Postprint of Research on IFWA-ABC-Based Cloud Computing Resource Task Scheduling Algorithm

Authors: Chen Xuan, Wang Dawei, Wang Changliang, Long Dan

Date: 2018-06-19T00:00:00+00:00

Abstract

To address the low efficiency and uneven resource allocation in cloud computing resource task scheduling, this work fuses the improved fireworks algorithm (IFWA) and artificial bee colony (ABC) algorithm into IFWA-ABC. First, the cloud computing resource task scheduling problem is described. Second, chaotic opposition-based learning and Cauchy distribution are employed to optimize the FWA initialization, the radii of core and non-core fireworks are optimized separately, and the optimal individual in FWA is obtained through the improved ABC algorithm. Finally, the IFWA-ABC algorithm is applied to cloud computing task scheduling. Simulation experiments comparing IFWA-ABC with FWA and ABC on metrics including virtual machines, execution time, cost consumption, and energy consumption demonstrate that IFWA-ABC possesses significant advantages and can effectively improve the efficiency of cloud computing resource allocation.

Full Text

Preamble

Cloud Computing Task Scheduling Algorithm Based on IFWA-ABC

Chen Xuan[1], Wang Dawei[1], Wang Changliang[1], Long Dan[2]
(1. Zhejiang Industry Polytechnic College, Shaoxing Zhejiang 312000, China; 2. Zhejiang University, Hangzhou 310058, China)

Abstract: To address the low efficiency of cloud computing resource task scheduling and uneven resource allocation, this paper combines the improved fireworks algorithm (IFWA) and artificial bee colony (ABC) algorithm into IFWA-ABC. First, the paper describes cloud computing resource task scheduling. Second, it employs chaotic reverse learning and Cauchy distribution to

optimize FWA initialization, optimizes the radii of core and non-core fireworks separately, and obtains the optimal individuals in FWA through the improved ABC algorithm. Finally, the IFWA-ABC algorithm is applied to cloud computing task scheduling. Simulation experiments demonstrate that IFWA-ABC exhibits clear advantages over FWA and ABC in terms of virtual machine load balancing, execution time, cost consumption, and energy consumption metrics, effectively improving cloud computing resource allocation efficiency.

Keywords: fireworks algorithm; artificial bee colony algorithm; cloud computing; chaotic reverse learning

0 Introduction

Task resource scheduling in cloud computing is a crucial component for measuring cloud computing performance, and the key to improving cloud computing efficiency lies in how to effectively and reasonably allocate tasks to resources [1]. Introducing intelligent algorithms into cloud computing resource scheduling has become mainstream in current research. On one hand, scholars employ single intelligent algorithms for cloud computing resource scheduling, such as PSO algorithms [2,3], genetic algorithms [4,5], ant colony algorithms [6,7], artificial bee colony algorithms [8], DAG scheduling algorithms [9], and fireworks algorithms [10]. On the other hand, hybrid intelligent algorithms are also applied to cloud computing resource scheduling. Reference [11] proposes a fusion of particle swarm optimization and ant colony algorithms for cloud computing resource scheduling, which improves algorithmic precision and scheduling efficiency but lacks comparison with more intelligent algorithms. Reference [12] employs genetic operators to generate initial pheromone distribution and uses bidirectional convergence ant colony operators to obtain precise solutions. While this approach improves solution accuracy and convergence speed, the use of genetic operators for initial pheromone generation increases algorithmic complexity and solution time. Reference [13] proposes improvements to both ant colony and particle swarm algorithms, leveraging their respective advantages to establish an ant colony-particle swarm algorithm for cloud computing resource scheduling efficiency. This algorithm reduces time consumption but decreases algorithmic precision.

Building upon the aforementioned research and addressing the limitations of FWA, this paper improves population initialization and fireworks radius, and adopts ABC to enhance the selection strategy. The resulting IFWA-ABC algorithm is then applied to cloud computing resource scheduling. Simulation experiments demonstrate that IFWA-ABC exhibits superiority over FWA and ABC algorithms in load balancing, execution time, and cost consumption.

