Cluster Comput
https://doi.org/10.1007/s10586-017-1479-y

Task scheduling of cloud computing using integrated particle swarm algorithm and ant colony algorithm

Xuan Chen[1] · **Dan Long**[2]

Received: 29 September 2017 / Revised: 25 November 2017 / Accepted: 3 December 2017

Abstract The optimization of task scheduling in cloud computing is built with the purpose of improving its working efficiency. Aiming at resolving the deficiencies during the method deployment, supporting algorithms are therefore introduced. This paper proposes a particle swarm optimization algorithm with the combination of based on ant colony optimization, which proposes the parameter determination into particle swarm algorithm. The integrated algorithm is capable of keeping particles in the fitness level at a certain concentration and guarantee the diversity of population. Further, the global best solution with high accurate converge can be exactly gained with the adjustment of learning factor. After the implementation of proposed method in task scheduling, the scheme for optimizing task scheduling shows better working performance in fitness, cost as well as running period, which presents a more reliable and efficient idea of optimal task scheduling.

Keywords Cloud computing · Particle swarm algorithm · Ant colony optimization · Task scheduling

1 Introduction

Cloud computing have been studied for several years already since Ramnath Chellappa's work [1] and developed in several different fields by a number of researchers in the past decade [2]. Cloud computing in commercial use must balance requirements of fast response, cost controlling and algorithm reliability [3]. The trend towards highly-integrated cloud computing system is gradually being implemented to back up or replace the traditional data analysis strategies [4,5]. Thus, the task scheduling step is deemed best able to meet the efficiency and stability demands. Nowadays, the promises of cloud computing have led to the development of task scheduling scheme, with the employment of intelligent algorithm as well as statistical experience [6]. The architecture of task scheduling is sufficiently robust to huge number of tasks so as to ensure the computing capability [7]. The devising of task scheduling calls for more efficient, more reliable and more convenient allocating techniques, along with ever shorter times to commercial use and an increasing demand for quality and advanced functionalities. As a result, Researchers never give up the intention to develop a methodology task scheduling.

The idea of task scheduling algorithm is based on filtering a given set of mutually exclusive constraint task $\{1, \cdots, p\}$ with a mapping from $start_i$ to $Conflict_i$ of each task i, where $start_i$ is the start time and $Conflict_i$ is the performance of task. On the other hand, $deal_{ij}$ is the time needed by the j^{th} resource point to process the task i^{th} in the node sets. The start time is defined as

$$start_i = \begin{cases} 0 & Conflict_i = \phi \\ \sum_{k=1}^{p} deal_{kj} & Conflict_i \neq \phi \end{cases} \quad (1)$$

Each task is performed after the completion of the previous tasks except for the $Conflict_i$ is empty.

In this case, a specific task scheduling time is divided into two parts: suspending time for all the previous tasks in processing sequence and performing time for the current task. For the working period representation, the finishing time end_i refers to the synergistic use of the aforementioned parts:

✉ Xuan Chen
734140999@qq.com

[1] Zhejiang Industry Polytechnic College, Shaoxing 312006, China

[2] Zhejiang University, Hangzhou 310007, China

Published online: 09 December 2017

$$end_i = start_i + deal_{ij} \cdot b_{ij} \tag{2}$$

together with

$$deal_{ij} = \frac{length_i}{capacity_j} \tag{3}$$

where $length_i$ and $capacity_j$ are the time duration of the task and the processing capacity of the node, respectively. The variable b_{ij} stands for the assignment of processing and is normalized in [0, 1] within this research. If $b_{ij} = 1$, the task i is carried out at resource point j; if $b_{ij} = 0$, then the task is suspended. Specifically, each task corresponds to only one resource node and vice versa.

Let m stands for the number of resources and n for the number of tasks. Computation of different resources is constrained by using the following formula:

$$\sum_{j=1}^{m} b_{ij} = 1 \tag{4}$$

On the basic of Eqs. (2) and (3), the processing order can be determined. In general, the time delay for resources switching and instruction transmitting, which is from the resource node to the task, can be ignored. Thus, the processing time from the initial to the node j is

$$over_j = \sum_{i=1}^{n} deal_{ij} \cdot b_{ij} \tag{5}$$

The processing proceeds until all tasks in the system are traversed and addressed. Thereby, time identification of task scheduling depends on the maximum value of processing duration. Seeing that the dealing with resource point contains the conducting of task, we shall also define:

$$time = \max_{1 \le i \le n} \{end_i\} = \max_{1 \le j \le m} \{over_j\} \tag{6}$$

The processing time for task scheduling basis mentioned above is easy to implement based on Eq. (5). Whereas, the task scheduling algorithm here developed is not obtained as the solution in cloud computing. It is also based on the integration of other methods. Optimizing the working performance of the task scheduling, a better computing result can be obtained. Therefore, the improvement of task scheduling plays an important role in the ongoing researches [8–11]. Observing from [12] for an analysis of this issue, a possible solution is proposed. (Related to this also seen in [13].) With the application of virtual machine resources, the tasks are re-allocated and the experimental outcomes are presented. Therefore, the employment of algorithms provides an opportunity for further development and analysis of the task scheduling, which can be helpful in cloud computing.

