An IFWA-BSA Based Approach for Task Scheduling in Cloud Computing

Xiaoxia Li

School of Artificial Intelligence and Big Data, Zibo Vocational Institute, Zibo, Shandong, 255000, China
E-mail: sdlixiaoxia@yeah.net

Received 28 October 2022; Accepted 15 November 2022;
Publication 14 January 2023

Abstract

Establishing an efficient cloud computing task scheduling model is the object of many scholars' research. In view of the low scheduling efficiency in cloud computing task scheduling, we propose a cloud computing task scheduling algorithm based on the fusion of the Fireworks Algorithm and Bird Swarm Algorithm (IFWA-BSA). Firstly, we describe the cloud computing task scheduling model based on time and cost constraint functions, secondly, we use chaotic backward learning and Coasean distribution for optimization in FWA initialization; we set thresholds for the radius of core fireworks and non-core fireworks for optimization; we filter the IFWA individuals after each iteration by BSA algorithm, and finally, we use the IFWA-BSA algorithm is used in cloud computing task scheduling model to solve the optimal solution. In the simulation experiments, IFWA-BSA has obvious advantages over ACO, PSO and FWA in the comparison of execution time and consumption cost indexes, which reduces the scheduling time and cost of cloud computing.

Keywords: Task scheduling model, cloud computing, chaotic inverse learning.

Journal of ICT Standardization, Vol. 11_1, 45–66.
doi: 10.13052/jicts2245-800X.1113

1 Introduction

In the 21st century, the concept of cloud computing [1] has emerged to make computer technology and network technology more widely used, and it provides more secure and reliable services for the majority of cloud users with its high flexibility, high scalability and ease of dynamic monitoring [2]. Task scheduling in cloud computing is a core part of cloud services. Whether each task gets virtual machines efficiently, whether it gets the best execution time, and whether it can save cost consumption are all issues that need to be considered for cloud computing tasks, and therefore, task scheduling is essentially a NP-complete problem [3]. A wide range of scholars have used population optimization algorithms for cloud computing task scheduling with good results [4]. For example, Genetic Algorithm (GA) [5], Particle Swarm Optimization (PSO) [6], Ant Clony Optimization (ACO) [7], Shuffled Frog Leaping Algorithm (SFLA) [8], Bat Algorithm (BA) [9], Monkey Algorithm (MA) [10], etc. The Fireworks Algorithm (FWA) is a group intelligence optimization algorithm proposed by Tan and Zhu [11] based on the form of exploding fireworks, which has high search efficiency and fast convergence speed, and the special adaptive mechanism of the algorithm itself can well balance the global and local search process, and has a good search accuracy. In order to improve the effect of cloud computing task scheduling, we propose a cloud computing task scheduling optimization method based on Improved Fireworks Algorithm-Bird Swarm Algorithm (IFWA-BSA) fusion, we use chaotic backward learning and Corsi distribution for optimization in FWA initialization; set different radii for core fireworks and non-core fireworks radii set different thresholds for optimization, and individual fireworks are filtered by the BSA algorithm after each iteration to improve the quality of the solution. The simulation experiments illustrate the extent to which our algorithm reduces the scheduling time and consumption cost of cloud computing.

The rest of this paper is organized as follows. This paper describes the progress of related work in Section 2; the time and task based optimization model for cloud computing task scheduling is introduced in Section 3; the method of IFWA-BSA and its implementation details are introduced in Section 4, and the performance of IFWA-BSA is described and validated in cloud computing task scheduling in Section 5; the full paper is summarized in Section 6.

2 Related Research

Currently, most cloud computing task scheduling focuses on reducing energy consumption and improving quality of service as the main research components.

(1) Cloud computing task scheduling based on energy consumption optimization

Literature [12] proposed the necessity and significance of energy consumption management in cloud computing task scheduling; Literature [13] proposed a new algorithm based on ant colony algorithm and order exchange migration algorithm, incorporating a pheromone precipitation strategy, which effectively reduces the number of active hosts, especially in the problem of VM migration with bottleneck resource characteristics, the algorithm has significant energy saving effect; Literature [14] proposed an energy-aware model for energy management of virtual machines, through which the energy consumption of virtual machines in invoking cloud computing tasks can be effectively reduced; Literature [15] proposes an algorithm combining a fractal mathematical prediction model and an ant colony algorithm scheduling model, which aims to minimise energy consumption and performs better in handling resource-intensive tasks and transient peak loads; Literature [16] proposes a systematic approach (EFFORT) for offload communication in the cloud. The proposed approach provides a promising solution to partially solve energy consumption issue for communication-intensive applications in a smartphone; Literature [17] illustrates the problem of balancing energy consumption and transmission latency in cloud computing systems by modestly reducing computing resources to save communication bandwidth and reduce transmission latency, and simulation Experiments show that this algorithm improves the performance of cloud computing; Literature [18] uses a dynamic voltage-frequency scaling approach to focus on virtual machine scheduling in cloud data centres, and combines the shortest job first and cyclic algorithms with Vibrant Quantum, which is proven to reduce the overall energy consumption of the system; Literature [19] proposes a task integration heuristic for energy saving, which considers the power consumption of idle computing resources;Literature [20] proposes a task integration heuristic algorithm for energy saving, which takes into account the power consumption of idle computing resources, reduces the total energy consumption of the data centre and ensures the completion quality of user tasks.

