Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 6, June 2018, pp. 1161-1170
DOI: 10.23940/ijpe.18.06.p7.11611170

Optimization of Particle Genetic Algorithm based on Time Load Balancing for Cloud Task Scheduling in Cloud Task Planning

Yenzhen Zhang, Shouming Hou*, and Li Chang

School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, 454000, China

Abstract

To solve the problems of long time consumption, imbalanced time load and low resource utilization for cloud task scheduling in cloud task planning, we propose an optimized strategy of particle genetic algorithm based on time load balancing. This strategy was adopted to improve the quality of particles by optimizing particle initialization operation. To ensure that better particles capable of more balanced time load are selected, a model of fitness in time load balancing was established. To prevent the particles from jumping out of the specified area in iterations, the element values of their location and velocity were processed in a standardized way. Finally, genetic crossover and mutation operators were introduced to avoid leading the algorithm to local optimization. This strategy could effectively improve the convergence rate of the particle genetic algorithm and the quality of solutions. The experimental results showed that the algorithm had greater power to search for a better global optimal solution, consumed less time, and reached a more balanced time load. With this algorithm, we may achieve better and more logical task scheduling sequences. Simultaneously, the idea owns a certain degree of practicality and generalization in many fields.

Keywords: model of time load balancing fitness; element value standardization; cloud task scheduling; particle genetic algorithm

(Submitted on March 7, 2018; Revised on April 29, 2018; Accepted on May 24, 2018)

1. Introduction

In cloud task planning, cloud computing is one of the most important core technologies. Cloud computing is the commercial development of grid computing, distributed computing, and parallel computing, and it is inherently more advantageous than other computing patterns [2,12]. Cloud computing systems allocate resources according to task requests and task attributes [8], but complicated cloud computing environments turn task scheduling into a very challenging problem [1,3,15,18]. Simultaneously, the time and cost of task scheduling become the criteria to measure the efficiency of algorithms and a goal to pursue.

In the cloud computing environment, tasks and resource nodes exhibit the properties of being heterogeneous, dynamic and random. Particle swarm optimization (PSO) is a stochastic optimization based on group intelligence, and it owns some flexibility in choosing the task nodes. In PSO, a particle stands for a possible solution to the problem. Particles change their positions via internal interaction of information within the group, so as to reach smart solutions. Presently, scholars at home and abroad have carried out a large amount of research on relevant algorithms of the paper. The reference [22] proposed an improved genetic particle algorithm, which can diversify the particles by adding the crossover and mutation operations of the genetic algorithm to expand the searching space of the particle and increase the global optimization strength. On this basis, reference [6] proposed a relatively efficient task scheduling algorithm by introducing the chaotic perturbation idea. Specifically, a chaotic sequence is generated by chaos search to replace the precocious particles, and it then makes precocious particles jump out of the local optimum and find the optimal solution as soon as possible, thereby improving the efficiency of cloud platform task scheduling. In reference [7], in order to solve the contradiction between population diversity and convergence in the PSO algorithm, the paper put forward an idea of optimizing particle swarm optimization by combining reverse learning with local learning. It then used the reserved particle location information to guide the part of the particles to

* Corresponding author.
E-mail address: housm@163.com

quickly reverse learning, quickly leading to the local optimum area, thereby improving the diversity of particle populations. The optimal particle is guided by the difference between the optimal particles for local learning and searching, and the particles can be dynamically adjusted at any time, so as to improve the solution accuracy and convergence speed. However, there is still a problem of unbalanced load and high energy consumption. In reference [19], the load balancing model was added to realize the balance of the task number at each resource node [13,14,17,20,23, 24,], but this balance was the only evenly distributed task number at each resource node.

In summary, optimized algorithms of task scheduling in cloud computing are constantly emerging, but these algorithms are basically targeted at completion time, load balancing and cost of tasks [5,9,10,16,25], whereas the attention on time load balancing at resource nodes is relatively scarce. In this paper, we started the optimization from the initialization mechanism for PSO in order to improve particle quality. Meanwhile, to avoid particles from running out of the specified area, the particle position elements were normalized. Then, the more logical and optimized particles were selected by establishing the time load balancing fitness function. At last, in order to avoid heading to the local optimum, crossover and mutation probability functions in line with the idea of genetic crossover and mutation were introduced. The algorithm effectively raised the capabilities of global search and convergence, reduced time consumption and achieved time load balancing.