1 Cloud Computing Resource Scheduling Task Description

Cloud computing task scheduling efficiency is a critical metric for evaluating cloud computing resource scheduling performance. This paper describes cloud computing resource task scheduling from three perspectives: virtual machine load, execution time, and cost consumption, using these as evaluation criteria for cloud computing resource scheduling effectiveness.

1.1 Virtual Machine Load

Virtual machine load is an essential component of cloud computing task resource scheduling. Since cloud computing resources change dynamically as tasks progress, recording parameters of individual virtual machines cannot reflect the overall dynamic situation of cloud computing. This paper adopts the following method to record virtual machines:

$$Re = [jid, VM_j, lastTime_j]$$

where: jid represents the unique identifier of the virtual machine; VM_j represents the performance parameters of the virtual machine itself; $lastTime_j$ represents the execution time of the last task in the virtual machine. Obviously, when each task iT is assigned to virtual machine VM_j, the final task execution time $lastTime'_j$ needs to be modified:

$$lastTime'_j = lastTime_j + taskTime_{ij}$$

Therefore, the function measuring the effectiveness of resource scheduling for task iT is as follows:

$$I = wt \times LoadVM + wc \times Time + we \times Cost + we \times Energy$$

where wt, wc, and we are influence factors for virtual machine, time, cost, and energy consumption respectively, all being random numbers between 0 and 1, with $wt + wc + we = 1$. Obviously, the influence weights of virtual machine load, execution time, cost consumption, and energy consumption in the task resource selection process can be set through the values of wt, wc, and we, thereby satisfying both system constraints on task execution time and user requirements for cost reduction.

1.2 Execution Time

Time consumption in cloud computing is an important component for measuring cloud computing resource scheduling effectiveness. This paper uses $vmTime$ to represent the execution of all tasks, meaning the actual completion time of each virtual machine is the sum of the time of all tasks assigned to that virtual

machine. $vmTime_j$ represents the predicted execution time of tasks assigned to the j-th virtual machine. Therefore, the execution time of a task iT is:

$$vmTime_j = \sum taskTime_{ij}$$

1.3 Cost Constraints

In addition to time, another important factor in cloud computing resource task scheduling is task cost consumption, which constitutes a vital component of cloud computing resource task scheduling. The total cost of each task is as follows:

$$Cost_{ij} = \frac{taskLenth_i}{VMcpu_j} + \frac{InputFilesize_i}{VMbw_j}$$

where $taskLenth_i$ and $InputFilesize_i$ represent the task length and related input file size of task iT respectively, while $VMcpu_j$ and $VMbw_j$ represent the computing capability and communication capability of the virtual machine respectively.

1.4 Energy Consumption

Cloud computing task energy consumption relates to the effectiveness of cloud computing resource scheduling. This paper uses $vmEnger$ to represent the energy consumption of all tasks, meaning the actual energy consumption of each virtual machine is the sum of the energy consumption of all tasks assigned to that virtual machine. $vmEnger_j$ represents the energy consumption of tasks assigned to the j-th virtual machine, and the energy consumption of a single task is:

$$vmEnger_j = \sum taskEnger_{ij}$$

---

2 FWA Algorithm

Tan and Zhu proposed the fireworks algorithm (FWA) in 2010 based on the morphology of fireworks explosions [14]. FWA exhibits characteristics of randomness, locality, and diversity, and is primarily used for optimization problems. Each firework in FWA serves as a feasible solution to the optimization problem. The number of sparks generated by each firework is calculated based on the fitness function—fireworks with better fitness values produce more sparks, while those with worse fitness values produce fewer. The mutation process during fireworks explosion ensures population diversity. FWA mainly consists of three components: the explosion operator, mutation operator, and selection strategy.