(2) Quality of service based cloud computing task scheduling
Literature [21] analyses the impact of QoS on the profit of cloud service providers, and formulates and solves a model for the profit maximisation problem;Literature [22] proposes a genetic algorithm-based association-aware optimal service scheduling method, which models and solves the association services based on QoS dependency, and finally obtains a higher quality service allocation portfolio; Literature [23] proposes an adaptive market-oriented combined double auction resource allocation for improving the service quality of tasks in cloud computing, and experimental results illustrate that the method can provide better service prices; Literature [24] proposes an algorithm that can effectively place application modules on network nodes while considering connection delay, Compared with traditional cloud computing deployment; Literature [25] proposes a grouped task scheduling algorithm with the goal of satisfying QoS by classifying tasks by user type, task type, task Literature [26] proposes a multi-QoS disk scheduling strategy (MQDS), aiming to support energy-saving and diverse QoS constraints towards reconfigurable heterogeneous cloud storage systems; Literature [27] proposes an enhanced QoS-based trustworthiness assessment model for cloud providers, where the trustworthiness is based on the cloud provider's historical reputation and the covariance-based trustworthiness. The model is based on the historical reputation of the service provider and on covariance-based mathematical techniques to assess the trustworthiness of user feedback, and experiments show that the model guarantees the quality of service; Literature [28] designs and develops a task processing framework that selects the best resources at runtime to process applications on virtual machines using improved particle swarm optimization (PSO); Literature [29] proposes the use of PSO in an Iaas platform to algorithm to optimise workflows to determine the most appropriate QoS considerations; Literature [30] proposes a scheduling algorithm-QoS-DPSO, that satisfies the time, reliability and cost of QoS, and experimental results show that QoS-DPSO can be effective in improving performance and achieving high reliability.

From the results of the above study, the use of intelligent optimisation algorithms for solving both energy and quality of service based cloud computing task scheduling has good results, but how to reduce cost and energy consumption is still the goal of continuous optimisation of cloud computing task scheduling.

3 Cloud Computing Task Scheduling Model Based on Time and Cost Constraint Functions

Traditional cloud computing task scheduling is only task scheduling efficiency as the first goal, while ignoring factors such as time and cost. For the premise of limited number of resources, we need more appropriate tasks to assign to the resources, which can effectively improve the utilization of resources. Therefore, this paper uses the execution time and consumption cost as the constraint functions for selecting tasks. These two constraint functions are described separately below. Set the set of tasks $T = \{T_1, T_2, \ldots T_n\}$ $(i \in n)$, VM_j denote j virtual machines, $T_{i\_taskLtenth}$ and $T_{i\_InputFilesize}$ denote the task execution length and other related information in task T_i, VM_{j_cpu} and VM_{j_bw} denote the computational capacity of the virtual machine to process data and the communication bandwidth required by the virtual machine, respectively.

$$resTime(I) = \frac{finishTime(I) - finishTime_{\min}}{finishTime_{\max} - finishTime_{\min}} \tag{1}$$

$$finishTime_{\max} = \frac{\sum_{i=1}^{N} T_{i\_TaskLength}}{M \times \min(VM_{j\_cpu})} + \frac{\sum_{i=1}^{N} T_{i\_InputFileSize}}{M \times \min(VM_{j\_bw})} \tag{2}$$

$$finishTime_{\min} = \frac{\sum_{i=1}^{N} T_{i\_TaskLength}}{M \times \max(VM_{j\_cpu})} + \frac{\sum_{i=1}^{N} T_{i\_InputFileSize}}{M \times \max(VM_{j\_bw})} \tag{3}$$

Equation (1) represents the time constraint function. $finishTime_{\max}$ and $finishTime_{\min}$ denote the maximum and minimum consumption time that a task may require, respectively, which is expressed as the concurrent execution time when all tasks are deployed on the worst-performing and best-performing virtual machines.

$$resCost(I) = \frac{finishCost(I) - finishCost_{\min}}{finishCost_{\max} - finishCost_{\min}} \tag{4}$$

$$finishCost_{\max} = finishTime_{\max} \times Max(VM_{j\_\cos t}) \tag{5}$$

$$finishCost_{\min} = finishTime_{\max} \times Min(VM_{j\_\cos t}) \tag{6}$$

Equation (4) represents the cost constraint function. $finishCost_{\max}$ and $finishCost_{\min}$ represent the maximum and minimum consumption costs required in the predicted task, which is expressed as the sum of the concurrent execution costs when all tasks are deployed on the worst performing and best performing VMs, and $VM_{j\_\cos t}$ represents the cost of VM consumption.

Therefore, the cloud computing task scheduling function in this paper is expressed as follows.

$$F(I) = \alpha \times resTime(I) + \beta \times resCost(I) \tag{7}$$

In Equation (7), α and β denote the weights of the time constraint function and the cost constraint function, respectively, both of which satisfy $\alpha+\beta = 1$. The optimal task scheduling solution is obtained by solving for $\min F(I)$

4 Cloud Computing Task Scheduling Based on IFWA-BSA Algorithm

4.1 Fireworks Algorithm

FWA mainly consists of three components: explosion operator, variation operator and selection policy.

(1) Explosion operator
In the initialization of the algorithm, a preliminary evaluation of the adaptation value corresponding to the location of the generated fireworks is required. The fireworks with better adaptation value (generally become core fireworks) can obtain more resources and can generate more fireworks in a small area with a strong local search capability, while the fireworks with poor adaptation value (generally become non-core fireworks) can only obtain fewer sparks, but have certain global search capability. Therefore, the blast radius and the number of sparks produced by each firework are calculated based on the fitness values of the other fireworks in the population. That is the blast radius A_i and the number of sparks S_i are calculated as follows:

$$A_i = \widehat{A} * \frac{f(x_i) - Y_{\min} + \varepsilon}{\sum_{i=1}^{N}(f(x_i) - Y_{\min}) + \varepsilon} \tag{8}$$

$$S_i = M \times \frac{y_{\max} - f(x_i) + \varepsilon}{\sum_{i=1}^{N}(y_{\max} - f(x_i)) + \varepsilon} \tag{9}$$

In Equations (8)–(9), $y_{\min} = \min(f(x_i))$ and $y_{\max} = \max(f(x_i))$ represent the minimum and maximum values of fitness in the current population, respectively. Use $\widehat{A}$ to adjust the blast radius, M is a constant to set the size of the number of sparks generated, and ε is a machine minimum parameter to avoid zero operation. In order to reduce the number of sparks produced

by fireworks with good location values, a limit on the number of sparks is needed, expressed as follows.

$$\widehat{S}_i = \begin{cases} round(a \times M), & S_i < aM \\ round(b \times M), & S_i > bM, a < b < 1 \\ round(S_i), & otherwise \end{cases} \tag{10}$$

In Equations (10), a, b is a constant, $round(\bullet)$ for the rounding function.

