Eng Int Syst (2021) 1: 27–32

Engineering
Intelligent Systems

Energy Saving Task Scheduling Based on Optimized Ant Colony Algorithm in Cloud Environment

Haiqin Liu[1,*] **and Haifeng Yi**[2,†]

[1] *Department of Mathematics and Information Engineering, Dongchang College of Liaocheng University, Liaocheng 252000, China*
[2] *Business School, Pingxiang University, Pingxiang 337055, China*

In a cloud environment, the majority of computational power requirements are concentrated in the cloud, resulting in higher energy consumption for data centers. A method of reducing energy consumption while also reducing the time span of task scheduling has become an urgent problem to be solved. In this paper, an optimal ant colony scheduling algorithm combined with a genetic algorithm is proposed, and an energy consumption factor is introduced into the algorithm. Experiments show that this algorithm can effectively improve the time efficiency of task scheduling and reduce energy consumption.

Keywords: Cloud computing; Task scheduling; Time span; Energy consumption; Genetic algorithm

1. INTRODUCTION

Cloud computing first decomposes a huge task into several sub-tasks and those sub-tasks are delivered to the cloud; the cloud being composed of a large number of distributed computers throughout the network. In order to quickly handle a large number of users' service requests and return service result data within the cloud environment, it is necessary to conduct reasonable and efficient scheduling of tasks in order to achieve global optimization. In addition, as a large amount of resources and computing are concentrated in the cloud, the energy consumption of the data center will gradually increase, leading to the waste of electricity, reduction in air quality, climate change and other issues.

At present, there are three main technologies to optimize the energy consumption of cloud systems, namely virtualization technology, shutdown/sleep technology, and voltage dynamic adjustment technology, however, all of them have some issues. Virtualization technology can improve the resource utilization of a system, however, virtualization computing itself has a certain amount of energy consumption. Shut down/sleep technology is effective in reducing the level of idle energy consumption, however the long time required for the computer to start from the shut down/sleep state may reduce system performance; deciding to shut down those computers is also a problem for this technology. Dynamic voltage adjustment techniques also often fail to achieve optimal results.

The root problem of energy consumption is task scheduling. A reasonable and efficient task scheduling scheme is the best way to balance system performance and energy consumption. Researchers have proposed a number of effective cloud computing task scheduling algorithms to improve system performance. For example, Seyed Morteza Babamir presents a static task scheduling method based on the Particle Swarm Optimization (PSO) algorithm where tasks are assumed to be non-preemptive and independent. They improved the performance of the basic PSO method using a load-

*E-mail of Haiqin Liu: lhqlhqok@163.com
†E-mail of Haifeng Yi (Corresponding author): 14202812@qq.com
[1]Haiqin Liu, lecturer at Dongchang College of Liaocheng University, Master's degree. Mainly engaged in cloud computing task scheduling and cloud computing information security research, computer software and application research.

$$R=\begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \\ r_{m1} & \cdots & r_{mn} \end{bmatrix}$$

Figure 1 Task and Resource Node Assignment Matrix.

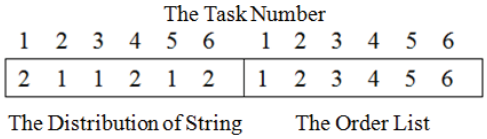

Figure 2 The Chromosome Encoding.

balancing technique [1]. The scheduling algorithm proposed by Fang Wang et al. is based on the Dynamic Adaptive Ant Colony algorithm, adding chaos and disturbance strategy to resource selection, reducing the occurrence probability of a local optimal [2]. Weijun Duan et al. [3] proposed a QoS Constrained Task Scheduling algorithm based on the fusion of a genetic algorithm and an ant colony algorithm. Li Li et al. [4] take into account the time span of task execution and the problems of load balancing and quality of service, defining a fitness function according to the task cost and predicted completion time, effectively balancing the resource load and improving the quality of service. The Cloud Computing Task Scheduling algorithm based on resource pre-classification proposed by Liangliang Feng [5] and the Task Scheduling algorithm based on Particle Swarm Optimization proposed by Shuxia Su hardly discuss energy consumption [6].

This paper proposes a Cloud Computing Task Scheduling algorithm combined with a genetic algorithm and an ant colony algorithm, which uses the excellent solution of the genetic algorithm running in the initial stage to optimize the initial pheromone of the ant colony algorithm, and introduces the energy consumption factor when calculating the fitness function, which improves the time efficiency of task scheduling and effectively reduces the energy consumption.