The remainder of this paper is organized as follows: Section 2 gives a brief description of the theory of Particle Swarm Algorithm and Ant colony algorithm. Section 3 presents the methodology for task scheduling based on integrated Particle Swarm Optimization algorithm. The experiment establishment to verify the proposed scheme and is presented in Sect. 4. Concluding remarks are given in Sect. 5.

2 Related work

In this section, we briefly describe now the theory of particle swarm algorithm and its adjustment as well as the ant colony behavior.

2.1 Particle swarm optimization (PSO)

The particle swarm optimization (PSO) model is originally defined for the study of group hunting of birds [14]. With the definition of iteration evolution, each particle corresponds to a potential solution with certain position and speed, while the particle keeps flying to increase the function fitness until an optimal position is selected [15]. The moving position and speed are determined by two extreme values, i.e., individual extreme value and global extreme value.

Let x be the position of particle and v be the speed. The evolution of PSO can be described as

$$\begin{aligned} v_t(k+1) &= w v_t(k) + c_1 r_1 \left[x_t^{PB}(k) - x_t(k) \right] \\ &\quad + c_2 r_2 \left[x_t^{GB}(k) - x_t(k) \right] \end{aligned} \tag{7}$$

together with

$$x_t(k+1) = x_t(k) + v_t(k+1) \tag{8}$$

where the subscript t is the serial number of particle sample; superscript PB and GB are individual extreme value and global extreme value respectively. Inertia weight coefficient w is in the range of [0.4, 0.9] and k refers to the iterative algebra. c_1 and c_2 represent acceleration constants while r_1 and r_2 are random factors ranging from 0 to 1.

2.2 Improvement of PSO *Gaussian Copual* function

Particle swarm algorithm is even combined with genetic algorithm (GA) to improve its performance. During the running of PSO algorithm, when a particle reaches an optimal position, other particles will soon get close to it. Supposing the current optimal position is the local optimum, the particle swarm will sink into local optimum, which is unable to search in the

solution space. Consequently, the application of genetic algorithm is to resolve the issue of convergence rate and inertia weight for obtaining the diversity of outcome [16]. Nevertheless, the selecting of GA parameter generally depends on experience, which may result in the vulnerable of the convergence stage [17]. In this research, we employ the *Gaussian Copual* function for globally optimizing the solution.

Let$(u, v) \in [0, 1]^2$, a two-dimensional *Gaussian Copual* is expressed as:

$$C_p(u, v) = \Phi_\rho(\Phi^{-1}(u), \Phi^{-1}(v)) \tag{9}$$

where Φ and Φ_ρ represent a standard normal distribution function and a two-dimensional normal joint distribution function, respectively while ρ is a linear dependent coefficient varies from -1 to 1. In line with Eq. (9), we have indeed a two-dimensional normal distribution as $f(x_1, x_2) = (2\pi\sigma_1\sigma_2\sqrt{1-\rho^2})^{-1}\exp\left[-\frac{1}{2(1-\rho^2)}\left(\frac{(x_1-a)^2}{\sigma_1^2} - \frac{2\rho(x_1-\alpha)(x_2-b)}{\sigma_1\sigma_2} + \frac{(x_2-b)^2}{\sigma_2^2}\right)\right]$. Based on the aforementioned characteristic, the calculation of *Gaussian Copual* can be deduced through Eq. (10).

$$C_p(uv) = \int_{-\infty}^{\varphi^{-1}(u)} \int_{-\infty}^{\varphi^{-1}(v)} \frac{1}{2\pi(1-\rho^2)^{\frac{1}{2}}} \exp\left\{-\frac{u^2 - 2\rho uv + v^2}{2(1-\rho^2)}\right\} dudv \tag{10}$$

Similarly, the symmetry of normal distribution can be obtained by the following formula exactly.

$$C_\rho(r_1, r_2) = \Phi_\rho(\Phi^{-1}(r_1), \Phi^{-1}(r_2)) = C_\rho(r_2, r_1) \tag{11}$$

We set $r_1 = r_2$ in the scope of $[0, 1]$ for initialization. Whereas, the variation of these parameter is not constrained but depends on the optimization outcome. The value of correlation coefficients ρ set as 0, 0.5, 0.8 and 1, respectively. The value of r_1 and r_2 is of X axis and Y axis respectively, which approaches the diagonal with the increase of ρ from (Fig. 1). Thus, the trend of PB and GB is in conformity with the change of r_1 and r_2, which can be solved with the regulation of ρ.