(2) Variation operator

In order to increase the diversity of the population, the variance operator is used to generate variance sparks. This variation spark, i.e., Gaussian variation spark, is mainly a random selection of a firework individual in the population and a Gaussian variation operation in a certain dimension to obtain the formula shown in (11).

$$\widehat{x}_{ik} = x_{ik} \times e \tag{11}$$

In Equations (11), e is the Gaussian variational operation of $N(1, 1)$.

(3) Selection strategy

In order to be able to select effective individuals in the current population to pass to the next generation, after the above 2 operations, the algorithm selects a certain number of individuals as fireworks for the next generation with the probability of selection:

$$p(x_i) = \frac{R(x_i)}{\sum_{x_j \in K} x_j} \tag{12}$$

$$R(x_i) = \sum_{x_j \in K} d(x_i - x_j) = \sum_{x_j \in K} \|x_i - x_j\| \tag{13}$$

where, $R(x_i)$ is the sum of the distances of all individuals except x_i in the current set of individual candidates. After operating in the above way, if the individual density is high, the probability of being selected around the individual will be reduced.

4.2 IFWA-BSA

(1) Population initialization based on chaos and Corsi distribution

The initial solution of FWA algorithm usually adopts a random approach, and this random processing method has some influence on the generation of

the optimal solution to a certain extent, assuming two extreme cases, one is when the generated random solution is very close to the optimal solution, the convergence of the algorithm will be fast, and the other is when the generated random solution is very far away or reversed, resulting in the algorithm needs to consume longer time to complete The other is when the generated random solution is far away or reversed, which causes the algorithm to take longer time to finish convergence. To address this problem, this paper introduces the concept of chaotic backward learning to initialize the solution, i.e., to obtain the reverse solution along with the current solution, and to select the better one by comparing the two, so as to improve the execution efficiency of the algorithm to a great extent. According to the position of the individual fireworks in the FWA algorithm, the initial value of the logistic mapping to generate the chaotic state is given as shown in Equation (14), and the inverse solution of its position is calculated as shown in Equation (15). where, a and b are the maximum and minimum values of the candidate solutions in the search space

$$x_{i,j} = x_{\min,j} + \delta_{i,j} \times (x_{\max,j} - x_{\min,j}) \tag{14}$$

$$x^*_{i,j} = x_{\min,j} + x_{\max,j} - x_{i,j} \tag{15}$$

Therefore, the reversed solutions and the original solutions are formed into a population, and the top half of the values are selected as the initialized solutions of the population in order of the highest to lowest fitness values.

In the FWA algorithm, the new generation of fireworks individuals are scattered around the parent individuals, and the generated solutions will decrease as the number of iterations increases, and the search range will gradually search to the vicinity of the parent individuals, which makes the algorithm easily fall into the local optimum and reduces the efficiency of finding the global optimum solution.

$$f(x) = \frac{1}{\pi(1 + x^2)} \tag{16}$$

(2) Set a threshold value to optimize the radius of non-core fireworks
According to the FWA algorithm, when a firework explodes, it will produce a large number of sparks in a certain area around it, and the range of sparks is the radius of the firework, the core firework is the individual firework with the smallest fitness value in the firework population, and the non-core firework radius is shown in Equation (17), it is obvious that this approach will find that

the radius of the firework with the smallest fitness value is very small, almost close to 0, so it wastes resources. To avoid this situation, a minimum radius threshold is set, which is expressed as follows:

$$A_{i,k} = \begin{cases} A_{\min,k}, & if\ A_{i,k} < A_{\min,k} \\ A_{i,k}, & otherwise \end{cases} \tag{17}$$

The threshold $A_{\min,k}$ is not set to a fixed value because of the dynamic effect of the algorithm during the iterative process, so a non-linear decreasing approach is used.

$$A_{j,k}(t) = (A_{start} + A_{end}) \times \frac{t}{1 + d_{\max}} \tag{18}$$

where, $d_{\max}$ is the maximum number of function evaluations, t is the current number of function evaluations, and A_{start} and A_{end} denote the initial and final radius values, respectively.

(3) Weighting operator to optimize the core fireworks radius

In the literature [31], it is mentioned that some adjustments need to be made to the explosion radius of the core fireworks in order to improve the performance of the algorithm, which obviously has some problems, mainly from the improvement of the local search ability, the individual fireworks do not communicate with each other, the proposed core fireworks radius calculation is too simple, only consider the scaling and directly ignore the search process, which is easy to cause the algorithm The proposed core firework radius calculation is too simple, only considering the scaling and directly ignoring the search process, which easily causes the algorithm to fall into local optimum. The proposed core fireworks radius calculation is too simple, only considering the scaling and ignoring the search process, which will easily cause the algorithm to fall into the local optimum. The algorithm converges too early, which makes it difficult for the algorithm to break out of the bound of the local optimum solution at multiple peaks. It is expressed as follows:

$$\widehat{X}_i = X_{best} + c \times (X_{best} - X_i) \tag{19}$$

$$c = \begin{cases} randc(0.1, 0.5) & if\ rand < \frac{t_1}{t_1 + t_2} \\ randc(0.5, 1.0) & otherwise \end{cases} \tag{20}$$

where X_{best} is the best firework position in the current location, c is a weighting factor controlling the distance between the first sparkle and the

current best firework position, so that the firework individuals have the ability to learn from the best individuals in the population and thus approach the historical best in the population. $randc(x, y)$ is a Cauchy distribution with position parameter x and scale parameter y. When the position parameter is small, it is beneficial for local search, using t_1 to denote the Cauchy distribution with a small position parameter, and when the position parameter is large, it is beneficial for global search, using t_2 to denote the Cauchy distribution with a large parameter. When the location parameter is small, it is beneficial to conduct local search, and when the location parameter is large, it is beneficial to conduct global search, and the distribution with a large parameter is represented by g.