2.1 Explosion Operator

In algorithm initialization, the fitness values corresponding to the positions of generated fireworks need preliminary evaluation. Fireworks with better fitness values (generally called core fireworks) can obtain more resources and generate more fireworks within a small range, demonstrating strong local search capability. Fireworks with poorer fitness values (generally called non-core fireworks) can only obtain fewer sparks but possess certain global search capabilities. Therefore, the explosion radius and number of sparks for each firework are calculated based on the fitness values of other fireworks in the population. The explosion radius A_i and number of explosion sparks S_i are calculated as follows:

$$A_i = \hat{A} \times \frac{f(x_i) - Y_{min} + \varepsilon}{\sum(f(x_i) - Y_{min}) + \varepsilon}$$
$$S_i = M \times \frac{Y_{max} - f(x_i) + \varepsilon}{\sum(Y_{max} - f(x_i)) + \varepsilon}$$

where Y_{min} and Y_{max} represent the minimum and maximum fitness values in the current population respectively. M is used to adjust the explosion radius, $\hat{A}$ is a constant used to set the size of spark generation, and ε is a machine minimum parameter used to avoid zero operations. To reduce the number of explosion sparks generated by fireworks with good position values, the spark count needs to be limited:

$$S_i = \begin{cases} round(a \cdot M) & S_i < a \cdot M \\ round(b \cdot M) & S_i > b \cdot M \\ round(S_i) & \text{otherwise} \end{cases}$$

where a and b are constants, and $round(\cdot)$ is a rounding function.

2.2 Mutation Operator

To increase population diversity, a mutation operator is used to generate mutation sparks. These mutation sparks are Gaussian mutation sparks, primarily obtained by randomly selecting a firework individual from the population and performing Gaussian mutation operations on certain dimensions:

$$x_{ik} = x_{ik} \times e$$

where e represents the Gaussian mutation operation on x_{ik}.

2.3 Selection Strategy

To enable effective individuals from the current population to be passed to the next generation, the algorithm selects a certain number of individuals as the next generation's fireworks after the above two operations. The selection probability is:

$$P(x_i) = \frac{R(x_i)}{\sum R(x_j)}$$

where $R(x_i)$ is the sum of distances between the current individual candidate set and all other individuals except x_i. Through this operation, if an individual has high density, the probability of being selected around that individual decreases.

---

3 IFWA-ABC-Based Cloud Computing Resource Scheduling Algorithm

To address the problems of premature convergence, low solution precision, and susceptibility to local optima in the FWA algorithm, this section improves the FWA algorithm in terms of initialization, fireworks radius, and selection strategy, and applies the resulting IFWA-ABC algorithm to cloud computing resource scheduling.

3.1 Improved FWA Algorithm

3.1.1 Chaotic Reverse Learning and Cauchy Distribution for Initial Position Optimization The initial solutions of the FWA algorithm are typically generated randomly, which to some extent affects the generation of optimal solutions. Consider two extreme cases: when the randomly generated solution is close to the optimal solution, the algorithm converges quickly; when the random solution is far from or opposite to the optimal solution, the algorithm requires longer time to converge. To address this issue, this paper introduces the concept of chaotic reverse learning for initialization, which simultaneously obtains the reverse solution of the current solution and selects the better one through comparison. This can significantly improve algorithm execution efficiency. According to the FWA algorithm, the position of firework individuals is given, and Logistic mapping is used to generate initial values in a chaotic state. The reverse solution of its position is calculated.

The reverse solution and the original solution form a group, which is sorted in descending order according to fitness values. The top half of the values are selected as the initial solutions for the population. In the standard FWA algorithm, new firework individuals are scattered around parent individuals, and the generated solutions continuously decrease with increasing iteration numbers, gradually narrowing the search range to the vicinity of parent individuals. This

makes the algorithm prone to falling into local optima and reduces the efficiency of finding global optimal solutions. Using random numbers generated by Cauchy distribution can make the search region more extensive, thereby improving the algorithm's global search capability. Therefore, the following density function is used for calculation:

$$f(x) = \frac{1}{\pi} \frac{\gamma}{\gamma^2 + (x - x_0)^2}$$

$$\gamma = \gamma_{final} + (\gamma_{initial} - \gamma_{final}) \times (1 - \frac{iter}{iter_{max}})$$

where $iter$ represents the iteration number, $iter_{max}$ represents the maximum iteration number, $\gamma_{initial}$ and γ_{final} represent the initial and final values of the scale parameters in the Cauchy distribution respectively.

