International Journal of Grid and Distributed Computing
Vol. 9, No. 11 (2016), pp.309-322
http://dx.doi.org/10.14257/ijgdc.2016.9.11.26

A Novel Approach to Task Scheduling using The PSO Algorithm based Probability Model in Cloud Computing

Li Ruizhi[1], Gao Jue[3], Gao Honghao[3], Bian Minjie[1] and Xu Huahu[2]

[1]*School of Computer Engineering and Science, Shanghai University, 200444, Shanghai, China*
[2]*Shanghai Shang Da Hai Run Information System Co., Ltd*
[3]*Computing Center, Shanghai University, 200444, Shanghai, China*
liruizhi@shu.edu.cn

Abstract

With the development of cloud computing technology, people not only want to pursue the shortest time to complete the tasks by using cloud computing, but also hope to take into the running costs of machines. Existing task scheduling algorithm in the cloud computing environment has been unable to meet people's needs. As an extension and generalization of the model checking theory, probability model checking is also used in many fields, such as random distributed algorithm and other areas. The task scheduling algorithm based on the particle swarm optimization algorithm combined with probability model is proposed in this paper. The algorithm defines the fitness functions of the time cost and the running cost. The fitness functions can improve the efficiency of the cloud computing platform. At the same time, the probability model can be used to analyze the running states of machines and the computing capability of the nodes in the cloud cluster. The probability, which is calculated by the probability model, provides the basis for changing particle swarm algorithm's the inertia factor and the learning factor, so as to solve the drawback that the inertia factor and the learning factor solely depend on the fixed value.

Keywords: *Particle Swarm Optimization Algorithm, Probability Model, Inertia Factor, Learning Factor, Auto-correcting*

1. Introduction

1.1. The Present and Problems of Task Scheduling

Cloud computing is a new product of the traditional computing and the grid computing. Cloud computing has the inherent advantages that other computing models do not have. In the environment of cloud computing, users can follow their own needs to use the services and infrastructure from the cloud provider. For some large tasks, the user can submit a task request and the cloud computing task scheduling center can dynamically allocate resources according to the user's request. With the development of cloud computing, people not only want to pursue the fastest completion time of tasks in cloud computing environment, but also want to take into account the running cost of machines.

The goal of task scheduling in cloud computing environment is to schedule tasks to the required resources, and try to make the task execution time as short as possible, meanwhile making full use of resources. However, the literature [1] shows that the task scheduling problem is a NP problem. So how to find the optimal solution to the scheduling problem has become more and more important.

Through reviewing the literatures, many scholars propose many methods, such as using heuristic algorithms and comparing task scheduling algorithms with other disciplines.

ISSN: 2005-4262 IJGDC

Literature [2] is based on the greedy strategy; it proposes the minimum execution time algorithm and the Min-Min algorithm to solve the load imbalance of the cloud system.

The genetic algorithm is improved by Literature [3]. In this paper, a scheduling problem based on grid technology is proposed. This algorithm can improve the availability of the algorithm by using the fitness functions of genetic algorithm.

In literature [4], task scheduling is based on the sense of perception. This algorithm is not only concerned with the inter dependence of scheduling tasks, but also respects the resources that they are using, such as memory and bandwidth and so on.

Literature [5], which is according to the resources of the heterogeneous cloud with different abilities and functions, proposes three task scheduling algorithms that are named by MCC, MEMAX and CMMN. This paper aims to the minimize completion time and the maximize resources in cloud computing environment.

In summary, the task scheduling algorithm is more and more important to the cloud computing platform. A good task scheduling algorithm not only can meet the needs of users, but also can reduce the energy consumption of cloud computing platform. There are many mature algorithms which are suitable for the cloud computing platform, but different algorithms maybe require different environments. Task scheduling algorithm, which is proposed in this paper is mainly, has been applied on the cloud rendering platform. The cloud rendering platform is kind of application in cloud computing environment.

1.2. The Present and Problems of Particle Swarm Optimization Algorithm

Particle swarm optimization algorithm (PSO algorithm) is proposed by Dr. Eberhart and Dr. Kennedy in 1995. The basic idea of PSO algorithm is originated from the process that the birds look for foods [7]. The particle swarm optimization algorithm is a kind of optimization algorithm, which is based on bionics, to achieve the purpose of seeking the optimal solution by cooperating and sharing information between the individual. Compared with other optimization algorithms, particle swarm optimization algorithm has the advantages of easy to implement and fast convergence. But the particle swarm optimization algorithm also has the defects of weak exploitation capacity, and it is easy to fall into the local optimal solution.