2.3 Ant colony optimization (ACO)

Ant colony behavior is a state-of-art model applied in swarm behavior [18]. Ants appear to move randomly and without purposes. However, when all behaviors in the ant colony are considered, it emerges as some kind of collective intelligence for problem resolution. Due to development of swarm intelligence, ant colony optimization is becoming a kind of supporting manner, which is a probabilistic technique for solving computational problems via path choosing [19].

The optimization of ant colony was initially proposed by Morco Dorigo in 1992 [20]. The algorithm originally aimed at choosing the shortest route leading from its colony to a food source. After that, the capability of ant colony optimization was realized and diversified to solve a wider class of numerical problems. For particle swarm optimization, it provides rapid discovery for a good solution, avoiding premature convergence in early stages [21]. The integration of these two methods utilizes the mechanism of a food source that the birds are attracted to. All birds are aware of the location of the target and tend to move towards this target. Similarly, the user also can also get to know the globally best position that one of the member has found. By changing the speed of each member, their rhythmic synchronization and they finally land on the target.

3 Implementation of PSO algorithm in task scheduling based on ant colony behavior

Task execution of cloud computing is based on "decomposition" and "scheduling", which means a larger task is decomposed into many smaller subtasks. Afterwards, a certain number of resource nodes are distributed for these subtasks to perform them. Finally, the running results are sent back to the users. We suppose that the running of the tasks is mutually independent. The aim of task scheduling is to distribute reasonable resource for every task and minimize the total task finishing time.

3.1 Particle coding

In order to employ the particle swarm algorithm and the ant colony algorithm to complete task scheduling, the particles should be coded firstly. We suppose that there are a tasks and $r$resources, and $a > r$. The coded sequence is $(\underbrace{x_1, x_2, \cdots x_n}_{a})(x_k \in [1, r])$, i.e., a particle where each task has a corresponding resource. Then all tasks run by every resource are gained by decoding the particles.

Matrix ETC_{ij} is defined as follows to be used to represent the expected execution time of task i at resource j.

$$ETC_{ij} = \begin{pmatrix} ETC_{11} & ETC_{12} & \cdots & ETC_{1r} \\ ETC_{21} & ETC_{22} & \cdots & ETC_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ ETC_{a1} & ETC_{a2} & \cdots & ETC_{ar} \end{pmatrix}$$