2. Description of task scheduling

In cloud computing systems, large tasks are often divided into sub-tasks, and then the sub-tasks are allocated to different resource nodes in line with the allocation strategy. Assume that there are m tasks and n processing nodes and $m \geq n$. The task lengths are, $l_1, l_2, \cdots l_i, \cdots l_n$ and the computing capacities of the processing nodes per unit time are $cp_1, cp_2, \cdots cp_j, \cdots cp_m$. The time required for each task to be processed at each node is as follows Equation (1):

$$t_{ij} = \frac{l_i}{cp_j} \tag{1}$$

Use the matrix **t** with a scale of $m \times n$ to store t_{ij} values. After scheduling, each processing node has its own task execution sequence.

Each particle position in the particle swarm algorithm represents the sequence of nodes in which all tasks are executed, i.e. a potential solution. Assume that the current position of the $s-th$ particle in the particle swarm algorithm is $x_s = (x_{s1}, x_{s2}, \cdots x_{si}, \cdots x_{sm})$, and $x_{si} \in [1, n]$. The current velocity is $v_s = (v_{s1}, v_{s2}, \cdots v_{si}, \cdots v_{sm})$, and $v_{si} \in [-n, n]$. For instance, there are 10 tasks and 4 nodes. The particle position element sequence is (3,2,2,1,2,3,1,4,4,1), i.e. task 1 is executed at processing node 3, and so on until task 10 is executed at processing node 1. The total task completion time T_j at node j is shown as Equation (2):

$$T_j = \sum_{i=1}^{m} t_{ij} \bullet e_{ij} \qquad (1 \leq j \leq n) \tag{2}$$

In which i indicates the task number, j indicates the node number, e_{ij} indicates that if task i is run at processing node j, the symbolic value is 1; otherwise, the value is 0.

The total task completion time is the maximum time each node takes to complete the task, which is shown in the following Equation (3):

$$T = \max_{j=1}^{n}(T_j) \tag{3}$$

In this paper, the states of the processing node include the running and the idle states. To better describe the resource utilization rate of nodes, the ratio between the running time and the task completion time is used to quantify and describe it. The corresponding description is as follows Equation (4):

$$\mu = \frac{\sum_{j=1}^{n} T_j}{n * \max_{j=1}^{n}(T_j)} \tag{4}$$

3. Algorithm Description

3.1. Particle Swarm Optimization

Particle swarm algorithm is a heuristic and stochastic optimized algorithm based on group intelligence. Particles can change their positions in line with their own speed and the interaction information within the group. It owns some advantages, such as search speed, high efficiency, conciseness, and suitability for scheduling optimization problem handling, and thus it realizes smart solutions. But the algorithm demonstrates premature convergence and easily heads to local optimal solutions. Therefore, considering the goal of optimization in this paper, the algorithm needs to be further improved.

3.1.1. Initialization Improvement

The traditional particle initialization randomly generates s particles, but the particles generated are often of poor quality. For instance, the distribution of tasks on the nodes is uneven; time load on the nodes is consequently not balanced and the operating efficiency is low. So, building a high-quality particle population can effectively improve the operating efficiency.

Of all the particles, forty percent are generated as follows: based on uniform distribution of nodes, the deviation in the number of tasks assigned for each node does not exceed 1, so the generated particles do not only improve resource utilization, but also the time load at each node is relatively balanced. The rest of the particles are randomly generated in the traditional way.

3.1.2. Establishment of Multi-objective Fitness Function

In the search, the particle swarm algorithm is based on the fitness function, and the particle fitness value is viewed as the judgmental basis to search for and update individual and global optimal solutions. Therefore, the particle fitness function is extremely important, which directly affects the convergence speed of the algorithm and the degree of optimization of final solutions. The total time of completing the task is one of the goals to be improved and achieved in this paper, and it is an indispensable part of the multi-objective fitness function. Time load balancing at the nodes is another important goal to be considered in this paper, which can greatly improve the utilization rate of resources, and greatly reduce the idle time of each node, so that the task completion time distribution of each node is relatively concentrated. So, time load balancing at the nodes is another part of the multi-objective fitness function to be considered. In order to prevent the imbalance of the two impact factors in calculating the multi-objective fitness function value, we determine the weight coefficient of the two impact factors k_1 and k_2 with several experiments, and the values are 0.9 and 0.3, respectively. Since we will select the more adaptable particles in the particle selection and update, the multi-objective fitness function based on time load balancing is as follows Equation (5):

$$f = \frac{Q}{k_1 * T + k_2 \sum_{j=1}^{n} (T - T_j)} \tag{5}$$

In which Q is a constant, and to improve the accuracy of the fitness value, it is set at 1000; $T - T_j$ is the difference between the total task completion time and the task completion time of each processing node.