After reviewing of the literatures, there are many scholars have proposed a lot of optimization strategies for the particle swarm optimization algorithm:

In literature [8], an adjustment that the inertia factor decreases by linear is presented. The method is more efficient than the method that the inertia factor only uses the fixed value, but it has limitations in dealing with the complicated problems.

A fuzzy rule algorithm with the inertia factor is proposed in literature [9]. This algorithm can solve more complex problems, but need to build the fuzzy rules. The realization process of this algorithm is complex with huge amount of calculation, so the algorithm's operational efficiency is very low.

Literature [10] proposes a random inertia factor algorithm, which is more suitable for solving dynamic system, relatives to the literature [8-9].

In summary, the particle swarm optimization algorithm has been studied by many scholars, and a series of improvement measures have been put forward. Improved particle swarm optimization algorithm in this paper not only considers the changing trend of the inertia factor and the learning factor, but also uses probability model for statistical analysis, which contains the running condition and the computing capability. This can provide the personalized inertial factor and learning factor for different computing nodes.

2. The Basic Idea of PSO Algorithm

Particle swarm optimization algorithm is derived from the complex adaptive system (CAS). CAS theory was formally proposed in 1994, and the members of CAS are called the individual. For example, when we study the bird system, each bird in this system is called the individual. The individual has adaptability, it can communicate with the environment and other individuals, and it can change their thinking and behavior according to the "learning" and "accumulate experience" in the process of communication. The evolution of the whole system including: produce a new level (the birds are born); appear differentiation and diversity (birds are divided into many small group); the appearance of new themes (birds find new foods in the process of looking for foods). So the individual of the CAS system has four basic characteristics which are the basis for the development of particle swarm optimization algorithm:

① The individual is active, independent and positive;

② The individual, the environment and other individuals interact with each other. This effect is the primary impetus to promote the whole system to develop and change;

③ The entire system needs to be combined with the influence of environment and the influence of the individual;

④the entire system may be affected by some random events.

The particle swarm optimization algorithm is based on the CAS system. Imagine a scenario: a group of birds randomly search food in a space, but there is only one piece food in this space. All the birds do not know where the food is, but they know how far from the current position to the food. So what is the best strategy to find the food? The most simple and effective method is that find the lucky bird, which is nearest to the food, and search its surrounding area.

The particle swarm optimization algorithm, which is inspired by the biological behavior, is used to solve the optimization problem. In the particle swarm algorithm, each potential solution of optimization problems can be imagined as a point in d-dimensional space. This point is known as particles. All particles have a value which is determined by the fitness functions, and each particle has the velocity which decides the direction and distance, then other particles will follow the lucky particle to continue searching for the optimal solution.

Suppose that there are N particles in the M dimensional space, so the dimension of each particle is also M dimension. The position of each particle can be expressed as the equation:

$$x_i = \left(x_{i_1}, x_{i_2}, \ldots, x_{i_{M-1}}, x_{i_M} \right) \tag{1}$$

The velocity of each particle can be expressed as the equation:

$$v_i = \left(v_{i_1}, v_{i_2}, \ldots, v_{i_{M-1}}, v_{i_M} \right) \tag{2}$$

Firstly, each particle should consider two special points:

①The historical optimal value (P_i), which is searched by the particle itself, can be expressed as the equation:

$$p_i = \left(p_{i_1}, p_{i_2}, \ldots, p_{i_{M-1}}, p_{i_M} \right), i = 1, 2, 3, ..., M \tag{3}$$

② The global optimal value (G_{best}), which is searched by all particles, can be expressed as the equation (there is only one global optimal value):

$$G_{best} = \left(G_1, G_2, \ldots, G_{M-1}, G_M \right) \tag{4}$$

Then, updating equations of particles' the positions and the velocities in the particle swarm optimization algorithm are given below:

$$v_i^{(k+1)} = \omega * v_i^k + c_1 * \xi * (p_i^k - x_i^k) + c_2 * \eta * (G_{best} - x_i^k) \tag{5}$$

$$x_i^{(k+1)} = x_i^k + \sigma * v_i^{(k+1)} \tag{6}$$

According to equation (5) and (6), the symbol ω represents the inertia factor, which means the coefficient of keeping the original velocity. The symbol c_1 represents the weight coefficient, which means the probability of tracking particle's own historical optimal value, and the symbol c_2 represents the weight coefficient of following the global optimal value, so both c_1 and c_2 are called the learning factors. The value range of the symbol ξ and η is from 0 to 1. When the particle's position is updating, the symbol σ, which is known as the constraint factor, can limit the velocity. The symbol σ, ξ and η often takes a fixed value of 1.