(4) Optimization of individual selection using BSA

In the FWA algorithm, each iteration of the process of excellent fireworks individuals can be passed to the next generation of the population, the FWA algorithm mainly selects the individual with the smallest fitness value, while the remaining individuals are using random states, obviously, this selection strategy has certain drawbacks, although in a certain iteration, the individuals are selected according to the small fitness value, but from a global point of view it is not necessarily guaranteed that the selected individuals will be better than The remaining individuals with random states are better, while the remaining individuals adopt random states, which is not conducive to the birth of the global optimal solution of the algorithm from the overall point of view. To address this situation, this paper introduces the Bird Swarm Algorithm [32], which has the features of simple and easy implementation, few control parameters, strong robustness and strong global search capability. Each firework individual is regarded as a flock of individuals, and the optimal individual obtained by the bird flock algorithm is the current filtered individual.

Each bird foraged with its own experience and that of the whole population, and individual positions were updated as follows.

$$x_{i,j}^{t+1} = x_{i,j}^{t} + (p_{i,j} - x_{i,j}^{t}) \times C \times rand + (g_j - x_{i,j}^{t}) \times S \times rand \quad (21)$$

where $x_{i,j}^{t+1}$ and $x_{i,j}^{t}$ denote the position of the individual in the $t+1$th and tth iterations of the ith bird in the jth dimension, respectively, $rand$ denotes the random number between (0,1), C and S are the perceptual and social driving coefficients, respectively, $p_{i,j}$ denotes the best position of the ith bird in the jth dimension, and g_j denotes the best position of the whole population in the jth dimension.

The birds are trying to move to the center of the population, when the inevitable competition between each other will have an impact and hindrance. Therefore, the equation for individual position updating under vigilance behavior is as follows:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + A_1 \times (mean_j - x_{i,j}^{t}) \times rand + A_2 \times (p_{k,j} - x_{i,j}^{t}) \times rand \tag{22}$$

$$A_1 = a_1 \times \exp\left(-\frac{pFit_i}{sumFit + \varepsilon} \times N\right) \tag{23}$$

$$A_2 = a_2 \times \exp\left(\left(\frac{pFit_i - pFit_k}{|pFit_k - pFit_i| + \varepsilon}\right) \frac{N \times pFit_k}{sumFit + \varepsilon}\right) \tag{24}$$

where $mean_j$ denotes the mean position in dimension j, $p_{k,j}$ denotes the position of the kth bird in dimension j, k is a random positive integer between $[1, N]$ and $k \neq i$. a_1 and a_2 are constants between [0,2], $pFit_i$ and $pFit_k$ denote the best fit values for the ith and kth bird, respectively, $sumFit$ denotes the sum of the best fit values for the whole population, and ε is a small constant with a denominator avoiding zero. A_1 denotes the indirect effect caused by the surrounding environment as individuals move toward the center of the population, and A_2 denotes the direct effect caused by a specific disturbance as individuals move toward the center of the population. When the fitness of individual k is better than the fitness of individual i, it means that individual i suffers from greater disturbance than individual k, and therefore individual k may also move toward the population center.

During the flight, the flock will fly to other places due to predation or other disturbances from outside, and when searching for food again, some individual birds will play the role of producers to find food, while others may play the role of beggars to follow the producers. Therefore, producers and beggars are defined as follows

$$x_{i,j}^{t+1} = x_{i,j}^{t} + randn(0,1) \times x_{i,j}^{t} \tag{25}$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + (x_{k,j}^{t} - x_{i,j}^{t}) \times FL \times rand \tag{26}$$

where, $k \in [1, N]$ denotes a Gaussian distribution with mean 0 and standard deviation 1, $k \in [1, N]$, $k \neq i$, $FL \in (0, 2]$ is used to indicate that the beggar follows the producer in need of food, and the frequency of the flock flying to other places is set to FQ and is a positive integer.

4.3 Scheduling Process

Step 1: Match the cloud computing task scheduling scheme with the fireworks position in the FWA algorithm, and find the best fireworks position that is the best task scheduling scheme.

Step 2: Initialize the initialization parameters of the FWA algorithm, the parameters of the BSA algorithm, and set the number of iterations.

Step 3: Initialize the population of FWA algorithm according to Equations (14)–(16).

Step 4: Optimize the fireworks radius separately, the core fireworks radius according to Equations (17)–(18), and the non-core fireworks radius according to Equations (19)-(20).

Step 5: Process Equations (21)–(26) of the optimal individual BSA algorithm for fireworks.

Step 6: When the maximum number of iterations is reached, the algorithm ends, turn to step 7, otherwise turn to step 3.

Step 7: Get the optimal fireworks location, i.e., get the optimal cloud computing task scheduling scheme.

5 Simulation Experiments

In order to further illustrate the effect of the algorithm in the cloud computing task scheduling model, we first verify the performance of the algorithm and then check the time and cost metrics in the cloud computing task scheduling model. In this paper, we choose the hardware platform CPU as Intel Core I5, memory as 8GDDR3, hard disk capacity as 1T, software environment as Windows 10 system, and matlab 2012 simulation software.

