Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 13, no. 7, November 2017, pp. 1063-1069
DOI: 10.23940/ijpe.17.07.p8.10631069

Research on Cloud Computing Task Scheduling based on Improved Particle Swarm Optimization

Shasha Zhao, Xueliang Fu*, Honghui Li, Gaifang Dong, Jianrong Li

College of Computer and Information Engineering, Inner Mongolia Agricultural University, Hohhot, 010018, China

Abstract

Particle swarm optimization (PSO) is a popular intelligent algorithm to solve the task scheduling optimization problem of work-flow system in cloud computing environment. However, this algorithm is easy to fall into the local optimality. It is the reason that the execution time and cost of the scheduling scheme are higher than other methods. Therefore, by improving the calculation method of the single particle success value, the traditional adaptive inertia weight particle group task scheduling algorithm is optimized. Through each particle fitness and local optimal value and global optimal value that divided into four cases to compare, the inertia weight improved can be used to adjust the particle velocity more accurately. It can better equilibrate search capacity of particles between global and local, and avoid the local maximum of the particles. In this paper, we more accurately describe the particle state and improve the inertia weight. We can get a scheduling scheme with lower execution time and lower cost. The analog results show that the improved algorithm is stable. The convergence accuracy is obviously improved. It can effectively avoid prematurely falling into the local optimality.

Keywords: particle swarm; cloud computing; adaptivity; inertia weight

(Submitted on July 25, 2017; Revised on August 30, 2017; Accepted on September 15, 2017)

(This paper was presented at the Third International Symposium on System and Software Reliability.)

1. Introduction

At present, the most commonly used task scheduling algorithm in cloud work-flow management system is proposed by Eberhart and Shi [1]. It uses a random optimization heuristic algorithm established by group intelligence [1,2]. PSO algorithm has the characteristics of few parameters, easy implement and fast convergence. However, this algorithm can't effectively control the velocity of the particle. It cannot equilibrate search capability between the global and local. The local convergence defect cannot find the optimal solution in the limited iteration times.

For solving the shortcoming of local convergence, the velocity updating formula of particle swarm algorithm increases the inertia weight parameter. In the process of algorithm implementation [1,2,3], the inertia weight is fixed as a constant. It makes the algorithm toilless to get into the local optimality. Therefore, Eberhart and Shi proposed the random inertia weight to optimize the dynamic change of the target [3]. Moreover, Eberhart and Shi proposed a linear reduction of inertia weight to improve the ability of particle swarm algorithm aiming to solve the problem of accuracy of the algorithm [4,5]. If the initial particle search is not a good location, it is toilless to run into the local optimality. On the basis of linearly reducing the inertia weight, Chatterjee and Siarry proposed that the exponential reduce the inertia weight to a certain extent. It can avoid the algorithm getting into local optimum [6]. However, the determination of the index value is a difficult problem, which leads to a great deal of calculation. It is not suitable for complex problem solving.

* Corresponding author.
E-mail address: fuxl@imau.edu.cn.

In summary, against the non-adaptive inertia weight, Nickabadi and other people proposed a particle swarm algorithm based on adaptive inertia weight [7]. According to the position state information of the whole particle group in the process of algorithm iteration, the algorithm can modify the inertia weight dynamically. However, the method of calculating the success value in this parameter only considers the merits of the position of the individual particles in different iterations. The adaptive mechanism cannot adjust the inertia weight accurately. It makes the PSO algorithm not precise enough to fall into the local optimal. Therefore, by improving the successful value calculation method in the traditional adaptive inertia weight, the particle swarm optimization algorithm based on the improved adaptive inertia weight is improved [7]. In summary, the simulation results show that the improved adaptive inertial weight particle group task scheduling algorithm can effectively avoid prematurely local optimality, and be better than other algorithms in the fitness value.