2.1. Inertial Factor

The inertia factor, which indicates the inheritance of the particle's own original velocity, is an important parameter in the particle swarm optimization algorithm. At present, many scholars focus on improving the particle swarm algorithm by changing the inertia factor. From many literatures, the main way is that the inertia factor decreases with the number of iterations. At the same time, some scholars propose that the inertia factor can be dynamically adjusted according to the change of the particle's velocity. For example, the parameters of particle swarm optimization are defined by literature [11], and the problem of exploring the optimal solution is transformed into another problem that how to optimize parameters, such as the inertial factor and the learning factor.

Through literature [7] can be seen: if the inertia factor gets bigger, it is helpful to jump out of the particle's own historical optimal value. In this case, it will be easy to search the global scope. The smaller inertia factor is helpful to accurate search the area where the particle is searching now, and it is convenient for the algorithm to converge and find the extreme point. Therefore, in the process of the particle swarm optimization algorithm, some methods are needed to adjust the inertia factor, so that the algorithm can achieve a balance between the global search and the accurate search.

In the literature [8], the most common method is using to adjust the inertia factor by the strategy of the linear decrease. And the literature [8] gives the equation:

$$\omega = (\omega_{max} - \omega_{min}) * (T_{max} - t) / T_{max} + \omega_{min} \tag{7}$$

The literature [14] is suggested that the changing trend of the inertia factor can use the convex function or the concave function, and it also gives the equation:

$$\omega = (\omega_{max} - \omega_{min}) * (t / T_{max})^2 + (\omega_{max} - \omega_{min}) * 2 * t / T_{max} + \omega_{max} \tag{8}$$

$$\omega = -(\omega_{max} - \omega_{min}) * (t / T_{max})^2 + \omega_{max} \tag{9}$$

Through the comparative analysis, it is found that the performance of the linear decrease strategy is better than the convex function, and the performance of the concave function is better than the linear decreasing strategy.

In this paper, the probability model is proposed based on the probability model, and the equation of the inertia factor is proposed as:

$$\omega(t) = (\omega_{max} - \omega_{min}) / 2 + P_G * \cos(t * \pi / T_{max}) \tag{10}$$

According to equation (10), the symbol ω_{max} and ω_{min} represent the maximum value and the minimum value of the inertia factor. The symbol t represents the current number

of iterations and the symbol T_{max} represents the maximum number of iterations. The symbol P_G, which means the probability that the particle is close to the global optimal value, is calculated by the probability model. If the symbol P_G is larger, the possibility of approaching the globally optimal value would be greater. This paper puts forward that the change of the inertia factor is the combination of concave function and convex function. The changing trend of the inertia factor will gradually decrease. When the iteration goes to a certain extent, the changing trend of the inertia factor should be slow reduction at first, but the inertia factor should be rapid reduction at the later stage of the particle swarm optimization algorithm.

2.2. Learning Factor

The learning factor is an important parameter in the particle swarm optimization algorithm, and it represents the cognitive behavior of itself and the whole particles' group. The learning factor c_1 and c_2 show that how to affect the global optimal value by the individual's experience and the group's experience, and the learning factor reflects the behavior of the information exchange between the particles. This is assuming that the value of c_1 can be vary large, the particle will search too much in the local space, where the particle is searching now. Otherwise, if the larger c_2 value is set at the beginning, the particle swarm algorithm will converge to the local optimal value quickly. So, particle swarm optimization algorithm should set a larger value of c_1 and a smaller value of c_2 in the early. In this way, the particles are dispersed into the space, while the majority of the particles emphasize the "individual's independent consciousness" and fewer particles are affected by the "social consciousness". This can increase the diversity of particles within the group. With the increase of the number of iterations, the c_1 will decrease and the c_2 will increase. Then it helps strengthen the convergence ability of the algorithm to the global optimal value.

In order to easily use the learning factor, many scholars always take the experience value. And it is convenient to calculate in different application environment. Many scholars have proposed many methods of changing the learning factors. For example, literature [12] proposes the theory of changing the learning factor by the number of iterations, and gives the equation as follows:

$$c_1(t) = c_{max} - (c_{max} - c_{min}) * t / T_{max} \tag{11}$$

$$c_2(t) = c_{max} + (c_{max} - c_{min}) * t / T_{max} \tag{12}$$

In the literature [13], it is proposed that the learning factors can be changed according to the running time of the algorithm, and gives the equation:

$$c_1 = (c_{1f} - c_{1f}) * iter / MAXITR + c_{1i} \tag{13}$$

$$c_2 = (c_{2f} - c_{2f}) * iter / MAXITR + c_{2i} \tag{14}$$

A new way of changing the learning factor is proposed. The new way uses the probability model to calculate the probability, and this paper gives the equation:

$$c_1(t) = c_{1init} + P_S * \cos(t * \pi / T_{max}) \tag{15}$$

$$c_2(t) = c_{2init} - P_G * \cos(t * \pi / T_{max}) \tag{16}$$

According to equation (15) and (16), the symbol c_{1init} represents the initial value of the individual's cognition of itself, and the symbol c_{2init} represents the initial value of the cognition of the group. The symbol t represents the current number of iterations and the symbol T_{max} represents the maximum number of iterations. The symbol P_G, which means the probability that the particle is close to the globally optimal value, is calculated

by the probability model. The symbol P_S, which means the probability that the particle is close to particle's own historical optimal value, is also calculated by the probability model. The changing trend of the learning factor is also the combination of concave function and convex function.

This chapter mainly introduces the basic idea of the particle swarm optimization algorithm and the important role of the inertia factor and the learning factor in particle swarm optimization algorithm. This paper proposes a method of using probability model to modify the inertia factor and the learning factor. In the next chapter, the particle swarm optimization algorithm which based on probability model will be introduced, and how to calculate the probability.

3. Improvement of PSO Algorithm based on Probability Model

The probability model checking is a kind of stochastic process system model with the characteristic of Markov. Probability model mainly has four kinds of probability models, which contains the labeled transition system (LTS), discrete-time Markov Chains (DTMC), continuous-time Markov Chains (CTMC) and Markov decision processes (MDP). In this paper, the particle swarm optimization algorithm needs to meet three conditions. Firstly, time and probability are independent from each other; secondly, the number of transitions between states is limited; finally, it satisfies the time-homogenous. So discrete-time Markov is used in this paper.

Discrete-time Markov Chains (DTMC) can be definition as $D = (S, s_{init}, P, L)$. The symbol S is the set of states; the symbol s_{init} is the set of initial states, which is satisfy the condition $s_{init} \in S$. The symbol P represents is probability matrix, which is satisfy the condition $P : S * S \to [0,1]$. The symbol L represents the set of atomic propositions, and the symbol L is satisfying the condition $L : S \to 2^{AP}$.

3.1. Tasks Encoding

The first step of the particle swarm optimization algorithm is to initialize the particles' group. Assumes that the cloud computing platform has N tasks and R resources, and then N tasks are divided into S subtasks. By satisfying the condition $S > N$, it can ensure that all resources are effectively used. The total number of subtasks should meet the following equation:

$$SumSubTask = \sum_{i=1}^{N} Task(i) \tag{17}$$

According to equation (17), the symbol $SumSubTask$ means the total number of subtasks. The symbol $Task(i)$ represents the number of all the subtasks which is contained in the ith task. Next, the position of each particle in the particle swarm optimization algorithm can be defined by the vector x_i, which is defined as:

$$x_i = \left\{x_{i1}, x_{i2}, \ldots, x_{i(k-1)}, x_{ik}\right\}, k \in \left[1, SumSubTask\right], i \in \left[1, R\right] \tag{18}$$

The kth subtask which is assigned to the ith particle can be represented as x_{ik}, which should satisfy the conditions $x_{ik} \in [1, R]$ and $x_{ik} \in \mathbb{N}^+$. Then, the velocity of each particle in the particle swarm optimization algorithm can be defined by the vector v_i, which is defined as:

$$v_i = \left\{v_{i1}, v_{i2}, \ldots, v_{i(k-1)}, v_{ik}\right\}, k \in \left[1, SumSubTask\right], i \in \left[1, R\right] \tag{19}$$

The velocity of the kth subtask which is assigned to the ith particle can be represented as v_{ik}, which should satisfy the condition $v_{ik} \in [-\mathrm{R}, R]$.

For the encoding rules of the subtasks, this paper orders the subtasks in accordance with the order of the natural numbers and gives the equation:

$$Code[i,j]=\begin{cases} j, i=1 \\ \sum_{i=2}^{i-1} Task(i)+j, i>1 \end{cases} \tag{20}$$

According to equation (20), the symbol $Code[i,j]$ means the coding of the jth subtask from the ith task, and the symbol $Task(i)$ represents the number of all the subtasks which is contained in the ith task.

3.2. Fitness Function

In this paper, the process of the particle swarm algorithm is implemented by using the logs to record the related data, such as time, results, etc. Now, people not only want to pursue the shortest time to complete the tasks in cloud computing environment, but also hope to take into the running cost of machines. So fitness functions are needed to solve this requirement.