5.1 Algorithm Performance Comparison

In order to verify the performance of the algorithms in this paper, four benchmark test functions (as shown in Table 1) were selected for algorithm validation. The comparison algorithms are ACO, PSO and FWA, and the parameters of the three algorithms are their respective original values. The mean value, minimum value, maximum value and standard deviation were chosen as the evaluation metrics, where the maximum and minimum values reflect the quality of the solution, the mean value reflects the accuracy of the

Table 1 Test function

No	Function	Testfunction
F1	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$
F2	Schwefel2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F3	Schwefel1.2	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)$
F4	Schewfel2.21	$f(x) = \max(abs(x_i))$

Table 2 Comparison results of F1 test functions in 4 dimensions

Algorithm Category	Dimension	Minimum Value	Maximum Value	Mean	Standard Deviation
ACO	2	1.2334	6.4591	9.7379	3.4958
	5	3.6781	8.9635	9.4385	8.5818
	10	62.4810	81.6937	72.630812	58.7447
	30	74.2703	93.3961	69.6857	62.7514
PSO	2	6.3461E-02	6.2145E+00	1.9559E-01	0.3861E-01
	5	1.3573E-06	4.97512E-05	2.1238E -06	1.2856E-01
	10	2.34394E+08	8.80602E+10	5.5058E+11	5.6410E+09
	30	4.3233E+12	6.1127E+13	5.4817E+12	1.6521E+12
FWA	2	1.4423E-09	2.8211E-08	1.3141E-08	4.2151E-07
	5	2.1591E-10	5.3261E-08	5.9421E-06	1.0824E-04
	10	1.1685E-07	1.0983E+04	9.5421E-05	1.6791E-05
	30	2.6491E-07	1.0412E+04	1.7021E+06	2.7132E+04
IFWA-BSA	2	0	2.6391E-10	1.1138E-13	3.9676E-15
	5	2.1681E-06	3.3945E-05	1.5915E-04	6.1594E-06
	10	4.9132E-07	4.4127E-05	1.8932E-07	6.9425E-03
	30	4.2132E-08	5.1768E-04	1.3169E-05	7.9189E-06

algorithm for a given number of iterations, and the standard deviation reflects the convergence speed of the algorithm.

The test results of the four algorithms for the four benchmark functions in dimension 2, 5, 10 and 30 are shown in Tables 2–5. From the results of these data, the algorithm in this paper outperforms the other three algorithms in the four metrics. When the dimension is 2, the minimum value of this paper's algorithm is almost 0 (except for F4), which shows the good quality of IFWA-BSA's solutions in the four benchmark functions. In other dimensions, although the IFWA-BSA algorithm does not obtain 0, the data result is

Table 3 Comparison results of F2 test functions in 4 dimensions

Algorithm Category	Dimension	Minimum Value	Maximum Value	Mean	Standard Deviation
ACO	2	0.2017	0.7435	0.6926	0.7372
	5	0.5749	0.7247	0.8601	0.5863
	10	4.7476	6.9389	4.8934	3.0541
	30	12.6634	27.8932	14.8923	15.89321
PSO	2	2.6115	3.9759	2.1301	1.2008
	5	4.4891	9.8722	3.6893	2.6915
	10	17.7692	27.0695	22.6029	19.4451
	30	34.0376	66.9916	42.51980	10.5843
FWA	2	1.9321E-10	5.9182E-07	3.9151E-05	9.2172E-03
	5	1.3262E-06	1.9183E-02	3.8324E-03	4.8021E-04
	10	2.2521E-04	1.4712E+03	3.4361E-03	3.1241E-05
	30	1.2281E-03	8.8581E+04	2.1261E+06	2.0891E+07
IFWA-BSA	2	0	4.8162E-06	5.7168E-04	1.2604E-06
	5	3.5123E-07	2.5291E-03	1.7869E-04	4.4914E-06
	10	7.0587E-07	4.8762E-03	3.6748E-04	9.3147E-05
	30	1.4331E-07	7.0266E-03	3.2329E-04	1.1260E-05

Table 4 Comparison results of F3 test functions in 4 dimensions

Algorithm Category	Dimension	Minimum Value	Maximum Value	Mean	Standard Deviation
ACO	2	0.0456	0.43155	0.9262	0.2697
	5	0.3899	0.4465	0.3814	0.9438
	10	1.6634	1.6469	2.8843	6.6702
	30	1.6928	6.0357	8.4928	5.9778
PSO	2	0.0795	0.04163	0.0935	0.0242
	5	0.6079	0.0514	0.0151	0.0372
	10	1.91657E-08	5.39703E-06	1.35589E-07	4.206236E-08
	30	8.9244E-14	5.9662E-18	4.1729E-16	6.5967E-14
FWA	2	6.8801E-15	2.1152E-13	2.1781E-10	3.7882E-08
	5	8.4612E-06	3.3651E-04	2.6613E-05	5.3987E-07
	10	4.7194E-07	1.8991E-06	1.6621E-02	3.0481E-05
	30	3.7271E-05	9.4891E-02	1.3062E-05	1.9813E-04
IFWA-BSA	2	0	8.5384E-04	3.7521E-05	1.3676E-06
	5	2.7207E-13	1.5872E-04	1.3317E-03	2.9698E-03
	10	3.8891E-08	7.2173E-03	4.9216E-05	1.4313E-03
	30	4.6707E-07	9.1665E+01	1.9022E+04	1.9563E+02