2. FORMAL DEFINITION OF TASK SCHEDULING IN CLOUD COMPUTING

Task scheduling in the cloud environment is formally defined as: there are n independent sub-tasks in the cloud computing environment, defined as task set $T = t_1, t_2 \ldots, t_n$, where t_n represents the ith subtask. M virtual resource nodes are defined as node set $V = v_1, v_2 \ldots, v_m$, where v_j represents the jth virtual resource node. N subtasks are assigned to execute on m virtual resource nodes, and each subtask can only run on one virtual resource node [7]. The following matrix is used to describe the assignment relationship between the task set T and the resource node set V where r_{ij} represents the allocation relationship between subtask t_j and virtual resource v_i. If task t_j is assigned to resource $r_{ij} = 1$; otherwise, $r_{ij} = 0$.

The goal of task scheduling in this paper is to minimize the total task completion time and reduce energy consumption.

3. DESIGN OF CLOUD COMPUTING TASK SCHEDULING ALGORITHM

3.1 Genetic Algorithm Task Scheduling

The genetic algorithm is a random search algorithm based on a biological evolution mechanism. In combination with the characteristics of the cloud computing environment, coding methods and genetic operators (selection, crossover, mutation) need to be specifically defined.

(1) The Genetic Code

There are many ways to encode chromosomes in genetic algorithms. In this paper, a short real coding method is adopted. This coding method is conducive to searching in a wide range, can improve algorithm efficiency, facilitates the introduction of problem information, and can simply and intuitively represent a solution space.

The chromosome coding representing the solution is divided into two parts, the left part is called the distribution string, which is used to represent the distribution state of the task, and the right part is called the sequence string, which reflects the sequence of task execution.

For example, the code “211212 123456” indicates the following distribution:

The above codes indicate that tasks T_2, T_3 and T_5 are assigned to resource R_1, while tasks T_1, T_4 and T_6 are assigned to resource R_2. The order of task execution on resource R_1 is T_2, T_3 and T_5, while the order of task execution on resource R_2 is T_1, T_4 and T_6.

(2) Fitness function

Considering that fitness function is proportional to individual performance, this algorithm takes the two indexes of task execution time and energy consumption into consideration comprehensively.

The optimal time span [8] is the time required from the execution of the first task to the completion of the last

task in a virtual resource node. The calculation formula is as follows:

$$T = \sum_{i-1}^{M} \left(w_i + \sum_{j-1}^{M} COM_{i,j} \right) \quad (1)$$

where w_i represents the time required to execute Task i, $COM_{i,j}$ represent the communication delay required for Task i and Task j to communicate. If i and j execute on the same virtual resource node, the value of $COM_{i,j}$ is 0.

The formula for calculating energy consumption is shown below:

$$EC = P_{\text{cal}} \times \sum_{i=1}^{M} w_i + P_{\text{com}} \times \sum_{i=1}^{M} \sum_{j=1}^{M} COM_{i,j} \quad (2)$$

where, P_{cal} represents the energy consumption required for calculation per unit time, and P_{com} represents the energy consumption required for communication per unit time, respectively known as calculation energy consumption and communication energy consumption.

The fitness function is shown below:

$$F = 1/(\eta \cdot T + \mu \cdot EC) \quad (3)$$

where η and μ respectively represent the emphasis on time span and energy consumption, namely the weight.

The selection operation uses a roulette, commonly used in genetic algorithm selection, where the probability that an individual is selected is obtained using the fitness function. The specific formula is as follows:

$$P_i = F_i / \sum_{j=1}^{N} F_j \quad (4)$$

where, N represents the population size.

(3) Crossover Mutation

The generation of new individuals in genetic algorithms is mainly realized through crossover and mutation operations.

In this paper, a uniform crossover strategy was adopted to randomly cross each gene on the chromosome according to the same probability. A single-point crossover strategy is adopted for allocation strings, that is, in the two target allocation strings that need to be crossed, one intersection point is randomly found and all the subsequent strings are swapped. The method used for the assignment string must be different to that used for the sequence string, as the sequence of the sequence string determines the order in which the task is executed, and this ensures that the cross-over solution is still feasible. To handle the intersection of sequence strings the intersection points must be identified and then the sequence of the strings after the intersection points is arranged in the order in which they are arranged in each other.

Due to the randomness of crossover operation, some good gene structures may be destroyed during crossover operation, which leads to a premature local convergence of new individuals, namely precocity. Therefore, the mutation operation of chromosomes is very necessary. The mutation operation can increase the diversity of the population and reduce the generation of early maturity. In this paper, the distribution string and the sequence string are mutated. The distribution string is mutated by selecting a random variation point in the distribution string and changing the resource number at that point to another resource number. The variation of the sequence string is conducted by randomly selecting a variation task, then randomly selecting a task assigned to the same virtual resource node with it, and exchanging the order of the two.