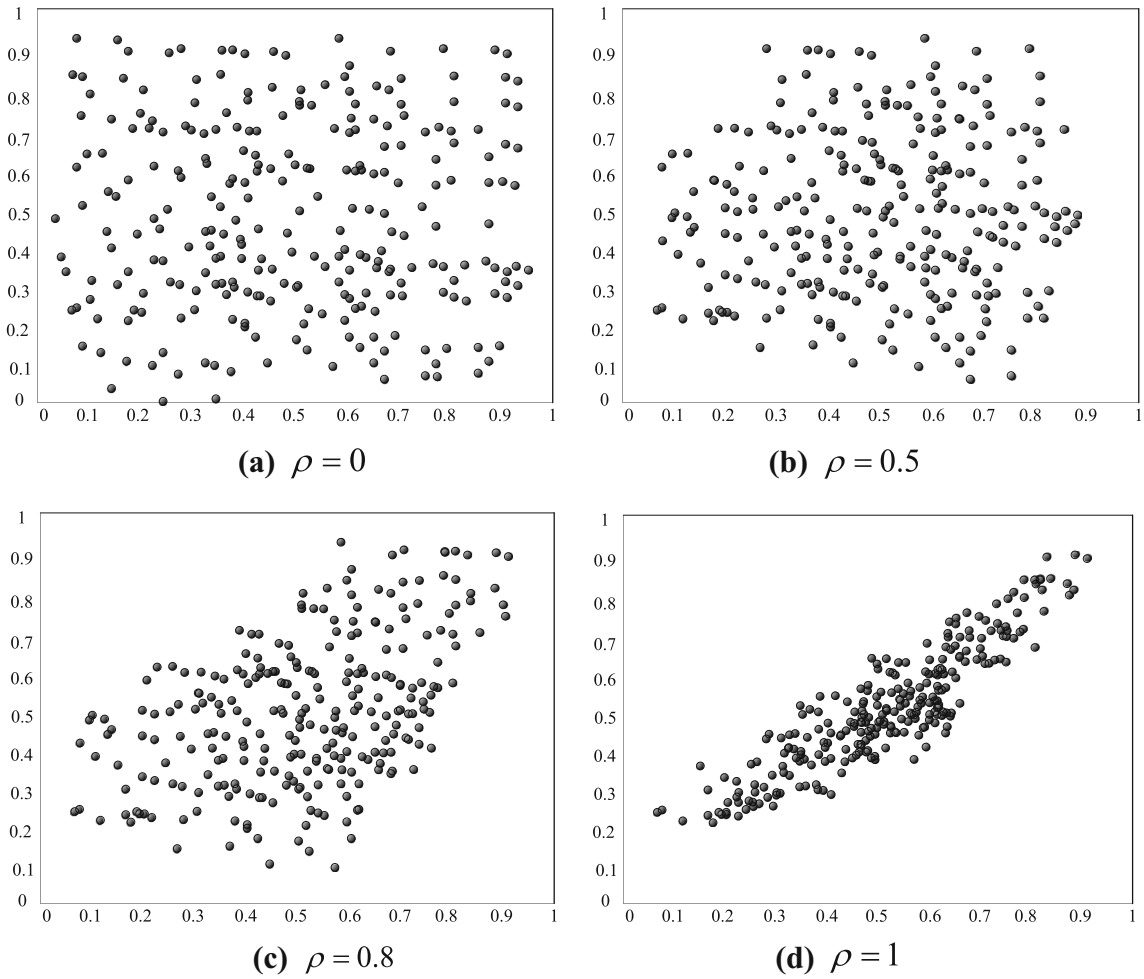

Fig. 1 Variation of random factor based on *Gaussian Copual*

Matrix RTC_{ij} is defined as follows to record the actual execution time of task i at resource j.

$$RTC_{ij} = \begin{pmatrix} RTC_{11} & RTC_{12} & \cdots & RTC_{1r} \\ RTC_{21} & RTC_{22} & \cdots & RTC_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ RTC_{a1} & RTC_{a2} & \cdots & RTC_{ar} \end{pmatrix}$$

The total task completion time at resource j is denoted in equation (12), where i represents the ith task at resource j, and a represents the total number of tasks executed at resource j.

$$resourceTime(j) = \sum_{i=1}^{a} RTC_{ij} (1 \le j \le r) \tag{12}$$

The completion time of total tasks after completing all the scheduling is calculated by equation (13).

$$taskTime = \max(resourceTime(j))(1 \le j \le r) \tag{13}$$

3.2 Particle swarm initialization

Suppose that a represents the number of tasks in a cloud computing, r represents the number of resources in the cloud computing, and s is the number of particles. Let the position of particle i be $x_i = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1r} \\ x_{21} & x_{22} & \cdots & x_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ x_{a1} & x_{a2} & \cdots & x_{ar} \end{pmatrix}$, where there are $x_{ij} \in \{0, 1\}, \forall i, \sum_{j=1}^{r} x_{ij} = 1, 1 \le i \le a$ and $1 \le j \le r$. The speed of particle i is $v_i = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1r} \\ v_{21} & v_{22} & \cdots & v_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ v_{a1} & v_{a2} & \cdots & v_{ar} \end{pmatrix}$, and there are $v_{ij} \in [-v_{\max}, v_{\max}]$, $1 \le i \le a$ and $1 \le j \le r$. When initialization is executed, $s(x_{ij})_{a \times r}$ matrices and $s(v_{ij})_{a \times r}$ matrices are randomly generated as the initial position and speed of the particle.

3.3 Fitness function

Each particle has a fitness to judge its quality [22,23]. The selection of fitness function is not fixed. The fitness function should be reasonably selected according to the unsolved problems, the optimization objective, the computational costs and other requirements. In general, the fitness function should possess the characteristics such as monotony, non-negative and maximization. The total task completion time should be optimized and seen as the judgment standard of task scheduling of cloud computing in this paper. Therefore, the fitness function is defined as (14), where s is the population size.

$$fitness(i) = \frac{1}{taskTime(i)}(i = 1, 2, \cdots s) \tag{14}$$

3.4 Combination of ant colony behavior and PSO algorithm

To further improve the generation quality of the initial pheromone and the convergence speed in the last stage of the ant colony algorithm, partial crossover optimization can be made for the previous particle swarm algorithm [24]. A certain selective probability should be given to every particle according to the fitness value. Crossover and mutation operations should be performed for the particles selected. The positions of two children generated by mutation operation are

$$x_{child,1}(t) = px_{parent,1}(t) + (1-p)x_{parent,2}(t) \tag{15}$$

$$x_{child,2}(t) = px_{parent,2}(t) + (1-p)x_{parent,1}(t) \tag{16}$$

where x_{child} represents the positions of child particles. x_{parent} represents the positions of parent particles. p is random numbers from 0 to 1.