2. Cloud computing task scheduling problem description

2.1. Standard PSO algorithm

In 1995, Kennedy proposed PSO algorithm, based on the individual's search space fitness value of the size of the individual to evaluate the merits and defects [8]. He supposed that in a *D*-dimensional search space, there is a particle population with a population size of *N*, the position of each particle *i* (*i* = 1,2, ..., *N*) in space can be expressed as $\mathbf{x_i} = (\mathbf{x_{i1}}, \mathbf{x_{i2}}, ..., \mathbf{x_{iD}})$. The velocity of particle *i* is expressed by $\mathbf{v_i} = (\mathbf{v_{i1}}, \mathbf{v_{i2}}, ..., \mathbf{v_{iD}})$ [8,9]. When the particle *i* is updated to *t* generation, *d*-th (*d* = 1,2, ..., *D*) dimensional velocity in the subspace is shown in the following equation:

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 rand_1(p_{id}(t) - x_{id}(t)) + c_2 rand_2(p_{gd}(t) - x_{id}(t)) \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

The p_{id} *(t)* represents historical optimal position of the current iteration particle *i* [8]. The p_{gd} *(t)* represents the global historical optimal position of the current iteration population; ω is the inertia weight. The larger ω value favors the global search, and the smaller ω is favorable for local development; *c1*, *c2* is the acceleration factor, usually between 0 ~ 2 [8]; $rand_1$, $rand_2$ is the 0 to 1 uniform distribution of two independent random number. The PSO algorithm is applied to cloud computing task scheduling problem. It needs to be improved and the particle is encoded.

2.2. Particle coding and initialization

There is a coding method. It is decimal coding method with *N* tasks, *M* resources, and *N*> *M*. When *N* = 9, *M* = 3, the coding sequence (3,1,2,3,1,1,2,3,2) is a particle, where apiece task maps to a resource, for example, *Task2* corresponds to *Resource2*. Then, the particles are decoded with the goal of obtaining all the tasks running on each resource, such as all tasks running on *Resource2* as {2,6,8}. Define the matrix **time[i, j]** to record the execution time of *Task i* on *Resource j*, and the total time for all tasks on *Resource j* is:

$$Time(j) = \sum_{i=1}^{n} time[i, j] \qquad (1 \le j \le M) \quad (3)$$

The *i* represents the number of tasks executed on *Resource j*; *n* represents the total number of tasks performed on *Resource j*. The total time for the system to complete all tasks is:

$$TaskTime = \max(Time(j)) \quad (1 \le j \le M) \quad (4)$$

Then, the population is initialized, with *T* tasks, *R* resources, *S* particles, and *T*>*R*. The *i*-th particle position is represented by the vector $\mathbf{x_i}$, defined as $\mathbf{x_i} = \{\mathbf{x_{i1}}, \mathbf{x_{i2}}, ..., \mathbf{x_{id}}\}$ [9]. $1 \leqslant i \leqslant S$, $1 \leqslant d \leqslant T$, $1 \leqslant \mathbf{x_{ij}} \leqslant R$. The *i*-th particle speed is represented by the vector $\mathbf{v_i}$, defined as $\mathbf{v_i} = (\mathbf{v_{i1}}, \mathbf{v_{i2}}, ..., \mathbf{v_{id}})$. $1 \leqslant i \leqslant S$, $1 \leqslant d \leqslant T$, $-R \leqslant \mathbf{v_{ij}} \leqslant R$. Initially, *S* particles are generated. The position and the speed of each particle are assigned random integers between [1, *R*] and [-*R*, *R*], respectively [9].

2.3. Fitness function

The fitness is a comprehensive index that evaluates the execution time and cost of the task scheduling scheme [10]. It shows the advantages and disadvantages of the particle position state in the particle swarm algorithm.

$$\text{Fitness} = \frac{1}{\text{TaskTime} + \text{Debt}} \tag{5}$$

$$\text{Debt} = \text{VMdebt} + \text{Timecost} + \text{MIPScost} \tag{6}$$

The *Debt* represents the cost of the scheduling scheme, as shown in equation (6). *Debt* consists of three parts. *VMdebt* is the cost of the virtual machine service, and if the number of virtual machines increases, *VMdebt* also increases; *Timecost* is the execution time ratio. The reciprocal of the execution time of the scheme is expanded according to the number of different tasks. It can be seen that the shorter the execution time, the greater the ratio, the greater the *Timecost*. *MIPScost* is the average processing power ratio, that all the virtual machine processing capacity to take the average. Similarly, according to the number of different tasks reduce the corresponding multiple, so that it has executable meaning. When the processing capacity is stronger, *MIPScost is* also greater. To sum up, the fitness can measure the time and cost of the scheduling plan. If the fitness of the scheduling program is higher, the time required and the cost is smaller. The location of the particle position is better. On the contrary, if the overhead is greater, the particle position is worse.