3.2 The Strategy of Integrating Genetic Algorithm and Ant Colony Algorithm

In the initial stage of iteration, the Ant Colony algorithm global search is not optimal due to the accumulation of pheromones in each path being fewer, which leads to a little difference among pheromones in each path, affecting the global convergence speed of the algorithm [9]. In contrast, the Genetic algorithm has the ability of fast global search. Therefore, the Genetic algorithm can be applied to the initial phase of the Ant Colony algorithm. After the optimal solution is obtained via the Genetic algorithm, 20% of the optimal individual is used to optimize the initial pheromone value of the Ant Colony algorithm, it can then be converted to an Ant Colony algorithm to continue the search. The combination of the two can greatly improve the global convergence rate.

(1) The Timing of the Switch

In the Ant Colony algorithm, due to the lack of pheromones in the initial stage, the search is blind. With the accumulation of pheromones, the ant's activities will gradually show a certain regularity, the search efficiency will also gradually improve, and the convergence speed will be accelerated.

The convergence efficiency of the Ant Colony algorithm and Genetic algorithm will meet with time. At the beginning, the convergence rate of the Genetic algorithm is relatively fast. After the intersection point, the convergence rate of the Genetic algorithm will decline sharply and eventually drop to a very low value. The Ant Colony algorithm is just the opposite, convergence is slow at first, and then fast after the intersection. Therefore, according to the characteristics of the convergence speed of the two algorithms, the end time of the Genetic algorithm, namely the beginning time of the Ant Colony algorithm, is selected appropriately to effectively improve the overall search efficiency and convergence speed of the algorithm.

(2) The Genetic Optimal Solution is used to Update the Initial Pheromone Value

The initial pheromone value of the Ant Colony algorithm is optimized by using the first 20% of the optimal individual obtained by the Genetic algorithm. Before

optimization, the value on the pheromone concentration table, that is the pheromone concentration, is initially set to the constant τ_{i0}. The pheromone concentration table was updated by using Formula (5) and the pheromone of the good individuals obtained by the Genetic algorithm. When updated, if task i is assigned to the resource node j, the value of the (i, j) point on the pheromone concentration table increases by $\tau_{ij}^{'}$.

$$\tau_{ij} = \tau_{i0} + \tau_{ij} \tag{5}$$

The pheromone value of a resource node with no task assignment remains τ_{i0}.

(3) Next Node Selection

The probability calculation formula of ant k choosing the next resource node j from the current resource node is as follows.

When k $\in$ CL, calculate by Formula (6); otherwise, its value is 0.

$$P_j^k(t) = \frac{\left[\tau_j(t)\right]^{\alpha}\left[\eta_j(t)\right]^{\beta}}{\sum_{k\epsilon CL}\left[\tau_k(t)\right]^{\alpha}\left[\eta_k(t)\right]^{\beta}} \tag{6}$$

When a and $b \in 0, 1$, the value of t he heuristic message is calculated using Formula (7).

$$\eta_j(t) = \phi_j = a \cdot \mathrm{cal}_j + b \cdot \mathrm{Comm}_j \tag{7}$$

The pheromone concentration value of resource node j at time t is expressed as $\tau_j(t)$. The value of the heuristic message is expressed as $\eta_j(t)$. Both α and β are weighting factors that indicate the degree of emphasis on pheromone concentration and heuristic information. The computing power of the resource node j is represented by Cal_j, and the communication power is represented by Comm_j. a and b represent the emphasis factors for the two aspects, and $a + b = 1$. The values of a and b vary according to the user's task requirements.

(4) The Strategy of Pheromone Update

The pheromone concentration was updated according to Formulas (8) and (9).

$$\tau_j(t+1) = (1-\rho) \times \tau_j(t) + \Delta\tau_j \tag{8}$$

$$\Delta\tau_j = \varphi_j \tag{9}$$

$\Delta\tau_j$ represents the increase of pheromone concentration in node j at time $j + 1$. ρ is the volatile factor of the pheromone. The smaller the value of a, the slower the volatilization of the pheromone, and vice versa. The value of ρ can be determined dynamically by using Formula (10) and Formula (11), which can reduce the degree of adverse influence on the algorithm caused by the value of ρ being too large or too small.

First, set the value of $\rho(0)$ to 1.