3.1.2 Setting Threshold to Optimize Non-core Fireworks Radius According to the FWA algorithm, when fireworks explode, a large number of sparks are generated in a certain area around them. The range of sparks is the fireworks radius. Core fireworks are individuals with the minimum fitness value in the fireworks population, while non-core fireworks have relatively larger fitness values. Obviously, this approach results in a very small radius for fireworks with minimum fitness values, almost approaching zero, thereby wasting resources. To avoid this situation, a minimum radius threshold is set, expressed as follows:

$$A_{ik} = \begin{cases} A_{min,k} & A_{ik} < A_{min,k} \\ A_{ik} & \text{otherwise} \end{cases}$$

where $A_{min,k}$ is not a fixed value because the algorithm produces dynamic effects during iteration. Therefore, a nonlinear decreasing method is adopted:

$$A_{min}(t) = A_{end} + (A_{start} - A_{end}) \times e^{-(\frac{t}{maxdt})^2}$$

where $maxdt$ represents the maximum number of function evaluations, A_{start} and A_{end} represent the initial and final radius values respectively.

3.1.4 ABC Algorithm-Based Selection Strategy In the FWA algorithm, excellent firework individuals are passed to the next generation population in each iteration. The FWA algorithm primarily selects individuals with minimum fitness values, while the remaining individuals are selected randomly. Obviously, this selection strategy has certain drawbacks. Although individuals with small fitness values are selected in a given iteration, this does not guarantee that the selected individuals are necessarily better than the remaining randomly selected individuals from a global perspective. Meanwhile, the random selection of remaining individuals is detrimental to the generation of global optimal solutions.

To address this issue, this paper introduces the Artificial Bee Colony (ABC) algorithm, which features simple implementation, few control parameters, strong robustness, and strong global search capability. Through the optimization role of each artificial bee individual, the optimal individual is ultimately found. In the ABC algorithm, there are three types of individuals: employed bees, onlooker bees, and scout bees. Each employed bee corresponds to a definite food source (the position of an individual in FWA) and updates the position around the food source during iteration. Based on the fitness value of the food source, onlooker bees are employed using a roulette wheel method for foraging to obtain new food sources. When no new food source appears after a certain number of searches, the food source is abandoned, and the employed bee becomes a scout bee to search for new food sources randomly.

a) Initialization. All individuals x_i in the FWA population are counted, and the fitness value of each individual is calculated to generate feasible solutions v_{ij}, where x_i is a D-dimensional vector.

b) Optimal food source. Searching in the relevant domain of the current food source and drawing on the idea of the DE algorithm, the formula for the new food source is:

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{rj}) + \phi_{ij}(x_{best} - x_{ij})$$

where x_{r1}, x_{r2}, x_{r3}, and x_{r4} are randomly selected targets, φ_{ij} and ϕ_{ij} are random numbers in $[-1, 1]$ and $[0.5, 1]$ respectively, and x_{best} is the current global optimal solution. Obviously, drawing on the influence of the differential algorithm, this can effectively maintain population diversity, while improving convergence speed and enhancing algorithm exploitation capability.

c) Employed bee probability. The probability of onlooker bees selecting employed bees is:

$$P_i = \frac{fit(x_i)}{\sum_{n=1}^{SN} fit(x_n)}$$

where $fit(x_i)$ represents the fitness value of the i-th solution, and P_i represents the richness of the food source. The richer the food source, the greater the probability of being selected by onlooker bees.