Time cost can be understood as the length of completion time for all tasks. Then this paper proposes that it can use Total Time Cost (TTC) which means the total time that all the tasks have been completed already. The TTC should meet the following equation:

$$TTC(i)=\max_{1\le i\le n} RTime(i,j) \tag{21}$$

According to equation (21), the symbol $TTC(i)$ means that the ith available resource finishes all the subtasks which is assigned to its. The symbol $RTime(i,j)$ represents the time when the jth subtask can be finished by the ith available resource. The symbol n represents the number of the tasks. So this paper defines the fitness function of the time cost as follows:

$$F_{time}(i)=\frac{1}{TTC(i)}, i\in[1,R] \tag{22}$$

Because the running cost has different evaluation methods in different cloud computing environment, so the running cost of this paper mainly considers the broadband cost, the power cost and the maintenance cost. Total Running Cost (TRC), which is required to complete all tasks, is proposed in this paper. And this paper uses Unit Running Cost (URC) to express the unit time of the running cost of machines, then the TRC should meet the following equation:

$$TRC(i)=TTC(i)*URC(i),\ \ i\in[1,R] \tag{23}$$

So this paper defines the fitness function of the running cost as follows:

$$F_{cost}(i)=\frac{1}{TRC(i)}, i\in[1,R] \tag{24}$$

In the particle swarm optimization algorithm, the best position of the particle will be replaced by the current position only when the fitness value of the current position is better than the historical optimal value. This paper uses Pb_i to express the best position of the ith particle. In the comparison process, this paper gives priority to the time cost, and then considers the running cost. So this paper gives the equation:

$$Pb_i(t+1)=\begin{cases} Pb_i(t), F_{time}(x_i(t+1))<F_{time}(x_i(t)) \\ Pb_i(t), F_{time}(x_i(t+1))=F_{time}(x_i(t))\,\&\,F_{cost}(x_i(t+1))\le F_{cost}(x_i(t)) \\ x_i(t+1), F_{time}(x_i(t+1))=F_{time}(x_i(t))\,\&\,F_{cost}(x_i(t+1))>F_{cost}(x_i(t)) \\ x_i(t+1), F_{time}(x_i(t+1))>F_{time}(x_i(t)) \end{cases} \tag{25}$$

In this paper, the symbol G_{best} is used to show the best position of all the particles, then G_{best} should satisfy the equation:

$$G_{best}(t) = \max_{1 \le i \le R} \left\{ F_{cost}\left(F_{time}(Pb_i(t)) \right) \right\} \tag{26}$$

According to equation (25) and (26), the symbol t represents the number of iterations and the symbol R represents the number of particles.

3.3. Construction and Calculation of Probability Model

Through the improvement of particle swarm optimization algorithm, there are two key parameters P_G and P_S. These two parameters control the convergence of the algorithm. So how to calculate these parameters calculate according to the probability model? Before introducing the detail of the calculation process, the particle swarm optimization algorithm which is based on the probability model will be introduced first.

The description of the particle swarm optimization algorithm, which is based on the probability model, is showing as follows:

①The initialization work of computing nodes should be finished at first, then go to ②;

②Initialize the particles' position and velocity, then go to ③;

③Initialize parameters, such as: the inertial factor, the learning factor and the maximum iteration, then go to ④;

④According to equation (22) and (24), then calculating the fitness value of particles, go to ⑤;

⑤According to equation (25) and (26), the historical optimal value of the particle and the global optimal value as well as the present position of the particle are compared. If the present position of the particle is better than the historical optimal value of the particle, go to ⑥. If the present position of the particle is better than the global optimal value, go to ⑦;

⑥Update the historical optimal value of the particle, and if the present position of the particle is better than the global optimal value, go to ⑦. Otherwise, go to ⑧;

⑦Update the global optimal value, go to ⑧;

⑧If the number of iterations reaches the maximum, go to ⑨; otherwise, update new inertia factor , new learning factor, new position and new velocity, go to ④;

⑨Output of the global optimal value; According to the logs and the probability model, P_G and P_S are recalculated.