If $0.95\rho(t-1) >= \rho_{\min}$,

$$\rho(t) = 0.95\rho(t-1) \tag{10}$$

Else,

$$\rho(t) = \rho_{\min} \tag{11}$$

Setting $\rho_{\min}$ as the minimum value of ρ can avoid the phenomenon that the convergence speed of the algorithm is too low because the value of ρ is too small.

Global pheromone update and local pheromone update use the same formula, but the value of $\Delta\tau_{\mathrm{j}}$ is different. When the global update occurs, the pheromone increment of the path node is:

$$\Delta\tau_j = \sum_{j-1}^{n} \varphi_j / n \tag{12}$$

That is, the average value of the comprehensive capacity value φ_{j} of all resource nodes on the path, and n represents the total number of resource nodes.

(5) Algorithm Description

Step 1 Initialize the Genetic algorithm parameters.

Step 2 Population initialization, initial evolutionary Generation number is set to 0.

Step 3 Calculate the adaptive values of the population, perform selection operations, carry out crossover and mutation.

Step 4 Increase the value of the evolutionary Generation number by 1.

Step 5 If the evolutionary Generation number is less than the maximum evolutionary Generation number, go to Step 3. Otherwise, move to Step 6.

Step 6 The initial pheromone value of the Ant Colony algorithm is optimized by using the first 20% of the optimal individual obtained by the Genetic algorithm.

Step 7 Initialize other parameters of the Ant Colony algorithm.

Step 8 Set the maximum number of searches and start the iteration.

Step 9 Initialize the tabu table, and calculate the probability of selecting the next node according to Formula (6) for each ant, determine the next node, refresh tabu table.

Step 10 Update the local pheromone concentration, store the optimal individual.

Step 11 If all ants have completed a task assignment, continue to Step 12, otherwise go to Step 9.

Step 12 Update the global pheromone concentration, increase the number of iterations by 1.

Step 13 Judge whether the number of iterations has reached the maximum. If it has, continue to Step 14; otherwise, go to Step 9 for iteration.

Step 14 Output the optimal solution.

Table 1 Main Run Parameter Settings Table.

Parameter name	Parameter value
Number of genetic populations	30
Crossover probability	0.7
Mutation probability	0.05
Number of genetic iterations	300
Computing power	40W
Communication power	34W
Weight of energy consumption	0.001
Ant colony size	50
Generations number of ants	400
α	1
β	2.7

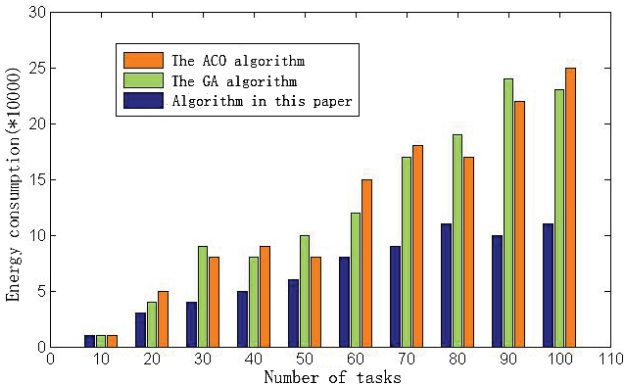

Figure 3 Comparison Figure of Energy Consumption.

4. RESULTS AND ANALYSIS

The simulation experiment platform used in this paper is the CloudSim cloud computing simulation platform developed by the University of Melbourne. The environment used for the experiment is configured as: Windows 7 operating system, CPU frequency 2.13 ghz, 3 GB memory, Eclipse 3.2, JDK 1.6.0.

In the experiment, the number of tasks is set to 10, 20, 30... 100, the number of virtual resource nodes is set to 8, and the main running parameter settings involved are shown in Table 1.

In order to verify the performance of the Optimized Ant Colony algorithm in this paper, in addition to using the Optimized Ant Colony algorithm in this paper to implement task scheduling, the Standard Ant Colony algorithm and Genetic algorithm were also used to implement task scheduling, their running parameters also take the corresponding parameters in Table 1. The running results of the three algorithms were compared and analyzed, as shown in Figure 3.

Figure 3 is a comparison of the energy consumption of the three scheduling algorithms as the number of tasks increases. As can be seen from the figure, the energy consumption of the three algorithms all increase with the increase of the number of tasks, however, the energy consumption growth trend of the Standard Ant Colony (ACO) algorithm and the Genetic algorithm (GA) without considering energy consumption is very high, the energy consumption of the algorithm in this paper also increases with the increase in the number of tasks, but the growth rate is relatively low. In general, the energy consumption of the algorithm in this paper is only close to that of the other two algorithms when the number of tasks is small. When the number of tasks is large, the energy consumption of the algorithm is much lower than that of the other two algorithms. The algorithm in this paper plays an effective role in controlling the increase of energy consumption, as shown in Figure 4.