In the ABC algorithm, when the food source corresponding to an individual's position in FWA is not improved during iteration, the food source is abandoned (i.e., the firework individual), and the position of the food source is placed in the tabu list. Simultaneously, the employed bee corresponding to the food source is converted into a scout bee, and a new FWA individual position is randomly generated to replace the original individual.

Based on the advantages of the fused algorithms, the steps for cloud computing resource task scheduling are as follows:

a) Map cloud computing task scheduling solutions one-to-one with firework positions in FWA, where the optimal firework position corresponds to the optimal task scheduling solution.

b) Initialize parameters for the FWA algorithm and ABC algorithm, and set the iteration count.

c) Initialize the FWA algorithm population.

d) Optimize firework radii separately: optimize core firework radii and optimize non-core firework radii.

e) Process firework individuals using formulas from the ABC algorithm.

f) If the maximum iteration count is reached, the algorithm ends; otherwise, return to step c).

g) Obtain the optimal firework position, i.e., the optimal cloud computing task scheduling solution.

---

4 Experiments

4.1 Algorithm Performance

To demonstrate the performance of the proposed algorithm, the IFWA-ABC algorithm is compared with the FWA algorithm and ABC algorithm on three classic test functions. Each algorithm runs 200 times, with comparison results shown in Table 1.

Table 1 Comparison of Test Results

Function	FWA	ABC	IFWA-ABC
f_1	3.78214e-2	5.38272e-2	4.12712e-2
f_2	7.36234e-2	5.89234e-2	9.39802e-2
f_3	-	-	-

From Table 1, it can be observed that IFWA-ABC demonstrates significant performance improvement compared to the FWA algorithm, indicating the effectiveness of the proposed algorithm.

4.2 Cloud Computing Task Scheduling

To illustrate the effectiveness of the IFWA-ABC algorithm in cloud computing resource scheduling, it is compared with FWA and ABC algorithms in

terms of virtual machine load balancing, time consumption, and cost consumption. The hardware environment uses a Core i3 CPU, 4GB DDR memory, and 1000GB hard disk capacity, with Linux operating system as the software environment and Cloudsim for simulating the cloud computing environment. Three different task sets are selected: Task1[100,1000], Task2[1000,10000], and Task3[10000,100000], with 500 resources. The relevant parameters for FWA and ABC algorithms follow references [8] and [10], while IFWA-ABC algorithm parameters are set as follows: $initial\gamma$ and $final\gamma$ are 10 and 50 respectively, $maxd$ is 100, A_{start} and A_{end} are 2.5 and 10 respectively, and φ_{ij} and ϕ_{ij} are 0.5 and 1 respectively, with p being 0.5.

4.3 Load Balancing Comparison

This paper tests the three task sets across 10 resource points $\{s1, s2, s3, s4, s5, s6, s7, s8, s9, s10\}$, where s1 to s10 process 50, 100, 150, 200, 250, 300, 350, 400, 450, and 500 tasks respectively. Since each cloud computing resource processing point has different capabilities, the loads vary. When the task quantity is small, the load balancing of the three algorithms is similar. As the task quantity gradually increases, IFWA-ABC demonstrates stable load balancing, while FWA and ABC produce different load values, indicating that IFWA-ABC is effective in cloud computing resource allocation with stable load distribution.

4.4 Execution Time Comparison

The results indicate that the execution time of all three algorithms increases with task quantity. When the task quantity is small, the execution times are similar. However, when the task quantity becomes large, the differences in execution time among the three algorithms become significant, with IFWA-ABC clearly outperforming FWA and ABC algorithms. This demonstrates that IFWA-ABC can adapt to task resource scheduling in cloud computing environments.

4.5 Cost Consumption Comparison

The results show that the cost consumption of all three algorithms increases with task quantity. When the task quantity is small, the cost consumption curves of the three algorithms grow smoothly. As the task quantity gradually increases, the curves rise to varying degrees, with IFWA-ABC showing a relatively gentle growth trend compared to the steeper increases of FWA and ABC algorithms. This indicates that IFWA-ABC has significant advantages in cost consumption and can adapt well to cloud computing requirements.