3.1.3. Location and Speed Updates

The particles calculate the position of the next-generation particles according to their own position and speed, and then we decide the quality of the next-generation particles in relation to the parent particles based on the fitness function value. We ensure the quality of the next-generation particles by selecting superior particles and removing inferior ones. At last, the global optimal solution is selected by comparing it to the current global optimal fitness value. In order to avoid the phenomenon

where there still exists local optimization and waste of resources in the particle swarm algorithm, particle position and velocity are updated by referring to document [4]. See the following Equation (6) and Equation (7):

$$v_{si}^{k+1} = \omega \bullet v_{si}^{k} + \xi \bullet \frac{1}{k} \bullet r_1^k (p_{si}^k - x_{si}^k) + \eta \bullet k^2 \bullet r_2^k (p_{gi}^k - x_{si}^k) \tag{6}$$

$$\omega = 0.9 - \frac{0.5(k-1)}{NC-1} \tag{7}$$

In which k is the number of iterations, NC is the total number of iterations and its value is 100, ξ takes a relatively large value, η takes a relatively small value, r_1 and r_2 are random numbers evenly distributed in the range [0,1]. v_{si}^k stands for the $i-th$ velocity of the $s-th$ particle in the $k-th$ iteration. p_{si}^k and p_{gi}^k are the values of the $i-th$ velocity of p_s^k and p_g^k, respectively. In addition, p_s^k and p_g^k respectively stand for the current best position of the particle itself and the global particles in the $k-th$ iteration. The related function are Equation (8) and Equation (9):

$$p_s^k = \begin{cases} x_s^{k+1} & , \quad k = 0 \\ \max(f(p_s^{k-1}), f(x_s^{k+1})) & , \quad k \geq 1 \end{cases} \tag{8}$$

$$p_g^k = \left\{ p_s^k / f_{\max} = \max(f(p_1^k), f(p_2^k) \cdots\cdots, f(p_s^k) \cdots\cdots, f(p_N^k)) \right\} , \qquad k \in [1, NC] \tag{9}$$

The corresponding position of the particle is updated as follows Equation (10):

$$x_{si}^{k+1} = x_{si}^k + v_{si}^{k+1} \tag{10}$$

Equations (6), (7), (8), (9) and (10) are used to update the velocity and position of the particles, and the quality of the particles is decided according to the multi-objective fitness function. In the end, the particles with the greatest fitness value are obtained, i.e. the optimal scheduling sequence takes the total task completion time and time load balancing into account.

3.1.4. Element Standardization

In task scheduling, each dimensional element value in the particle position is the number of the corresponding processing nodes and it is a natural number, i.e. $x_{si} \in [1, n]$. However, after iterations according to Equations (6), (7), (8), (9) and (10), supposing the particle position element value is x'_{si}, its value range becomes $[-n+1, 2n]$: not all natural numbers, and it also goes beyond the actual value range, so the element values need to be normalized. In the process, to ensure that the relations between each dimensional element remain constant in the particle swarm algorithm, we need to contract them in proportion. In this paper, we use absolute value, logarithm, rounding and introduce adjustment constant to standardize the particle position element value, as shown in Equation (11):

$$x_{si}^{k'} = round(\frac{\log(\exp(|x_{si}^k|) + e) + C}{2}) \tag{11}$$

In which C is the adjustment constant and its value is 0.4. Its function is to slightly adjust the range of the function value. e is the constant and its value is 2.71828.

Considering that the particles may fly out of the scope of the feasible domain, we need to standardize the particles' current speed element values. Suppose the current speed is v_{si}^k, and the detailed treatment is shown in Equation (12) below.

$$v_{si}^{k'} = \begin{cases} -n & , \quad v_{si}^{k} < -n \\ v_{si}^{k} & , \quad -n \le v_{si}^{k} \le n \\ n & , \quad v_{si}^{k} > n \end{cases} \tag{12}$$

3.2. Genetic Crossover and Mutation

Genetic algorithm is a smart method that simulates the natural evolution process to search for the optimal solution. It is a popular algorithm for task scheduling in the cloud computing environment. Genetic crossover and mutation operations constantly generate new individual particles, extend searching space, mitigate the local optimum and search for global optimal solutions. In this paper, GA mutation and crossover operators are introduced to find the best solutions in the PSO algorithm quickly. Assume that the standard values of crossover and mutation probabilities are PC and PM, and their values are 0.9 and 0.01, respectively.