Table 5 Comparison results of F4 test functions in 4 dimensions

Algorithm Category	Dimension	Minimum Value	Maximum Value	Mean	Standard Deviation
ACO	2	0.1616	0.9114	0.9021	0.3707
	5	0.2796	0.8903	0.5869	0.6509
	10	8.8652	9.6769	9.6036	12.5890
	30	7.0823	9.8543	6.5832	2.8123
PSO	2	0.3331	0.1367	0.0378	0.1073
	5	0.5258	4.4758	3.8657	8.7475
	10	4.2168	9.5327	10.6723	3.6972
	30	15.8712	39.9261	26.7326	4.9559
FWA	2	9.1872E-07	2.949E-03	1.1893E-05	4.1220E-03
	5	2.9221E-05	7.3146E-02	2.1728E-03	1.1487E-05
	10	1.8233E-02	9.9751E-04	6.3901E-02	2.1713E-04
	30	6.5151E-04	1.6971E+02	1.3331E+04	2.2146E-02
IFWA-BSA	2	3.1982E-05	4.9127E-04	3.1693E-04	8.2232E-03
	5	1.4818E-13	3.8148E-06	1.8466E-06	6.8425E-06
	10	3.38381E-08	1.6025E-04	4.9814E-06	2.3558E-04
	30	2.1878E-11	1.3631E-03	5.19841E-06	2.2744E-02

still the smallest. The above experimental results illustrate that this shows that this paper's algorithm has a good performance in terms of algorithm performance, which lays a good foundation for the subsequent development of task scheduling in cloud computing.

5.2 Comparison of Task Scheduling Effects

In order to better illustrate the scheduling effect of the algorithm in this paper, we select three different task sets with a certain number of resources, and set the number of tasks as [100, 200] for Task1, [1000, 5000] for Task2, and [10000, 50000] for Task3, respectively.

(1) Execution time comparison

Figures 1–3 show the comparison of the execution time of the four algorithms with three different sets of tasks. Figure 1 shows that in the comparison effect of Task1 task, the time required for task scheduling under all four algorithms shows different degrees of increase as the number of tasks is increasing, but the overall difference is not significant, due to the overall small difference in the number of tasks. Figure 2 shows the comparison effect of Task2 task, because the task has been expanded nearly 100 times compared with Task1,

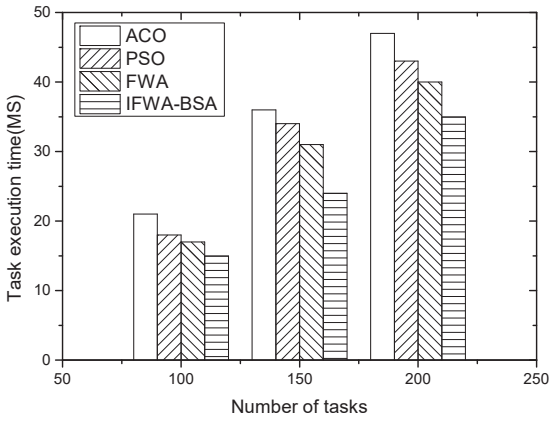

Figure 1 Comparison of the execution time in Task1 of the four algorithms.

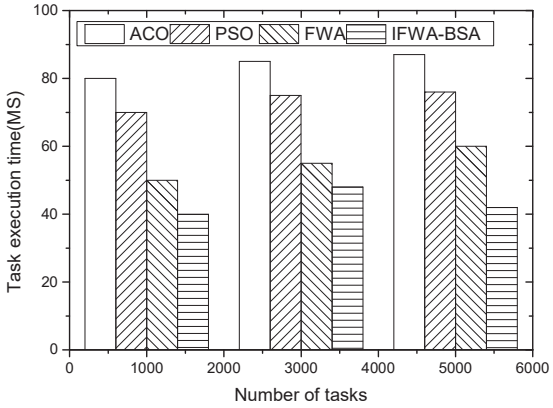

Figure 2 Comparison of the execution time in Task2 of the four algorithms.

so the consumption time between the four algorithms increases, but it is found from the figure that the consumption time of the other three algorithms under different task numbers is not large, while the execution time of the algorithm in this paper is slightly reduced when the number of tasks is larger, which may be related to the algorithm in local convergence is well. This may be related to the good improvement of local convergence of the algorithm. Figure 3 shows the comparative effect of Task3 tasks, with the increasing number of tasks, the algorithm in this paper has a more stable effect compared with the other algorithms, which shows that the algorithm in this paper can adapt to the scenario of larger number of tasks.

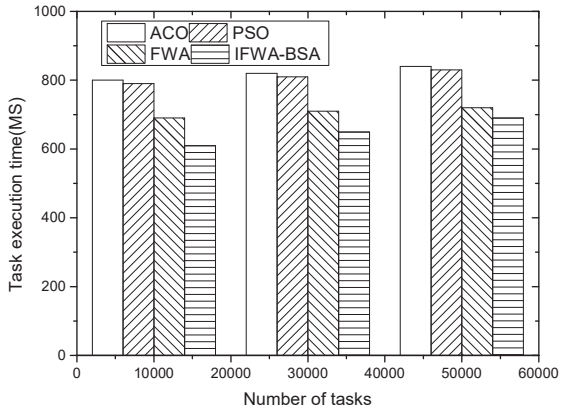

Figure 3 Comparison of the execution time in Task3 of the four algorithms.

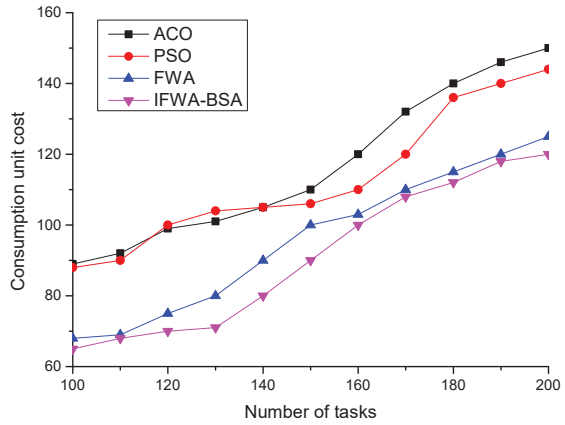

Figure 4 Comparison of consumption unit cost in Task1 of the four algorithms.