3. Adaptive inertia weight

3.1. Traditional adaptive inertia weight

Nickabadi and other people proposed an adaptive inertia weight AIW [7]. It first calculates the success of a single particle (as shown in equation (7)). Then, calculates the success rate of the entire particle population (as shown in equation (8)). Finally, updates the inertia weight. *S(i,t)* is the success value of the particle *i* in the *t*-th iteration [9]; *pbest* is the optimal position of the particle *i* at the *t*-th iteration [9]; *fitness(pbest)* is the fitness of the optimal position of the particle *i* at the *t*-th iteration. When the fitness of the current particle is less than the last optimal position, the success value of the particle in this iteration is 1. It indicates that the position of the particle in the search is better than the previous time. It shows the search is successful. On the other hand, the current position of the particle is equal to the last optimal position. In the next two iterations, the optimal position of the particle fitness has not changed: the success value is set to 0. When the position at which the particle moved in this search is worse than the last optimal position, the search fails.

$$S(i, t) = \begin{cases} 1, \text{fitness}(\text{pb est}_i^t) < \text{fitness}(\text{pb est}_i^{t-1}); \\ 0, \text{fitness}(\text{pb est}_i^t) = \text{fitness}(\text{pb est}_i^{t-1}). \end{cases} \tag{7}$$

$$P_s(t) = \sum_{i=1}^{n} S(i, t)/n. \tag{8}$$

$$\omega(t) = (\omega_{max} - \omega_{min})P_s(t) + \omega_{min}, 0 \le \omega_{min} \le \omega_{max} \le 1, \tag{9}$$

P(t) is the success rate of the particle swarm at *t* iterations. It indicates that the proportion of particles in the group. And the iterations of the particles are better than before. $\sum S(i,t)$ is the sum of the success values of all particles; *n* is the number of particles in the entire particle population. $\omega(t)$ is the inertia weight at the *t*-th iteration. It is used to adjust the initial velocity of the particles at the iteration.

AIWPSO algorithm generated scheduling program execution time and cost higher than the ordinary PSO algorithm, because the inertia weight AIW in the success of the calculation method only considers two cases. The current particle fitness is better than the last. The successful value is 1 otherwise it is 0. The method does not fully analyze the position information of the particles. The global optimal position is not considered, so the calculation of the success rate and inertia weight is not accurate enough. Therefore, the method can not accurately adjust the inertia weight value in each iteration. It cannot effectively balance the global and local search of the algorithm in the iterative process, so the algorithm is localized and cannot find the optimal solution.

3.2. Improved adaptive inertia weight

In order to describe the position of the particle in a more accurate way, the fitness value of each particle is compared with the global optimal value in the successful value calculation method. In this paper, the fitness value of each particle is compared with the global optimal value. The success value of the improved adaptive inertia weight is calculated as follows:

$$S(i,t)=\begin{cases}1, \text{fitness}(\text{pb est}_i^t) > \text{fitness}(\text{pb est}_i^{t-1}) \wedge \text{fitness}(\text{pb est}_i^t) > \text{fitness}(\text{gb est}^t);\\ 0.7, \text{fitness}(\text{pb est}_i^t) > \text{fitness}(\text{pb est}_i^{t-1}) \wedge \text{fitness}(\text{pb est}_i^t) < \text{fitness}(\text{gb est}^t);\\ 0.3, \text{fitness}(\text{pb est}_i^t) > \text{fitness}(\text{pb est}_i^{t-1}) \wedge \text{fitness}(\text{pb est}_i^t) = \text{fitness}(\text{gb est}^t);\\ 0, \text{fitness}(\text{pb est}_i^t) = \text{fitness}(\text{pb est}_i^{t-1});\end{cases} \tag{10}$$

In the formula (7), the current fitness of each particle is compared with the previous iteration. The global optimal value *fitness(gbest^t)* is added to *S(i,t)* in eq. (10). The success value of the new *S(i,t)* calculation method is expanded to four cases: 1,0.7,0.3,0. The calculation method is based on a more detailed evaluation of the particle position state. When the search is locally optimized, the result of the comparison with the global optimal fitness produces three kinds of result values, so as to obtain *S(i,t)* more accurately. More accurate *S(i,t)* can improve the accuracy of the particle group's success rate (eq. (8)) and the adaptation of the inertia weight. Therefore, the algorithm can adjust the particle velocity more accurately to balance the search capability of the particles between global and local. It can avoid the local optimum.