3.2.1. Crossover Operator

The larger the individual fitness value is, the smaller the crossover probability becomes, which is conducive to the late evolution of the population, but bad to the initial stage. Therefore, we need to make further improvement. In this paper, we employ the method of adaptive random crossovers to randomly select two individuals (i.e. two particles) from the population and calculate their crossover probability value. The crossover probability function is shown in the following Equation (13).

$$P_c = \begin{cases} \dfrac{c_1(f_{\max} - f') + c_2(f' - f_{avg})}{f_{\max} - f_{avg}} & , \quad f' > f_{avg} \\ c_1 & , \quad f' < f_{avg} \end{cases} \tag{13}$$

In which f' is the maximum fitness value of the two particles, $f_{\max}$ is the maximum fitness value of the global particle, f_{avg} is the average fitness value of the global particle, and c_1 and c_2 are crossover factors.

If $P_c \ge PC$, then they do not cross over; otherwise, they cross over in the following way. Assume that the two particles are $x_{s_1} = (x_{s_1 1}, x_{s_1 2}, \cdots x_{s_1 i}, \cdots x_{s_1 m})$ and $x_{s_2} = (x_{s_2 1}, x_{s_2 2}, \cdots x_{s_2 i}, \cdots x_{s_2 m})$. They randomly generate two crossing points i' and i'', and the sequences of the two particles between the two crossing points exchange. Then, the two particle sequences generated are $x_{s_1} = (x_{s_1 1}, x_{s_1 2}, \cdots x_{s_1 (i'-1)}, x_{s_2 i'}, \cdots x_{s_2 i''}, x_{s_1 (i''+1)} \cdots x_{s_1 m})$ and $x_{s_2} = (x_{s_2 1}, x_{s_2 2}, \cdots x_{s_2 (i'-1)}, x_{s_1 i'}, \cdots x_{s_2 i''}, x_{s_2 (i''+1)}, \cdots x_{s_2 m})$.

3.2.2. Mutation Operator

This paper adopts an adaptive random mutation to randomly select an individual (i.e. a particle) and calculates the probability of its mutation. The mutation probability is shown in the following Equation (14):

$$P_m = \begin{cases} \dfrac{m_1(f - f_{avg}) + m_2(f_{\max} - f)}{f_{\max} - f_{avg}} & , \quad f > f_{avg} \\ m_1 & , \quad f < f_{avg} \end{cases} \tag{14}$$

In which f is the fitness function value of the particle. If $P_m \ge PM$, then it does not mutate; otherwise, it mutates in the following way: two mutated points i' and i'' are randomly generated, and the particle mutation randomly occurs within the two points.

This crossover and mutation operation not only simulate random variation between individual genes, but also fit the optimization goal of the paper's algorithm. It can influence the direction and magnitude of the crossover and mutation according to historical optimal value and individual optimal value, and improve the adaptability of the individual particle to the evolutionary environment.

Within the idea of this article, the corresponding flow chart is shown in Figure 1,