(2) Unit consumption cost

Figures 4–6 show the comparison of the unit consumption cost of the four algorithms under three different sets of tasks. The cost consumption of the four algorithms is found to increase with the increasing number of tasks in Task1 of Figure 4, and the corresponding curves of all four algorithms show substantial fluctuations, while the cost consumption of the algorithm in this paper is the lowest compared with the other three algorithms, but in general, the four algorithms are not very different from each other. The cost per unit of consumption of the four algorithms is found to increase linearly with the increasing number of tasks in Task2 in Figure 5, which shows that

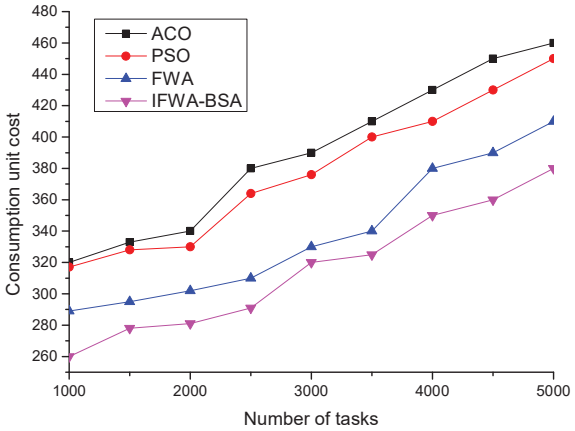

Figure 5 Comparison of consumption unit cost in Task2 of the four algorithms.

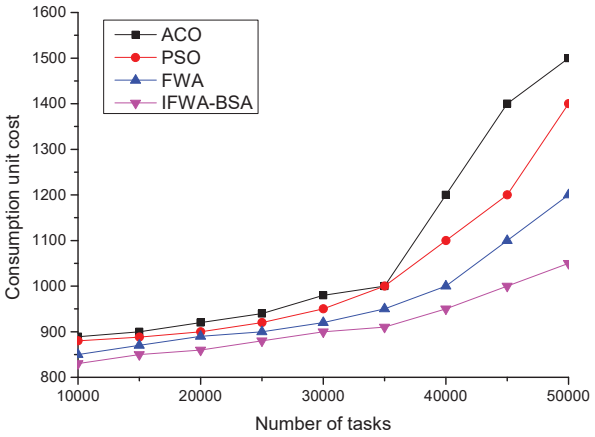

Figure 6 Comparison of consumption unit cost in Task3 of the four algorithms.

the cost of all four algorithms for cloud computing tasks increases to different degrees after the number of tasks increases, but overall, the algorithms in this paper occupy certain advantages; the Task3 task in Figure 6 finds that when the number of tasks is between 10000 and 35000, the four algorithm curves increase gently with the increase of the number of tasks, and the four curves are relatively flat, but after the number of tasks in 35000, all four algorithm curves show different degrees of increase, and the ACO algorithm increases the most, while the algorithm of this paper increases the least, which shows that the algorithm of this paper presents a better algorithm after going through

population initialization, radius setting and individual screening. This indicates that the algorithm of this paper shows better algorithm performance after population initialization, radius setting and individual screening, and therefore has better effect on unit consumption cost.

6 Conclusion

For the problem of long consumption time and high cost in task scheduling in cloud computing, we proposed IFWA-BSA algorithm for solving the problem. From the simulation results, we found that IFWA-BSA algorithm has certain advantages over ACO, PSO and FWA in terms of time and cost of task scheduling, especially under the condition of large number of tasks, and we will continue our research in the next step.

References

[1] L. Wang, G. V. Laszewski, A. Younge, X. He, M. Kunze, J. Tao and C. Fu, 'Cloud computing: a perspective study', New generation computing, Vol. 28, No. 2, pp. 137–146, Jun, 2010.

[2] S. Marston, Z. Li, S. Bandyopadhyay, J.zhang and A.ghalsasi, 'Cloud computing-The business perspective', Decision support systems, Vol. 51, No. 1, pp. 176–189. April. 2011.

[3] N. G. Hall, 'Scheduling problems with generalized due dates', IIE transactions, Vol. 18, No. 2, pp. 220–222. Feb, 1986.

[4] Z. H. Zhan, X. F. Liu, Y. J. Gong Y J, J. Zhang, H. S-H. Chung and Y. Li, 'Cloud computing resource scheduling and a survey of its evolutionary approaches', ACM Computing Surveys, Vol. 47, No. 4, pp. 1–33, Jul, 2015.

[5] J. Gu, J. Hu, T. Zhao and G. Sun, 'A new resource scheduling strategy based on genetic algorithm in cloud computing environment', Journal of computers, Vol. 7, No. 1, pp. 42–52. Jan, 2012.

[6] J. Soares, T. Sousa, H. Morais, Z. Vale, B. Canizes and A. Silva, 'Application-Specific Modified Particle Swarm Optimization for energy resource scheduling considering vehicle-to-grid', Applied Soft Computing, Vol. 13, No. 11, pp. 4264–4280, Nov, 2013.

[7] Q. Yu, L. Chen, B. Li, 'Ant colony optimization applied to web service compositions in cloud computing', Computers & Electrical Engineering, vol. 41, pp. 18–27. Jan, 2015.