The speed of the progeny particles is:

$$v_{child,1}(t) = \frac{v_{parent,1}(t) + v_{parent,2}(t)}{|v_{parent,1}(t) + v_{parent,2}(t)|} v_{parent,1}(t) \tag{17}$$

$$v_{child,2}(t) = \frac{v_{parent,1}(t) + v_{parent,2}(t)}{|v_{parent,1}(t) + v_{parent,2}(t)|} v_{parent,2}(t) \tag{18}$$

In the formulas, v_{child} represents the speed of the child particles, and v_{parent} represents the speed of the parent particles.

3.5 Implementation steps

Step 1: Define the related parameter values in the algorithm and randomly generate the initial population.

Step 2: Code the particles and initialize the population.

Step 3: Randomly divide the group into two subgroups, and adopt different processing methods for different subgroups.

Step 4: For each subgroup, specific algorithm is utilized to update the position and speed of the particles. After that, the two subgroups are integrated.

Step 5: Judge whether iterative conditions can meet the stop conditions or achieve the iteration times.

Step 6: If iterative conditions cannot meet the stop conditions or achieve the iteration times, please execute step 3. Otherwise, please execute step 7.

Step 7: Select the top10% of individuals with the best fitness values from Step 4 and generate the initial pheromone of the ant colony algorithm.

Step 8: Establish the task scheduling model of the ant colony algorithm and initialize the parameters of the algorithm.

Step 9: Every ant selects transfer nodes, updates the local pheromone and adds the nodes selected to the tabu list.

Step 10: When all the ants complete a cycle, the global pheromone should be updated.

Step 11: Evaluate whether iteration conditions can meet the stop conditions or achieve the iteration times.

Step 12: If the iterative conditions cannot meet the conditions or achieve the iteration times, please execute step 11. Otherwise, the global best solution is obtained.

4 Case analysis

4.1 Experimental setup

To illustrate the advantages of PSO–ACO algorithm, the proposed algorithm is tested in comparison with traditional scheduling algorithms. In the laboratory setup, 5 computers with intel core i3 processor, 4GDDR3 memory and 500 G hard-drive capacity are used as research objects. One computer is set as a server, and the other four computers are used to imitate cloud-based clients. All experimental parameters were showed in Tables 1, 2, and 3.

4.2 Solution of optimal task scheduling based on ACO-PSO

(1) *Iteration frequency determination* Figure 2 describes the iteration time comparison of PSO–ACO algorithm with PSO and ACO algorithms under different task sizes. When the task scale is less than 30,000, the iteration time of the algorithm in this paper is not much different from other algorithms. When the task scale is more than 150,000, the actual iteration time of PSO–ACO algorithm is between 85 and 90, i.e., the best solution can also be gained without the execution of

Table 1 Parameters of PSO–ACO algorithm

Parameter name	Parameter symbol	Parameter value
Population size	s	100
Number of resources	r	5
Number of tasks	a	[2,110000]
Controlling factor	k	3.0
Maximum inertia weight	$w_{\max}$	0.9
Minimum inertia weight	$w_{\min}$	0.4
The initial iterative value of c_1	c_{1s}	2.0
The last iterative value of c_1	c_{1e}	0.5
The initial iterative value of c_2	c_{2s}	1.5
The last iterative value of c_2	c_{2e}	3.0
Crossover probability	P_c	0.9
Mutation probability	P_m	0.02
The maximum iteration time in the first stage	$t_{1\max}$	50
The maximum iteration time in the second stage	$t_{2\max}$	150
Heuristic factor	α	1.0
Heuristic factor	β	1.0
Volatilization factor	ρ	0.2

Table 2 Parameters of ACO algorithm

Parameter name	Parameter symbol	Parameter value
Number of resources	r	5
Number of tasks	a	[2,110000]
Heuristic factor	α	1.0
Heuristic factor	β	1.0
Volatilization coefficient	ρ	0.2
The maximum iteration time	$t_{\max}$	200

approximately 10% of the optimization. The analysis above demonstrates that PSO–ACO algorithm has certain advantages on the aspect of big task scheduling.

(2) *Variance determination* To better verify the superiority of PSO–ACO algorithm compared to the other two algorithms, we executed three algorithms for 100 times respectively and then a variance statistical analysis is performed on the best solutions gained. Figure 3 shows the best solution is mainly the minimum value of the solution execution costs in previous iterations. Because different tasks have different lengths, the calculated variances are different. The variance value of the best solution of PSO–ACO algorithm is more stable compared to other algorithms with the increase in task size, which is proved to effectively adapt to task distribution in cloud computing.