According to the definition of discrete time Markov chains, the model of the particle swarm optimization algorithm, which is based on probability model, can be defined as $D = (S, s_{init}, P, L)$, and the contents of each element in the tuple are showing as follows:

① $S = \{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9\}$;

② $s_{init} = S_0$;

③
$$P=\begin{bmatrix} 0 & P_{01} & P_{02} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{12} & P_{13} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{32} & 0 & P_{34} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{42} & 0 & 0 & P_{45} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & P_{56} & 0 & P_{58} & 0 \\ 0 & 0 & P_{62} & 0 & 0 & 0 & 0 & P_{67} & P_{68} & 0 \\ 0 & 0 & P_{72} & 0 & 0 & 0 & 0 & 0 & P_{78} & 0 \\ 0 & 0 & P_{82} & 0 & P_{84} & 0 & 0 & 0 & 0 & P_{89} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \begin{cases} P_{01}+P_{02}=1 \\ P_{12}+P_{13}=1 \\ P_{32}+P_{34}=1 \\ P_{42}+P_{45}=1 \\ P_{56}+P_{58}=1 \\ P_{62}+P_{67}+P_{68}=1 \\ P_{72}+P_{78}=1 \\ P_{82}+P_{84}+P_{89}=1 \end{cases};$$

④
$$\begin{cases} L(S_0)=\{Initialize\,the\,computing\,nodes\} \\ L(S_1)=\{Initialize\,the\,particles'\,position\,and\,velocity\} \\ L(S_2)=\{Abnormal\,state\} \\ L(S_3)=\{Initialize\,parameters\} \\ L(S_4)=\{Calculate\,the\,particles'\,fitness\,values\} \\ L(S_5)=\{Compare\,the\,fitness\,values\} \\ L(S_6)=\{Update\,particles'\,the\,historical\,optimal\,position\} \\ L(S_7)=\{Update\,the\,global\,optimal\,position\} \\ L(S_8)=\{Determine\,whether\,the\,number\,of\,iterations\,reaches\,the\,maximum\,value\} \\ L(S_9)=\{End\} \end{cases}$$

In this paper, the state transition diagram can be constructed by the probability model (see Figure 1).

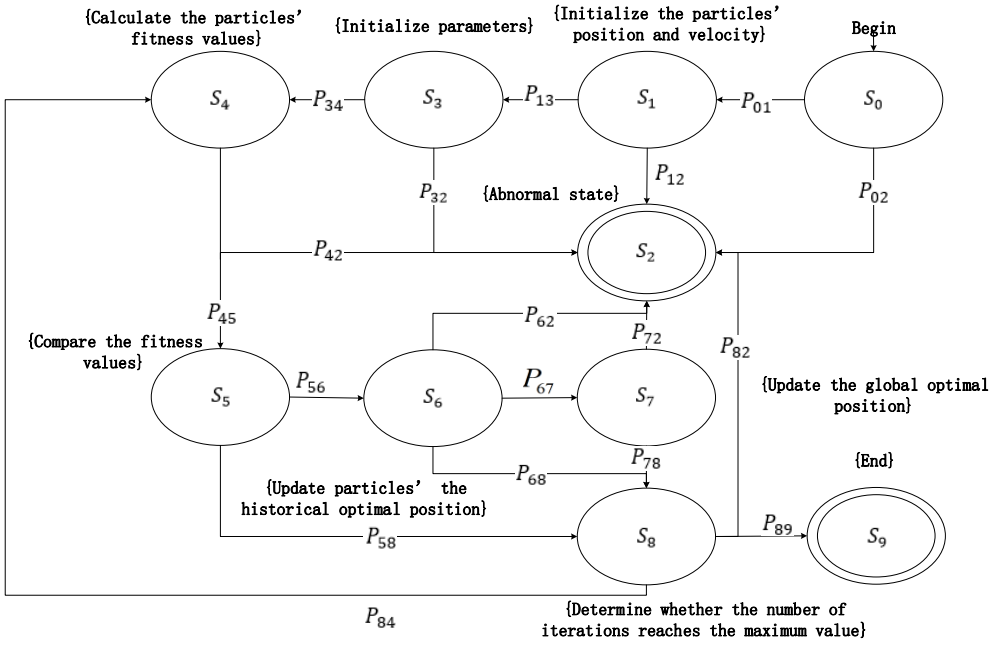


Figure 1. The State Transition Diagram

The probability that transfers between the states can be calculated by the logs. According to the boundary reachability of the probability model, this paper defines $PrReach^{\leq k}(s,T)$ that means the probability of transition from the state s to the set T in k times. The symbol $PrReach^{\leq k}(s,T)$ should satisfy the equation:

$$PrReach^{\le k}(s,T)=\begin{cases}1, s\in T\\ 0, k=0\\ \sum_{s'\in S}P(s,s')*PrReach^{\le k-1}(s',T), k>0\&s\notin T\end{cases} \tag{27}$$

According to the equation (27), this paper gives the equation of P_S and P_G. The symbol P_G represents the probability that the particle swarm algorithm finds the global optimal value (within the range of error), and is also the probability of the transition from a state to the state S_7. So there are the equations of P_S and P_G:

$$P_G = PrReach^{\le k}\left(S_0,\{S_7\}\right) \tag{28}$$

$$P_S = PrReach^{\le k}\left(S_0,\{S_6\}\right) \tag{29}$$

This chapter mainly introduces the boundary reachability of the probability model to change the inertia factor and the learning factor. Through this way can control the process of searching and converging. The Chapter 4 also did some experiments to verify the feasibility of this algorithm.