[8] C. Fang, L. Wang L, 'An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem', Computers & Operations Research, Vol. 39, No. 5, pp. 890–901. May, 2012.
[9] R. Valarmathi, T. Sheela, 'Ranging and tuning based particle swarm optimization with bat algorithm for task scheduling in cloud computing', Cluster Computing, Vol. 22, No. 5, pp. 11975–11988. Dec, 2019.
[10] V. Seethalakshmi, V. Govindasamy, V. Akila, 'Hybrid gradient descent spider monkey optimization (HGDSMO) algorithm for efficient resource scheduling for big data processing in heterogenous environment', Journal of Big Data, Vol. 7, No. 1, pp. 1–25. Jul, 2020.
[11] Y. Tan, Y. Zhu, 'Fireworks algorithm for optimization', In International conference in swarm intelligence. Springer, Berlin, Heidelberg, 2010: 355–364.
[12] N. Chaurasia, M. Kumar, R. Chaudhry and O. P. Verma, 'Comprehensive survey on energy-aware server consolidation techniques in cloud computing', The Journal of Supercomputing, Vol. 77, No. 10, pp. 11682–11737, Mar, 2021.
[13] Y. Gao, H. Guan, Z. Qi, Y. Hou and L. Liu, 'A multi-objective ant colony system algorithm for virtual machine placement in cloud computing', Journal of computer and system sciences, 2013, Vol. 79, No. 8, pp. 1230–1242, Dec, 2013.
[14] R. Zolfaghari, A. Sahafi, A. M. Rahmani, and R. Rezaei, 'An energy-aware virtual machines consolidation method for cloud computing: Simulation and verification', Software: Practice and Experience, Vol. 52, No. 1, pp. 194–235. Jun, 2022.
[15] H. Duan, C. Chen, G. Min and Y. Wu, 'Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems', Future Generation Computer Systems, Vol. 74, pp. 142–150. Sep, 2017.
[16] S. U. R. Malik, H. Akram, S. S. Gill, H. Pervaiz and H. Malik, 'EFFORT: Energy efficient framework for offload communication in mobile cloud computing', Software: Practice and Experience, Vol. 51, No. 9, pp. 1896–1909. Apr, 2021.
[17] R. Deng, R. Lu, C. Lai, T. H. Luan and H. Liang, 'Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption', IEEE Internet of Things Journal, Vol. 3, No. 6, pp. 1171–1181. Dec, 2016.
[18] J. K. Jeevitha, G. Athisha, 'A novel scheduling approach to improve the energy efficiency in cloud computing data centers', Journal of Ambient

Intelligence and Humanized Computing, Vol. 12, No. 6, pp. 6639–6649. Jul, 2021.

[19] R. C. Lee, A. Y. Zomaya, 'Energy efficient utilization of resources in cloud computing systems', The Journal of Supercomputing, Vol. 60, No. 2, pp. 268–280. Mar, 2012.

[20] X. Chen, L Cheng, C Liu, Q. Liu, J. Liu, Y. Mao and J. Murphy, 'A WOA-based optimization approach for task scheduling in cloud computing systems', IEEE Systems Journal, Vol. 14, No. 3, pp. 3117–3128. Sep, 2020.

[21] J. Mei, K. Li, K. Li, 'Customer-Satisfaction-Aware Optimal Multiserver Configuration for Profit Maximization in Cloud computing', IEEE Transactions on Sustainable Computing, Vol. 2, No. 1, pp. 17–29. Mar, 2017.

[22] H. Jin, X. Yao, Y. Chen, 'Correlation-aware QoS modeling and manufacturing cloud service composition', Journal of Intelligent Manufacturing, Vol. 28, No. 8, pp. 1947–1960. Apr, 2017.

[23] A. Umer, B. Nazir, Z. Ahmad, 'Adaptive market-oriented combinatorial double auction resource allocation model in cloud computing', The Journal of Supercomputing, Vol. 78, No. 1, pp. 1244–1286. Jun, 2022.

[24] S. R. Hassan, I. Ahmad, J. Nebhen and A. U. Rehman, 'Design of Latency-Aware IoT Modules in Heterogeneous Fog-Cloud Computing Networks', Cmc-Computers Materials & Continua, Vol. 70, No. 3, pp. 6057–6072. Oct, 2022.

[25] H. G. E. D. H. Ali, I. A. Saroit, A. M. Kotb, 'Grouped tasks scheduling algorithm based on QoS in cloud computing network', Egyptian Informatics Journal, Vol. 18, No. 1, pp. 11–19. Mar, 2017.

[26] X. You, D. Sun, X. Lv, S. Gao and R. Buyya, 'MQDS: An energy saving scheduling strategy with diverse QoS constraints towards reconfigurable cloud storage systems', Future Generation Computer Systems, Vol. 129, pp. 252–268. Apr, 2022.

[27] H. Hassan, A. I. El-Desouky, A. Ibrahim, E. M. EI-Kenawy and A. Ibrahim, 'Enhanced QoS-based model for trust assessment in cloud computing environment', IEEE Access, Vol. 8, pp. 43752–43763. Mar, 2020.

[28] M. Kumar, S. C. Sharma, 'PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing', Neural Computing and Applications, Vol. 32, No. 16, pp. 12103–12126. Jun, 2020.

[29] M. Farid, R. Latip, M. Hussin and N. A. W. A. Hamid, 'A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing', Symmetry, Vol. 12, No. 4, pp. 551–577, Apr, 2020.
[30] W. Jing, C. Zhao, Q. Miao, H. Song and Guang. Chen, 'QoS-DPSO: QoS-aware Task Scheduling for Cloud Computing System', Journal of Network and Systems Management, Vol. 29, No. 1, pp. 1–29. Oct, 2021.
[31] S. Zheng, A. Janecek, J. Li and Y. Tan, 'Dynamic search in fireworks algorithm', In 2014 IEEE Congress on evolutionary computation (CEC). IEEE, 2014: 3222–3229.
[32] X. B. Meng, X. Z. Gao, L. Lu, Y. Liu and H.zhang, 'A new bio-inspired optimisation algorithm: Bird Swarm Algorithm', Journal of Experimental & Theoretical Artificial Intelligence, Vol. 28, No. 4, pp. 673–687. Jun, 2016.

Biography

Xiaoxia Li received her bachelor's degree in computer science and technology from Liaocheng University in 2003 and her master's degree in computer application and management from Qingdao University in 2013. She is currently a lecturer in the Artificial Intelligence and Big Data College of Zibo Vocational Institute, and her research interests are computer networks, computer applications, big data and cloud computing.