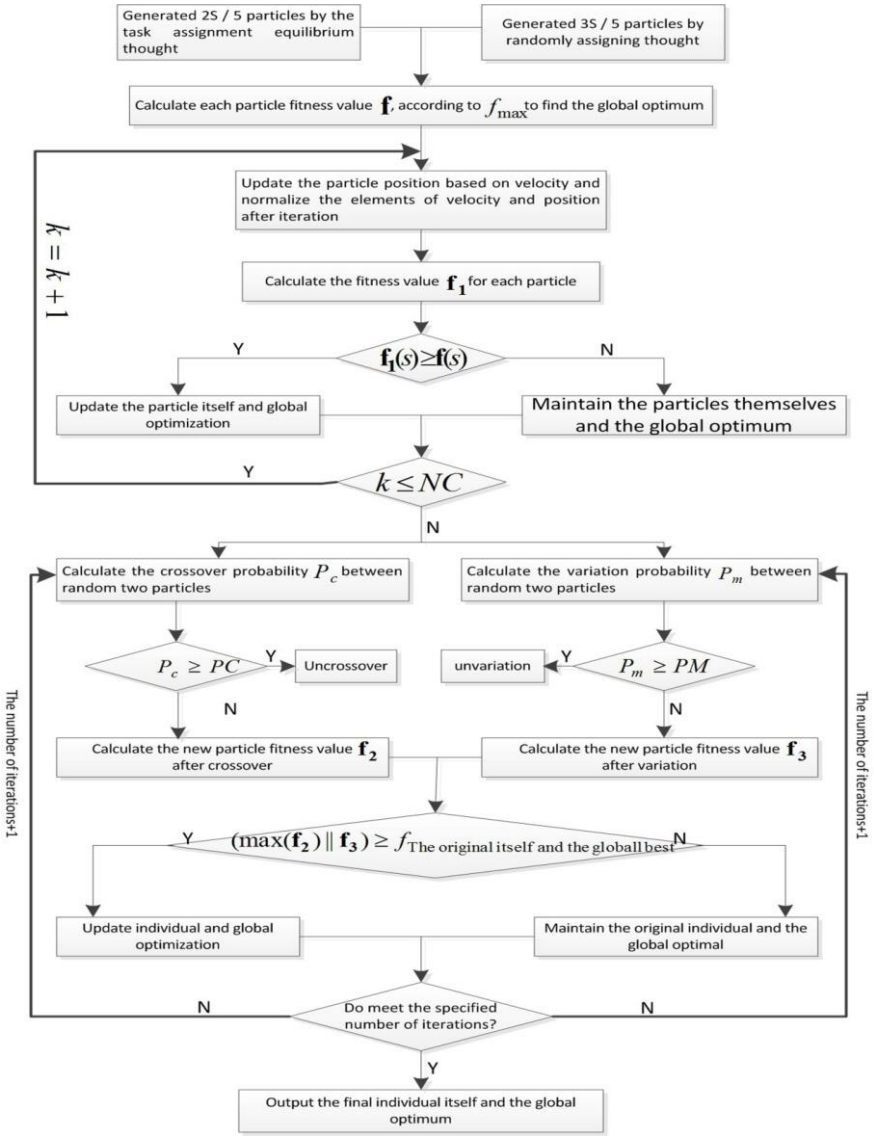

Figure 1. Flowchart of Algorithm

4. Simulation Experiment and Result Analysis

In the paper, the hardware platform of simulation includes the processor Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz, the memory 4.0G, and the graphics card NVIDIA GeForce GTX 750. The software platform includes Window 10 and MtalabR2014. The parameter settings in the simulation are shown in Table 1.

Table 1. The parameter value setting

parameter name	Parameter symbol	Parameter value
Task length	l	[0.1M,1M]
Processing node capability	cp	[100,1000]
Population size	S	100
The number of tasks	m	[50,500]
The number of resources	n	10
The constant of learning factor	ξ	65
	η	0.001
The constant of crossover probability function	c_1	0.9
	c_2	0.6
The constant of mutation probability function	m_1	0.1
	m_2	0.001

4.1. Analysis of multi-objective fitness function value

In this paper, the fitness function takes into account the relation between the total task completion time and time load balancing, which avoids selecting inferior particles and making poor selections under the influence of a single factor. In this simulation, we refer to the data from the genetic annealing algorithm in document [11]; the number of tasks is 10, the length is (19365,49809,30218,44157,16754,18336,20045,31493,30727,31017), the number of nodes is 4 and the node processing capacity is (278,289,132,209). When the total task completion time is 334.52s, the fitness function value and the corresponding task assignment sequence are shown in Table 2 below. The relation between the fitness values is shown in Figure 2 below.

Table 2. Table of related data

Sequence number	The corresponding elements in the sequence										The value of Fitness
1	2	4	1	3	2	4	2	1	1	2	3.184
2	2	4	1	3	2	2	4	1	1	2	3.1909
3	1	4	2	3	1	1	4	2	1	2	3.1878
4	4	1	2	3	1	1	4	2	1	2	3.1839
5	4	2	1	3	4	2	2	1	4	1	3.1789
6	1	1	2	3	2	2	1	4	2	4	3.1601
7	1	1	2	3	2	2	1	4	4	2	3.159
8	2	4	1	3	2	2	4	1	2	1	3.191
9	4	1	2	3	1	4	1	4	2	2	3.1858
10	1	4	2	3	4	1	1	2	1	2	3.176