Table 3 Parameters of PSO algorithm

Parameter name	Parameter symbol	Parameter value
Population size	s	100
Number of resources	r	5
Number of tasks	a	[2,1100000]
Controlling factor	k	3.0
Maximum inertia weight	$w_{\max}$	0.9
Minimum inertia weight	$w_{\min}$	0.4
The initial iterative value of c_1	c_{1s}	2.0
The last iterative value of c_1	c_{1e}	0.5
The initial iterative value of c_2	c_{2s}	1.5
The last iterative value of c_2	c_{2e}	3.0
The maximum iteration time	$t_{\max}$	200

4.3 Experimental results analysis

Experiments are conducted to better illustrate the efficiency of the algorithm. A certain number of nodes are established while the number of tasks twice as many as the node. The PSO–ACO algorithm, the ACO algorithm and the PSO algorithm are adopted for contrast experiments under the same conditions.

(1) *Fitness function evaluation* With the number of cloud computing resources equal to 20 and the number of task equal to 20,000, the solution curves of the best scheduling schemes of three algorithms are shown as in Fig. 4. The PSO–ACO algorithm gains the best scheme when the time is 130. The PSO algorithm gains the best scheme when the time is 150. The GA algorithm also gains the best scheme when the time is 170. At the same time, extreme disturbance, inertia weight and learning factor are improved to improve the local convergence speed of the algorithm.

(2) *Completion time evaluation* In the cloud computing system with the number of cloud resources at 50 and the number of tasks from 20,000 to 100,000, the completion times of the best cloud computing resource scheduling schemes of the ACO algorithm, the PSO algorithm and the PSO–ACO algorithm are shown in Fig. 5. The competitions between tasks become fiercer, the conflict probability of the tasks increases and the task completion times of all the algorithms increase with the increase in the number of tasks. Results show that the PSO–ACO algorithm can improve the convergence speed, avoiding the local best solution of the algorithm and be more suit-

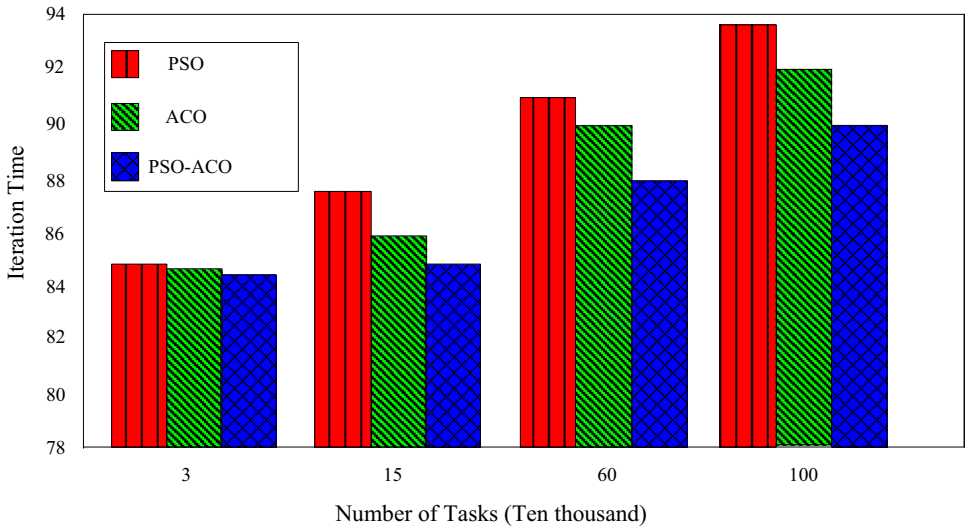

Fig. 2 Iteration time comparison of three algorithms

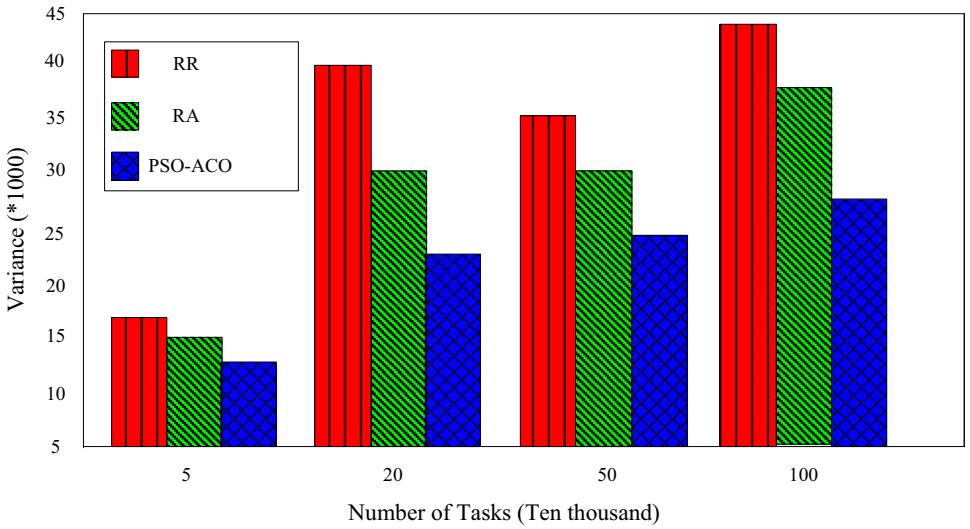

Fig. 3 Variance comparison of three algorithms

able for the solution of the cloud computing resource scheduling problem for large scale tasks by improving the particle swarm algorithm.