The improved adaptive inertia weight changes with the different search states of the particle swarm. It can dynamically change the search speed of the particle. When the success rate $P_s(t)$ of the particle swarm is large, it shows that the whole particle group is far from the optimal value. At this time, the particle is required to search globally at a large initial velocity. By increasing the value of $\omega(t)$, the particle is close to the optimal value at a larger initial velocity. When the success rate $P_s(t)$ of the particle swarm is small, it shows that the whole particle swarm is close to the optimal value. By reducing the value of $\omega(t)$, the particle search speed can be reduced. The optimal solution can be refined to ensure the accuracy of the algorithm. In this paper, the algorithm can improve the search capability of the dynamic balance algorithm according to the current position state of the particle. It changes the velocity of the particle by the improved success value formula.

3.3. Particle Swarm Task Scheduling Algorithm Based on Improved Adaptive Inertia Weight

Algorithm The process of the optimal fitness

Input: the maximum number of iterations *ITERATIONS*, the number of tasks *N*, the number of virtual machines *M*;
Output: optimal scheme, optimal fitness.The basic implementation of the improved algorithm is as follows:

1. **For** *i* = 1 **to** *n* **do**;
2. Initialize the particle's initial position $\mathbf{x}_i$ and the initial velocity $\mathbf{v}_i$;
3. **End for**
4. **For** *i* = 1 **to** *n* **do**
5. Calculate fitness;
6. **End for**
7. The optimal global optimal scheduling scheme is found from n scheduling schemes.
8. **For** *i* = 1 **to** *ITERATIONS* **do**
9. Update the scheduling scheme according to the update search speed, assign the task to the corresponding virtual machine;
10. **For** *j* = 1 **to** *n* **do**
11. Update fitness
12. **End for**
13. The optimal global optimal scheduling scheme is found from *n* scheduling schemes.
14. **For** *j* = 1 **to** *n* **do**
15. Calculate the success value of the current scheduling scheme *S*;
16. *Sum* = *Sum* + *S*;
17. **End for**
18. Calculate the success rate;Update the inertia weight according to the success rate;
19. **End for**
20. **Return** best fitness;
21. **End**

The scheduling algorithm can dynamically balance the global search and local search of the particles by updating the inertia weight dynamically. It can avoid the algorithm getting into the local optimal, so as to obtain the optimal scheduling scheme.

4. Simulation experiment and analysis

4.1. Simulation of experimental environment and algorithm parameters

By using Cloudsim to build the cloud computing environment and eclipse as the development platform, there are five algorithms: the improved adaptive inertial weight particle swarm optimization algorithm, the standard PSO algorithm, the traditional adaptive inertial weight particle swarm algorithm, the NAIWPS algorithm proposed in [9], and Sequential scheduling algorithm. The convergence and optimal fitness of each algorithm are compared and analyzed respectively. The algorithm parameters are shown in Table 1.

Table 1. Improved Adaptive Inertia Weight Particle Swarm Algorithm Parameters

parameter name	Parameter symbol	Parameter value
Population size	*s*	30
Number of resources (number of virtual machines)	*r*	3
Number of tasks	*t*	50~300
Maximum inertia weight	ω_{max}	1
Minimum inertia weight	ω_{min}	0
Learning factor	*c1*、*c2*	2
Maximum number of iterations	*Iteration*	10~100

4.2. Results and Analysis

The total number of tasks is 50 to 300, and the load of a single task is a random value in the range of 30 to 3000 [11]. Each algorithm runs 100 times and takes an average value. The results are shown in the following figures:

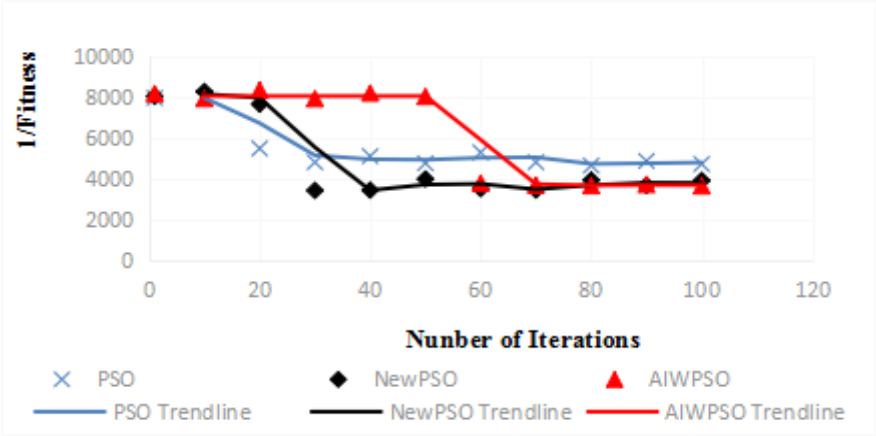

Figure 1. When the number of tasks is 50, the algorithm converges

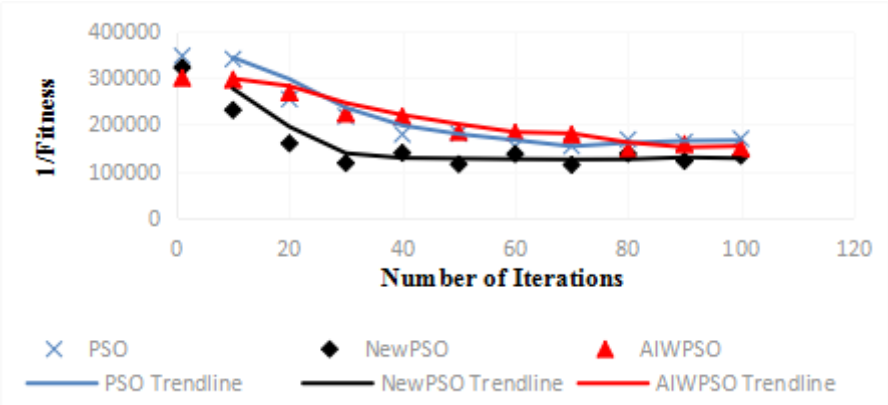

Figure 2. When the number of tasks is 300, the algorithm converges

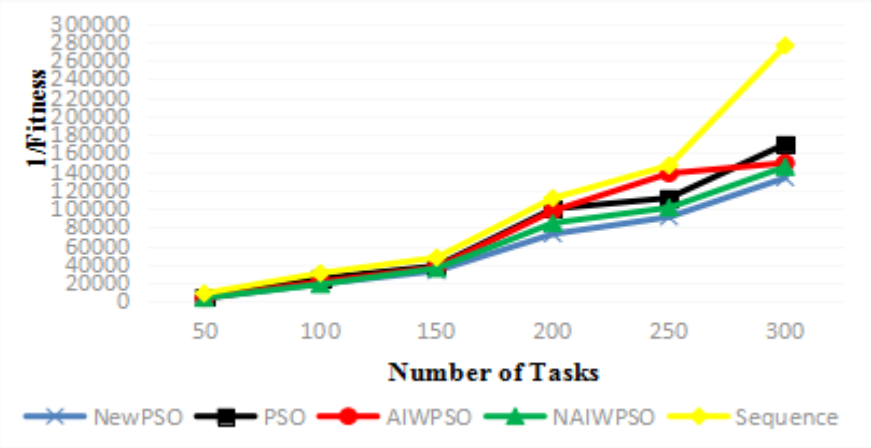

Figure 3. Five kinds of algorithm fitness

Figure 1 and Figure 2 show that the convergence of the NewPSO algorithm is stable. The convergence accuracy is higher than that of the PSO scheduling algorithm and AIWPSO scheduling algorithm. Figure 3 shows the optimal fitness values of the five algorithms from 50 to 300. It can be clearly seen that the NewPSO scheduling algorithm is far superior to the Sequence scheduling algorithm. It is also superior to the NAIWPSO scheduling algorithm mentioned in the literature [7]. Therefore, it can be seen from the simulation experiment data that with the increase of the scale of the problem, the number of tasks increases. Improved adaptive inertia weight PSO algorithm is better to reflect the adaptability of the improved adaptive inertia weight and better to balance the global and local search.

5. Conclusions

The simulation results show that the improved adaptive inertia weight PSO scheduling algorithm can describe the position of the particle more accurately by simplifying the calculation method of the success value. It avoids the local optimum solution. It can more efficiently obtain better scheduling. However, the work of this paper is in the stage of theoretical analysis and simulation. It needs to be validated on the real cloud computing platform. The fitness function of the algorithm only considers the total value of execution time and cost. Therefore, it is necessary to further study.

Acknowledgements

This research was financially supported by Chinese Natural Science Foundations (61363016, 61063004), Key Project of Inner Mongolia Advanced Science Research (NJZZ14100), Inner Mongolia Colleges and Universities Education Department Science Research (NJZC059), Natural Science Foundation of Inner Mongolia Autonomous Region of China (NO.2015MS0605, NO.2015MS0626 NO.2015MS0627, and NO.2017MS0605), and Inner Mongolia Autonomous Region Science and Technology Project (through the precipitation of GSM network on-line monitoring and data transmission system development), and Ministry of Education Scientific research foundation for Study abroad personal[2014] 1685.