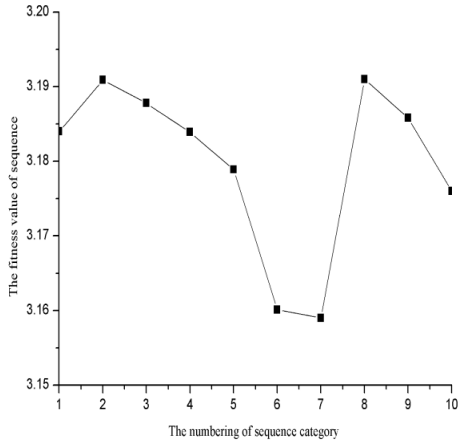

Figure 2. Corresponding fitness value of the sequence

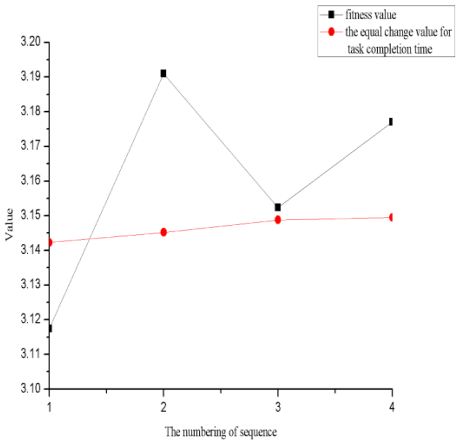

Figure 3. Comparison between the fitness value and the total task completion time

As seen from Table 2 and Figure 2, when the total task completion time is the same, the task scheduling sequence is different and the corresponding fitness function value is different. The greater the fitness value is, the smaller the difference between the total task completion time and the task completion time of each node, and the more balanced the time load gets. For instance, the fitness function value of Sequence 8 is the largest, so Sequence 8 is selected when the total task completion time is 334.52s in the system, and further optimization is realized.

When the total task completion time is different in the algorithm, the fitness function value, the corresponding task assignment sequence and the sum of difference between the total task completion time and the task completion time of each processing node (i.e. sum of difference) are shown in Table 3 below. The relation between the total task completion time and the fitness function values is shown in Figure 3.

Table 3. Table of related data

No.	Total task completion time T	Elements in the corresponding sequence										Fitness value	$(T/100)-2$	$\sum_{j=1}^{n}(T-T_j)$
1	334.23	3	4	2	1	1	3	4	2	1	2	3.1174	3.1423	66.58413
2	334.52	2	4	1	3	2	2	4	1	2	1	3.191	3.1452	41.03716
3	334.88	1	2	2	3	2	1	1	4	1	4	3.1523	3.1488	52.78522
4	334.95	1	4	2	3	2	2	4	2	1	1	3.177	3.1495	44.34329

It can be seen from Table 3 and Figure 3 that the total task completion time of number 1 is relatively short, but the sum of the difference is too large, leading to time load imbalance at the processing nodes. So, the sequence's comprehensive performance is poor. Compared with No. 1, the total task completion time of No. 2 is slightly longer, but the sum of the difference is relatively small, so the comprehensive performance is relatively high. The comprehensive performance of the sequence is determined by the difference between the total task completion time and the sum of the difference, so that the two factors reach a certain equilibrium, which corresponds with the value of the fitness function. Then, the fitness function value can be used as the criteria to select the optimal and the fittest task assignment sequence.

4.2. Analysis of total task completion time

The simulation is based on a task length of 0.1M-1M, 50-500 tasks of different sizes randomly generated and 10 processing nodes. The comparison in the total task completion time between the GA、PSO、PSO-GA algorithms and the algorithm in the paper is shown in Figure 4. For the PSO-GA algorithm, we refer to document [21].

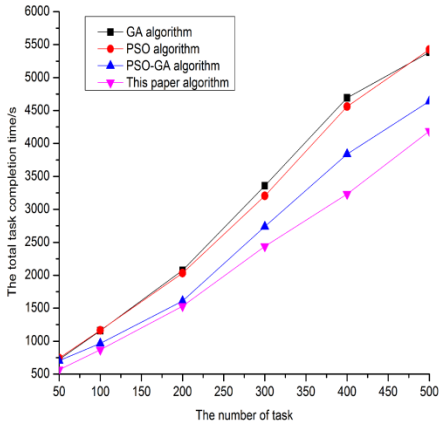

Figure 4. The comparison of the total task completion time

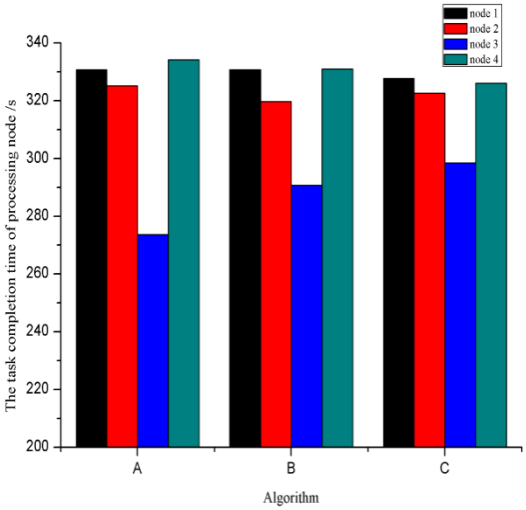

Figure 5. The comparison of time load balancing

It can be seen from Figure 4 that the total task completion time of this algorithm is less than that of the GA、PSO and PSO-GA algorithms. When the number of tasks is less than 200, the difference in total task completion time between the PSO-GA algorithm and our algorithm is not obvious; but, with an increase in number of tasks, the gap in total task completion time is relatively significant, i.e. when the task scale is too large, the advantage of this algorithm is more obvious.