Figure 4 shows the change curve of task execution time span of the three algorithms as the number of tasks increases. In general, the time span of this algorithm is lower than that of the Ant Colony algorithm and Genetic algorithm. However, in some cases, for example, when the number of tasks is 70, the task execution time of this algorithm is higher than the GA algorithm, when the number of tasks is 80, it is higher than the ACO algorithm. However, when the number of tasks is 70 and 80, the energy consumption of this algorithm is much lower than that of the other two algorithms. To the extent that it remains acceptable to users, it is also possible to trade a slightly longer task execution time for lower power consumption.

5. CONCLUSION

This paper presents an algorithm with a dynamic fusion of the Genetic algorithm and the Ant Colony algorithm, and applies it to cloud computing task scheduling. It has been proved by experiments that it has a higher efficiency of task

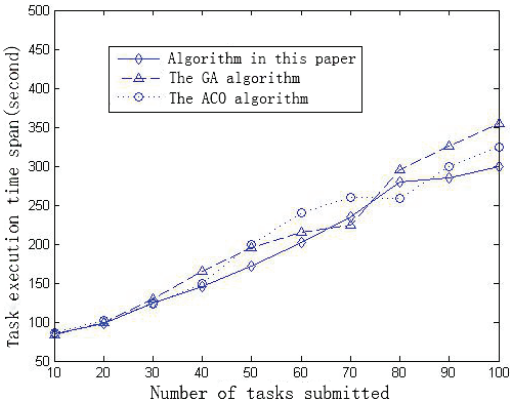

Figure 4 Task Execution Time Span Contrast Figure.

execution time. In this paper, the algorithm not only focuses on improving the efficiency of task execution time, but also takes into account the factor of energy consumption, where a formula for calculating energy consumption is introduced into the fitness function. Experiments show that the proposed algorithm is effective in energy saving. With the continuous development of cloud computing technology and intelligent algorithms, better scheduling algorithms will play a more important role in cloud computing task scheduling.

Acknowledgments

This work was funded by the Research on Education and Teaching Reform (key) project of Dongchang College of Liaocheng University, through the "Research on the application of teaching mode of 'large-class teaching and small-class discussion' based on the status quo of independent colleges" (2018JGA01); and the Ministry of Education industry-academic cooperative education project "Research on practical teaching reform of software engineering (school-enterprise cooperation) specialty under the background of new engineering" (201901173025).

REFERENCES

1. Ebadifard F, Babamir S M, 2018, A PSO-based Task Scheduling Algorithm Improved Using a Load-Balancing Technique for The Cloud Computing Environment, Concurrency Computat Pract Exper. 30(12): e4368.
2. Wang F, et al., 2013, Cloud Computing Task Scheduling based on Dynamic Adaptive Ant Colony Algorithm, Computer Application, 33(11): 3160–3162, 3196.
3. Duan W J, et al., 2014, In the Cloud Computing Environment, Genetic Algorithm and Ant Colony Algorithm are Integrated into QoS Constrained Task Scheduling, Computer Application, 34(S2): 66–69.
4. Li L, Wang J Y, Zang Z X, et al., 2016, Research on Cloud Resource Scheduling based on Improved Ant Colony Algorithm of RBD-DE, Software Guide, 15(3): 1–5.
5. Feng L L, et al., 2013, The Cloud Computing Task Scheduling Algorithm based on Resource Pre-Classification is Tested, Computer Simulation, 30(10): 363–367.
6. Su S X, et al., 2014, Application of Particle Swarm Optimization in Cloud Computing Task Scheduling. Journal of Nanjing Normal University: Natural Science, 37(4): 145–149.
7. Lakra A V, Yadav D K, 2015, Multi-objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization, Procedia Computer Science, 107–113.
8. Srikantaiah S, Kansal A, Zhao F, 2008, Energy Aware Consolidation for Cloud Computing, Hot Power '08 Proceedings of the 2008 conference on Power Aware Computing and Systems, 10–15.
9. Tawfeek MA, El-Sisi A, Keshk AE, et al, 2014, Cloud Task Scheduling based on Ant Colony Optimization, International Conference on Computer Engineering & Systems, IEEE, 64–69.