(3) *Execution costs evaluation* An important standard to measure the algorithm performance in cloud computing is how to effectively reduce the execution costs. Execution costs mainly refer to the costs of every task consumed in the distributed virtual machine. Figure 6 shows that there is not much difference between the execution costs of PSO–ACO algorithm and the execution cost rates of the other algorithms when the task scale is not large. The differences between the three curves increases with the increase of the task scale, which indicates the proposed algorithm can adapt the tasks with larger scale.

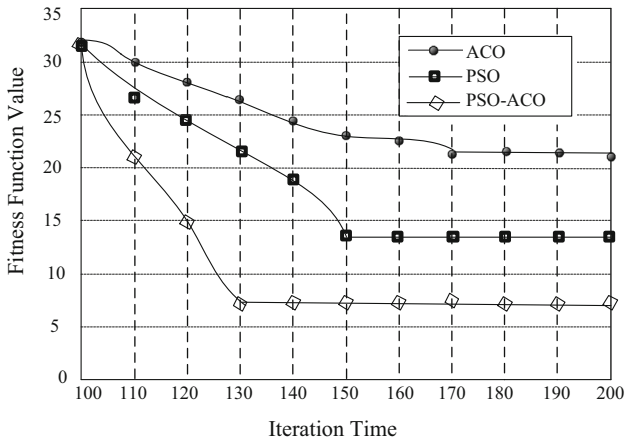

Fig. 4 Iteration curves of three algorithms

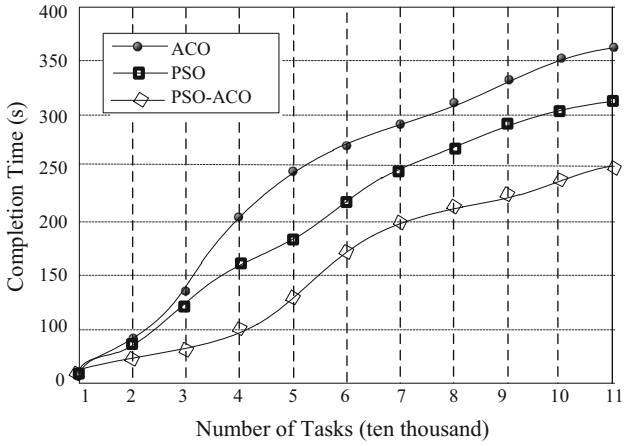

Fig. 5 Completion time comparison of three algorithms

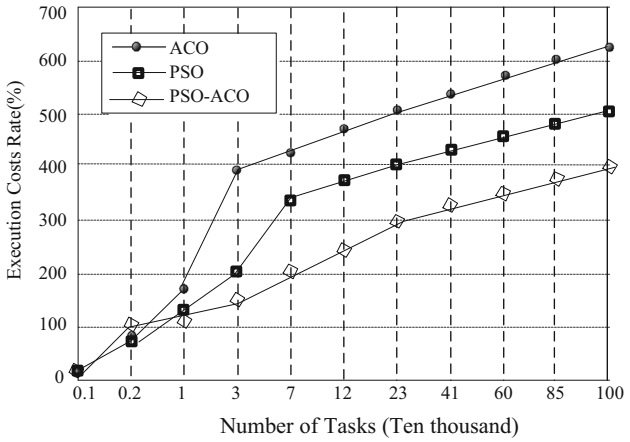

Fig. 6 Execution costs comparison of three algorithms

5 Conclusions

This work has presented and extended the task scheduling by PSO–ACO algorithm in cloud computing. Though a lot has to be done in establishing a comprehensive task scheduling framework, the nature of PSO algorithms has to be preserved while the convergence and iteration of such algorithm should be optimized. Also from the analysis it is clear that ACO and *Gaussian Copual* function can be integrated under the consideration of PSO algorithm vulnerability. The experiment of task scheduling optimization is carried out in laboratory environment, in order to obtain a practical and feasible improvement scheme with the combination of PSO–ACO algorithm. Experimental results indicate that the improved PSO algorithm in this paper can better solve the fitness, cost and running time issues. The optimization results are obviously better than those gained by using traditional PSO algorithm, thus providing an efficient method for the optimal task scheduling.

Future work will address more complex situations where multiple tasks are present to explore whether the proposed algorithm can be extended to a multi-scheduling method. We would also like to investigate the impact of PSO–ACO algorithm within other issues.