4. Experimental Results and Examples

In this paper, the test environment is based on the hybrid cloud platform which contains of ten CPU nodes, and the parameters of cloud computing environment can refer to the table 1. The particle swarm optimization algorithm is mainly applied on the cloud computing platform, which is mainly used to deal with image rendering and video key frame rendering. The initialization parameters of the particle swarm optimization algorithm can refer to the table 2.

Table 1. The Parameters of Cloud Computing Environment

Node Group	The number of nodes	OS	CPU	Graphics card	Memory
Group 1	4	CentOS	i3-4160	GT720	4GB
Group 2	3	Windows 7	i5-4460	GT720	4GB
Group 3	3	Windows 7	i7-4790	GT720	4GB

In order to compare with this paper's algorithm and this paper provides two other algorithms for comparison:

① The first algorithm is that P_G and P_S are set to a fixed value of 1, so that the change trend of the inertia factor and the learning factor is the same as the algorithm in this paper. This set of contrast is used to observe whether the probability model plays a role in the improvement of the algorithm.

② The second algorithm uses the equation (3) and (7) to change the inertia factor and the learning factor by the linear decrease. This group of contrast is used to detect whether the nonlinear change of the inertia factor and the learning factor is better than the linear one. The initialization parameters of the task scheduling algorithm in experiment are referred to the table 2.

Table 2. The Parameters of Algorithm

Parameters of this paper's algorithm	Value	Parameter name of ②	Value
R	10	R	10
The number of tasks	10-50 (The number of tasks is randomly assigned according to the amount of the pictures.)	The number of tasks	10-50 (This data is the same as this paper's algorithm.)
The size of pictures	5760*4320	The size of pictures	5760*4320
ω_{max}	1	ω_{max}	1
ω_{min}	0.1	ω_{min}	0.1
c_{1_int}	2.5	c_{max}	3.0
c_{2_int}	0.8	c_{min}	0.3
T_{max}	500	T_{max}	500
P_G	0.5	-	-
P_S	0.5	-	-

4.1. Experiment and Analysis of the Algorithms

In the experiment, the number of rendering images can be 10, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, and the performance of the algorithm is analyzed by comparing the time cost and the running cost. Here additional description of the experimental conditions: Firstly, this paper divides tasks according to the different number of pictures, and the total number of tasks must be less than the total number of pictures. Secondly, each task contains that the number of pictures can't exceed 20% of the total number of pictures. Then, the average time cost and the average running cost are calculated by the average of F_{time} and F_{cost}. Finally, if a task is assigned to only one a picture, the subtask will be divided by the way of image segmentation.

From Figure 2, Figure 3, and the comparison of the results can be known: when the number of tasks is small, the average time cost and the average running cost of the three algorithms are relatively high. With increasing the number of tasks, the average time cost and the average running cost of the three algorithms are gradually deceasing, but the difference between the three algorithms is very small. When the number of tasks increases to a certain extent, the average time cost and the average running cost which relative to the other two algorithms is significantly reduced. Eventually, the three algorithms will gradually stabilize. It shows that the particle swarm algorithm based on probability model can effectively reduce the time cost and the running cost, especially when the number of tasks is large.

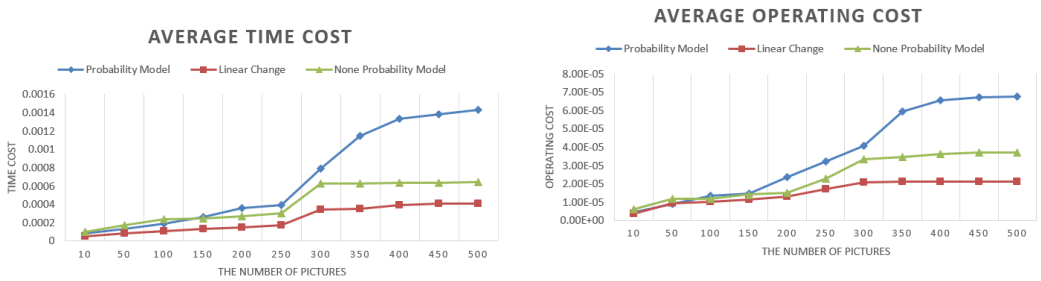


Figure 2. The Chart of the Average Time Cost and the Average Running Cost

4.2. The Example of the Cloud Rendering Project

The particle swarm algorithm based on probability model is applied to the project "The integrated service platform based on hybrid cloud rendering ". The project is mainly to achieve the cloud rendering platform for public service, and this platform can quickly and efficiently render pictures or video key frames, and feedback pictures to the users. Figure 3 is related to the finished product of this platform.