4.3. Analysis of time load balancing

Within the same software and hardware simulation platform and the same experimental parameters, this simulation refers to the data in the genetic annealing algorithm from document [24]. There are 10 tasks and 4 processing nodes. This paper tested the genetic annealing algorithm, PSO-GA algorithm and our algorithm. The task completion times of each processing node for the three algorithms in this paper are (330.8, 325.21, 273.62, 334.23), (330.8, 319.74, 290.77, 330.98) and (327.8, 322.62, 298.56, 326.05), respectively. The corresponding comparison of the three algorithms on four processing nodes is shown in Figure 5.

In Figure 5, A, B and C stand for the genetic annealing algorithm, PSO-GA algorithm and the algorithm in this paper. The length of each bar represent the node's task completion time. From it, we can see that the longest bar of the algorithm (C) is the shortest when compared with the longest bar in the first two algorithms (A and B). That is, it takes the shortest time to complete the total task, and the task completion time of each processing task is relatively concentrated, i.e. time load is relatively balanced. Combined with Formula (4), we draw a conclusion that performance value of this algorithm in time load balancing is 97.25%, which has 2.71% and 1.15% improvement, respectively, when compared to the first two algorithms.

5. Conclusions

In this paper, we propose a particle genetic algorithm based on time load balancing. The algorithm preserves the optimal solution of the population and individual history, and the comprehensive performance of multi-objective optimization is higher than that of the standard PSO-GA algorithm. We propose the method of equalized initialization at particle processing nodes,

multi-objective optimization fitness function and particle position and velocity element standardization. It contributes a new idea for task scheduling. Combined with experimental simulation data, we prove that the multi-objective fitness function is feasible. Based on crossovers and mutations in the genetic algorithm, the crossover mutation function is established to avoid the local optimal phenomenon. At last, we get the task assignment sequence, which takes time load balancing at processing nodes into account and consumes far less time. This idea will be well-applied in the virtual scene, but can also be extended to route selection and fuzzy system control. In addition, in the future work we will combine this with deep learning, NSGA2 and other optimization algorithms.

Acknowledgements

This work was partly financially supported through grants from the National Natural Science Foundation of China (No.61503124), Research on 3D Digital Pavilion System Based on Unity3D (No.15A520016), and Research on Key Technology of Mobile Augmented Reality Application Development Platform Based on Clouds (No.172102210273). The authors thank the 3 anonymous reviewers for their helpful suggestions. Thank you very much to my teachers Hou Shouming and Chang Li for their guidance.