References

1. Chellappa, R.K.: Intermediaries in Cloud-Computing: A New Computing Paradigm, INFORMS Annual Meeting, Dallas, 26–29 Oct 1997
2. Takabi, H.: A semantic based policy management framework for cloud computing environments, Doctor Dissertation, University of Pittsburgh, Pittsburgh (2013)
3. Yi, P.: Peer-to-peer based trading and file distribution for cloud computing, Doctor Dissertation, University of Kentucky, Lexington (2014)
4. Egedigwe, E.: Service quality and perceived value of cloud computing-based service encounters: evaluation of instructor perceived service quality in higher education in Texas, Doctor Dissertation, Nova Southeastern University, Fort Lauderdale (2015)
5. Rochwerger, B., Breitgand, D., Levy, E., et al.: The reservoir model and architecture for open federated cloud computing. IBM J. Res. Dev. **53**(4), 1–17 (2009)
6. Nurmi, D., Wolski, R., Grzegorczyk, C. et al.: The eucalyptus open-source cloud-computing system. In: Proceeding of the CRID, pp. 124–131 (2009)
7. Ochwerger, B., Breitgand, D., Levy, E., et al.: The reservoir model and architecture for open federated cloud computing. IBM J. Res. Dev. **53**(4), 1–17 (2009)
8. Li, J.Y., Mei, K.Q., Zhong, M., et al.: Online optimization for scheduling preemptable tasks on IaaS cloud systems. J. Parallel Distrib. Comput. **72**(2), 666–677 (2012)
9. Etminani, K., Naghibzadeh, M.A.: Min-min max-min selective algorithm for grid task s cheduling. In: 3rd IEEE/IFIP International Conference in Central Asia on Internet. IEEE Computer Society, Washington, pp. 1–7 (2007)
10. Xie, L.X.: Analysis of service scheduling and resource allocation based on cloud computing. Appl. Res. Comput. **32**(2), 528–531 (2015)
11. Shi-yang, Y.: Sla-oriented virtual resources scheduling in cloud computing environment. Comput. Appl. Softw. **32**(4), 11–14 (2015)
12. Guo, L., Zhao, S., Shen, S., et al.: Task scheduling optimization in cloud computing based on heuristic algorithm. J. Netw. **7**(3), 547–553 (2012)
13. Li, J., Peng, J., Cao, X., et al.: A task scheduling algorithm based on improved ant colony optimization in cloud computing environment. Energy Proc. **10**(13), 6833–6840 (2011)
14. Kennedy J, Eberhart R. Particle swarm optimization[C], Proceedings of IEEE International Conference on Networks, 1995: 39-43
15. Graham, J.K.: Combining particle swarm optimization and genetic programming utilizing LISP, Master Dissertation. Utah State University, Logan (2005)
16. Juang, C.F.: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans. Syst. Man Cybern. **34**(2), 997–1006 (2004)
17. Eberhart, R., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation, pp. 84–88 (2000)
18. Zuo, L., Shu, L., Dong, S., Zhu, C., Hara, T.: A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. IEEE Access **3**, 2687–2699 (2015)

19. Deneubourg, J.L., Pasteels, J.M., Verhaeghe, J.C.: Probabilistic behaviour in ants: a strategy of errors. J. Theor. Biol. **105**(2), 259–271 (1983)
20. Dorigo, M.: Optimization, learning and natural algorithms. Doctor Dissertation, Pilotenico di Milano, Italie (1992)
21. Prakasam, A., Savarimuthu, N.: Metaheuristic algorithms and probabilistic behaviour: a comprehensive analysis of ant colony optimization and its variants. Artif. Intell. Rev. **45**(1), 97–130 (2016)
22. Cha an-min.: Research on task scheduling based on particle swarm and ant colony algorithm for cloud computing. Master Dissertation, Nanjing University of Aeronautics and Astronautics, Nanjing (2016)
23. Jiang, M., Luo, Y.P., Yang, S.Y.: Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. Inf. Process. Lett. **102**(1), 8–16 (2007)
24. Gutjahr, W.J.: A graph-based ant system and its convergence. Future Gener. Comput. **16**(8), 873–888 (2000)

Xuan Chen is an associate professor at School of Design and Art at Zhejiang Industry Polytechnic College. He got a master's degree in Computer Science from University of Electronic Science and Technology of China in 2013 and has held and presided over 8 projects at the department and municipal level. His research direction is cloud computing, wireless sensing and algorithm design.

Dan Long is a Ph.D. at the Medical Image R&D Center, Faculty of Science, Zhejiang University. He obtained the doctoral degree from Zhejiang University in 2012 and has participated in 2 national-level natural fund research projects. His research direction is algorithm design and image processing.