4.6 Energy Consumption Comparison

When the task quantity is small, the differences in energy consumption among the algorithms are not substantial, though IFWA-ABC maintains a certain advantage. As the task quantity gradually increases, IFWA-ABC demonstrates

clear advantages over FWA and ABC algorithms, proving that IFWA-ABC can effectively reduce energy consumption.

5 Conclusion

Addressing the challenges of cloud computing task resource scheduling, this paper applies the FWA algorithm to task scheduling and improves the FWA algorithm itself by fusing it with the ABC algorithm to form IFWA-ABC. Simulation experiments demonstrate that IFWA-ABC achieves good results in load balancing, execution time, cost consumption, and energy consumption, proving its capability to adapt to task scheduling in cloud computing environments.

References

[1] Zhang Jianxun, Gu Zhimin, Zheng Chao. Survey of research progress on cloud computing [J]. Application Research of Computers, 2010, 27 (2): 429-433.

[2] Masdari M, Salehi F, Jalal M, et al. A survey of pso-based scheduling algorithms in cloud computing [J]. Journal of Network and Systems Management, 2017, Vol. 25, No. 1, pp: 122-158.

[3] Kumar N, Patel P. Resource management using ANN-PSO techniques in cloud environment [C]// Proc of International Congress on Information and Communication Technology. 2016: 419-428.

[4] Shahdi-Pashaki S, Teymourian E, Tavakkkoli Moghaddam R. New approach based on group technology for the consolidation problem in cloud computing-mathematical model and genetic algorithm [J]. Computational and Applied Mathematics, 2016, DOI: 10.1007/s40314-016-0362-4.

[5] Aziza H, Krichen S. Bi-objective decision support system for task-scheduling based on genetic algorithm in cloud computing [J]. Computing, 2018, 100 (2): 65-91.

[6] Ragmani A, Omri A E, Abghour N, et al. A performed load balancing algorithm for public Cloud computing using ant colony optimization [C]// Proc of the 2nd International Conference on Cloud Computing Technologies and Application. 2016: 7847703.

[7] Satapathy. A basic simulation of ACO algorithm under cloud computing for fault tolerant [C]// Proc of International Conference on Data Engineering and Communication Technology. 2017: 465-472.

[8] Yang Haijun. A job scheduling algorithm based on artificial bee colony in

cloud computing [J]. Mathematics in Practice and Theory, 2012, 42 (10): 115-120.

[9] Xu Jianrui, Zhu Huijuan. Multi-objective scheduling algorithm of DAG tasks in cloud computing [J/OL]. Application Research of Computer, 2019, 36 (1). [2018-01-10]. http://www.arocmag.com/article/02-2019-01-023.html.

[10] Huang Jinwei, Guo Fang. Multi-objective task scheduling based on fireworks algorithm in cloud computing [J]. Application Research of Computers, 2017, 34 (6): 1718-1720.

[11] Chen X. Task Scheduling of cloud computing using integrated particle swarm algorithm and ant colony algorithm [J]. Cluster Computing, DOI: 10.1007/s10586-017-1479-y, 2017.

[12] Zhao Junpu, Yin Jinyong, Jin Tongbiao, et al. Application of genetic ant colony algorithm in cloud computing resource scheduling [J]. Computer Engineering and Design, 2017 (3): 693-697.

[13] Sa Rina. Cloud computing resource scheduling scheme based on ant colony particle swarm optimization algorithm [J]. Journal of Jilin University: Sci Ed, 2017, 55 (6): 1518-1522.

[14] Tan Y, Zhu Y. Fireworks algorithm for optimization [C]. Berlin: Springer, 2010: 355-364.

[15] Zheng S, Janecek A, Li J, et al. Dynamic search in fireworks algorithm [C]// Evolutionary Computation. 2014: 3222-3229.

Note: Figure translations are in progress. See original paper for figures.