Figure 3. The Products of the Cloud Rendering Platform (1-4)

5. Conclusions

Aiming at the task scheduling problem in cloud computing environment, this paper proposes the particle swarm optimization algorithm based on the probability model to solve the problem of task scheduling. The mechanism of using the probability model to calculate the inertia factor and the learning factor is established. By this method not only can avoid falling into the historical optimal value, but also can effectively complete the task scheduling. At the same time, the fitness functions of the time cost and the running cost are considered, so that the resources of the cloud computing platform can be effectively utilized. In the future, the authors will continue to finish the formal verification of probability models, in order to test whether the algorithm meets the actual requirements.

Acknowledgments

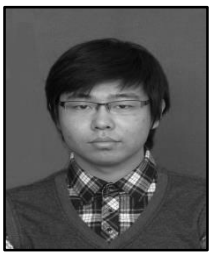
This paper is supported by National Natural Science Foundation of China (NFSC) under Grant No.61502294, Natural Science Foundation of Shanghai under Grant No.15ZR1415200, Innovation Project of Next Generation Internet Technology of CERNET under Grant No. NGII20150609, Foundation of Science and Technology Commission of Shanghai Municipality under Grant No. 14590500500, and The Special Development Foundation of Key Project of Shanghai Zhangjiang National Innovation Demonstration Zone under Grant No. 201411-ZB-B204-012.

References

[1] Ullman J D. NP-Complete Scheduling Problems.[J]. Journal of Computer & System Sciences, 1975, 10(3):384-393.
[2] Zhou Z, Hu Z. Task Scheduling Algorithm based on Greedy Strategy in Cloud Computing[J]. Open Cybernetics & Systemics Journal, 2015, 8(1):111-114.
[3] Prakash Shiv, Vidyarthi Deo Prakash. Maximizing availability for task scheduling in computational grid using genetic algorithm[J]. Concurrency & Computation Practice & Experience, 2015, 27(1):193-210.
[4] Xu Y, Li K, He L, *et al*. A Hybrid Chemical Reaction Optimization Scheme for Task Scheduling on Heterogeneous Computing Systems[J]. IEEE Transactions on Parallel & Distributed Systems, 2015:3208-3222.
[5] Tillenius M, Larsson E, Badia R M, *et al*. Resource-aware task scheduling[J]. ACM Transactions on Embedded Computing Systems (TECS), 2015, 14(1): 5.
[6] Panda S K, Jana P K. Efficient task scheduling algorithms for heterogeneous multi-cloud environment[J]. Journal of Supercomputing, 2015, 71(4):1505-1533.
[7] Song M P, Gu G C. Research on particle swarm optimization: a review[C]// Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on. IEEE, 2004:2236-2241 vol.4.

[8] Shi Y, Eberhart R C. Empirical study of particle swarm optimization[C]// Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on. IEEE, 1999:32-49.
[9] Shi Y, Eberhart R C. Fuzzy adaptive particle swarm optimization[C]// Evolutionary Computation, 2001. Proceedings of the 2001 Congress on. IEEE, 2001:101-106 vol. 1.
[10] Eberhart R C, Shi Y. Tracking and optimizing dynamic systems with particle swarms[C]// Evolutionary Computation, 2001.Proceedings of the 2001 Congress on.IEEE, 2001:94-100 vol. 1.
[11] Wang S, Lo D, Jiang L. Active code search: incorporating user feedback to improve code search relevance[C]// Acm/ieee International Conference on Automated Software Engineering. ACM, 2014:677-682.
[12] Suganthan P N. Particle Swarm Optimiser with Neighbourhood Operator[C]// Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on. IEEE, 1999.
[13] Ratnaweera A, Halgamuge S, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3):240-255.
[14] Chen Guimin, Jia Jianyuan, Han Qi. Study on the Strategy of Decreasing Inertia Weight in Particle Swarm Optimization Algorithm[J]. Journal of Xi'an Jiao Tong Universtity, 2006, 40(1):53-56.

Authors



Li Ruizhi, The author is a Master candidate in graduate students in School of Computer Engineering and Science Shanghai University. His main research is about 3D visualization and cloud rendering. He works at the Information Technology Office of Shanghai University during the study of Master.