References

1. M. Alouane, and H. E. Bakkali, *"*Virtualization in Cloud Computing: Existing Solutions and New Approach,*" International Conference on Cloud Computing Technologies and Applications.* IEEE, pp. 116-123, 2017.
2. R. Buyya, C. S. Yeo, S. Venugopal, J. Brobery, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systemss*, vol. 25, no. 6, pp. 599-616, 2009.
3. K. Chadha, and A. Bajpai, "Security Aspects of Cloud Computing," *International Journal of Computer Applications*, vol. 40, no. 8, pp. 43-47, 2012.
4. Q. Cai, D. Shan, and W. Zhao, "Resource Scheduling in Cloud Computer based on Improved Particle Swarm Optimization Algorithm," *Journal of Liaoning Technical University*, 2016.
5. H. Chang, and X. Tang, "A Load-balance based Resource-scheduling Algorithm under Cloud Computing Environment," *International Conference on Web-Based Learning. Springer, Berlin, Heidelberg*, pp. 85-90, 2010.
6. Y. U. Guo-Long, Z. W. Cui, and Y. Zuo, "Cloud Platform Scheduling Method based on Optimized Particle Swarm Optimization Algorithm," *Journal of Inner Mongolia Normal University*, 2016.
7. H. Grillo, D. Peidro, M. M. E. Alemany, and J. Mula, "Application of Particle Swarm Optimisation with Backward Calculation to Solve a Fuzzy Multi-objective Supply Chain Master Planning Model," *International Journal of Bio-Inspired Computation*, vol. 7, no. 3, pp. 157-169, 2015.
8. G. Jung, and K. M. Sim, "Agent-based Adaptive Resource Allocation on the Cloud Computing Environment," *International Conference on Parallel Processing Workshops* IEEE, pp. 345-351, 2011.
9. N. J. Kansal, and I. Chana, "Cloud Load Balancing Techniques: A StepTowards Green Computing," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 1, pp. 238-246, 2012.
10. G. Kumughato, and J. Priya, "A Survey of Load Balancing Techniques in Cloud Environment," *International Journal of Advanced Research in Computer Science*, vol. 5, no. 1, 2014.
11. X. L. Li, "Research on Cloud Computing Task Scheduling based on Simulated Annealing Genetic Algorithm," *Huazhong Normal University*, 2016.
12. N. D. Lockin, and Acknowledgments, "Journal of Cloud Computing," *Communications of the ACM*, vol. 4, no. 2, pp. 50-58, 2013.
13. H. Mehta, P. Kanungo, and M. Chandwani, "Decentralized Content Aware Load Balancing Algorithm for Distributed Computing Environments," *Proceedings of the International Conference & Workshop on Emerging Trends in Technology. ACM*, pp. 370-375, 2011.
14. A. M. Nakai, E. Madeira, and L. E. Buzato, "Load Balancing for Internet Distributed Services Using Limited Redirection Rates," *Dependable Computing (LADC), 2011 5th Latin-American Symposium on.* IEEE, pp. 156-165, 2011.
15. P. Rajanna, and J. Gyani, "A Comparative Study of Cloud and Grid Computing Security Solutions," *International Journal of Computer Science and Electronics Engineering*, vol. 2, no. 1, pp. 1-8, 2012.
16. M. Randles, D. Lamb, and A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing," *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on.* IEEE, pp. 551-556, 2010.
17. M. A. Sharkh, A. Ouda, and A. Shami, "A Resource Scheduling Model for Cloud Computing Data Centers," *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International.* IEEE, pp. 213-218, 2013.
18. S. Selvarani, and G. S. Sadhasivam, "Improved Cost-based Algorithm for Task Scheduling in Cloud Computing," *IEEE International Conference on Computational Intelligence and Computing Research.* IEEE, pp. 1-5, 2011.
19. Y. Wang, Y. Sun, and Y. Sun, "Task Scheduling Algorithm in Cloud Computing based on Fairness Load Balance and Minimum Completion Time," *Advances in Engineering Research*, 2016.
20. S. C. Wang, K. Q. Yan, W. P. Liao, and S. S. Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network," *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on.* IEEE, pp. 1: 108-113, 2010.

21. X. H. Wang, D. S. Zou, "An Cloud Computing Task Scheduling Method based on Particle Swarm Optimization Genetic Algorithm (PSO-GA)," *World Science and Technology Research and Development*, no. 02, pp. 110-114, 2014.
22. J. Xu, and Y. Tang, "Research of Improved Particle Swarm Optimization based on Genetic Algorithm for Hadoop Task Scheduling Problem," vol. 2, pp. 60-66, 2015.
23. Y. Zhao, and W. Huang, "Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud," *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on.* IEEE, pp. 170-175, 2009.
24. J. Zhu, and D. Xiao, "Multi-dimensional QoS Constrained Task Scheduling Mechanism for Load Balancing Under Cloud Computing," *Computer Engineering and Applications*, vol. 49, no. 9, pp. 85-89, 2013.
25. Z. Zhang, and X. Zhang, "A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation," *Industrial Mechatronics and Automation (ICIMA), 2010 2nd International Conference on.* IEEE, vol. 2, pp. 240-243, 2010.

Yuzhen Zhang is a Master's student from the School of Computer Science and Technology, Henan Polytechnic University of Science and Technology. Her research interests include digital simulation of engineering and virtual reality technology.

Shouming Hou graduated from the School of Computer Science and Technology of Northeastern University. Now, he is a professor at the Computer Science and Technology, Henan Polytechnic University, a teacher at the Henan Provincial Department of Education Academic Technology Leaders, Henan Province Outstanding Teachers, National Manufacturing Informatization 3D CAD Certification Training Instructor, and VRP Virtual Reality Certification Training Instructor. His current research interests include digital simulation of engineering, virtual reality technology and intelligent information processing.

Li Chang is a Master's student from the School of Computer Science and Technology, Henan Polytechnic University of Science and Technology. Her research interests include digital simulation of engineering and virtual reality technology.