References

1. A. Azimifar, S. Payan, “Enhancement of Heat Transfer of Confined Enclosures with Free Convection Using Blocks with PSO Algorithm,” Applied Thermal Engineering, 2015
2. I. Boulkaibet, L. Mthembu, F. De Lima Neto, T. Marwala, “Finite Element Model Updating Using Fish School Search and Volitive Particle Swarm Optimization,” Integrated Computer-Aided Engineering, 2015
3. H. Fard, R. Prodan, T. Fahringer, “A Truthful Dynamic Workflow Scheduling for Commercial Multicloud Enmronments,” IEEE Trans on Parallel and Distrihnted Systems, pp. 1203-1212, 2013
4. Zhiqiang Gao, Lixia Liu, Weiwei Kong, Xiaohong Wang, “A Composite Framework of Cuckoo Search and PSO Algorithm,” Applied Mechanics and Materials, 2015
5. Jianan Lu, Yonghua Chen, “Particle Swarm Optimization (PSO) Based Topology Optimization of Part Design with Fuzzy Parameter Tuning,” Computer-Aided Design and Applications, 2014
6. Xuejun Li, Jia Xu, Erzhou Zhu, Yewen Zhang, “New Adaptive Inertia Weight Calculation Method in Task Scheduling Algorithm ,” Journal of Computer Research and Development, pp. 1990-1999, 2016
7. A. Nickabadi, M. Ebadzadeh, R. Safabakhsh, “A Novel Particle Swarm Optimization Algorithm with Active Inert Weight,” Applied Soft Computing, pp. 3658-3670, 2011
8. N. Netjinda, B. Sirinaovakul, T. Achalakul, “Cost Optimization in IaaS for Dependent Workload with Particle Swarm Optimization,” The Journal of Supercomputing, pp. 1579-1603, 2014

9. D.B. Prakash, C. Lakshminarayana, "Multiple DG Placements in Distribution System for Power Loss Reduction Using PSO Algorithm," Procedia Technology, 2016
10. Marco Simone, Alessandro Fanti, Giuseppe Mazzarella, "Ridge Waveguide Optimization with PSO Algorithm," Journal of Electromagnetic Waves and Applications, 2015
11. Richa Yadav, Maneesha Gupta, "New Improved Fractional Order Integrators Using PSO Optimisation," International Journal of Electronics, 2015

Shasha Zhao is a graduate student, graduated from the Inner Mongolia Agricultural University in 2015 and received her bachelor's degree in information management. Her research interest is the task scheduling problem based on cloud computing platform.

Xueliang Fu is a professor in algorithms of graph theory at Inner Mongolia Agricultural University in China. He received th e MS and PhD in software engineering from DALIAN University of Technology in 2005 and 2008 in China. He received hi s BS in computer science in 1992. His research interests include Domination Number of Graphs , Intelligent Computation and Data Mining. He has published about 50 papers in international journals, conference proceedin gs. Now Fu Xueliang is the Vice Dean of the College of computer and information engineering of Inner Mongolia Agricultu ral University.

Honghui Li received the BS, MS and PhD degrees all in computer science from Electronic technology University,China, In ner Mongolia University, China and Concordia University, Canada in 1992, 2002, and 2012 respectively. She is currently a professor in Computer science & technology with School of Computer and Information Engineering, Inner Mongolia Agric ultural University, China. Her research interests include network optimization and Network planning algorithms. She has pu blished about 25 papers in international journals and conference proceedings.

Gaifang Dong received the BS and MS degrees both in computer science from Inner Mongolia Normal University, China and GuiZhou University, China, in 2002 and 2005 respectively. She is currently a Ph.D. student and an Associate Professor at the College of Computer and Information Engineering in the Inner Mongolia Agricultural University. Her research interests include computational intelligence, bioinformatics computing and parallel computing. She has published about 15 papers in international journals and conference proceedings.

Jianrong Li received the BS degree in electric automation in 1999 and the MS degree in control theory and control engineering in 2004 that both from Inner Mongolia Polytechnic University. She is currently a Ph.D. student and an Associate Professor at the College of Computer and Information Engineering in the Inner Mongolia Agricultural University. Her research interest is task scheduling algorithm water resources management for cloud computing.