design and Evaluation of Small-Large Outer Joins in Cloud Computing Environments," *Journal of Parallel and Distributed Computing*, Vol. 110, pp. 2-15, 2017
16. D. H. Wu, F. Kong, and Z. C. Ji, "Convergence Analysis of Chicken Swarm Optimization Algorithm," *Journal of Central South University* (*Science and Technology*), Vol. 48, No. 8, pp. 2105-2112, 2017
17. H. R. Tizhoosh, "Opposition-based Learning: A New Scheme for Machine Intelligence," in *Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, pp. 695-701, 2005
18. H. Wang, Z. J. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing Particle Swarm Optimization using Generalized Opposition-based Learning," *Information Sciences*, Vol. 181, No. 20, pp. 4699-4714, 2011
19. Y. M. Bai, "Particle Swarm Optimization and Its Application," LanZhou Jiaotong University, pp. 8-9, 2013

Liru Han is lecturer at Zhejiang University of Water Resources and Electric Power. She received her master's degree from Hebei University of Technology. Her research focuses on cloud computing.