Applying Probability Model to The Genetic Algorithm Based Cloud Rendering Task Scheduling

Guobin Zhang[1,3], Huahu Xu[1,3]
1. School of Computer Engineering and Science, Shanghai University, 200444 Shanghai, P.R. China
3. Shanghai Shang Da Hai Run Information System Co., Ltd, Shanghai 200444,
mr_zhang2011@163.com

Honghao Gao[1,2] , Ankang Liu[4]
2. Computing Center, Shanghai University, 200444 Shanghai, P.R. China
4. Department of Mathematics, Shanghai University, 200444 Shanghai, P.R. China

***Abstract*—There are huge amount of tasks and data to be processed in cloud rendering environment. How to effectively schedule them is the key to ensure the overall performance of the cloud rendering environment. In this paper, an improved task scheduling algorithm based on genetic algorithm (PMGA) and probability model is proposed, which aims to minimize the total time and cost of task scheduling . First, the fitness function relating to the total time and task cost is improved under the consideration of the user's satisfaction to the rendering services. Then, a probability model is constructed for the scheduling algorithm, which is used to achieve the non-linear adaptive adjustment of the crossover rate function and the mutation rate function. As a result, the evolutionary ability of poor individuals in the population can be enhanced, avoiding the stagnation in the early stages. Finally, experiments are performed to demonstrate that PMGA has a better ability of optimization than that of traditional adaptive genetic algorithm (AGA). Our approach contributes to reduce the total time and the scheduling cost of the cloud rendering tasks.**

Keywords-Cloud Rendering; Task scheduling; Genetic Algorithm; The Fitness Function; Probability Model; The Crossover Rate; The Mutation Rate

I. INTRODUCTION

Due to the advantages of cloud computing, cloud rendering is considered as a promising application, focusing on the rendering services and providing the computing and data resources. The basic principle of cloud rendering is that the cloud rendering system divides tasks submitted by users into some independent subtasks, and dynamically allocates the subtasks to resource nodes based on appropriate scheduling strategies. After all tasks are completed, the rendering results will be merged together and returned to the users [1,2].

Most cloud computing platforms adopt MapReduce [3] to support distributed computing. Many studies have shown that the problem of task scheduling under cloud environment is an NP-complete problem [4]. Cao et.al., [5] proposed a task scheduling optimization algorithm based on ABC algorithm, which reduces the total time of completion task. Xu et.al., [6] proposed a multi-adaptive particle swarm optimization strategy for task scheduling in cloud, which has taken into account the task completion time, the operating cost and the balancing of resource load. Jie et.al., [7] proposed a genetic simulated annealing algorithm for task scheduling with dual fitness, which effectively balances the demands of the users and improve the users' satisfaction. Liu et.al., [8] proposed a task scheduling algorithm based on genetic algorithm and ant colony algorithm, which improves the speed of getting the optimal solution.

However, due to the heterogeneity of the cloud computing platform, any single intelligent group algorithm is easy to fall into the local optimum, premature and other defects. The improvement is mainly focused on the nature of the algorithm itself, ignoring the guiding role of the information in the process of evolution. The adaptability of these algorithm are not robust enough, and the scheduling time and cost can not be balanced well. The efficient task scheduling algorithm is still a long-term challenge in the research of cloud computing.

Genetic algorithm was proposed by Professor John Holland of the University of Michigan in the 1970s. It is a global optimization probabilistic search algorithm based on the natural evolution process including selection, crossover and mutation operations on the genes of individuals or potential solutions [9]. The basic principle is to apply a coding technique to the individual in the population, then simulate the evolutionary process of the population. The standard genetic algorithm (SGA) usually has only one population, the crossover rate and the mutation rate are fixed ones, resulting in its slow convergence.

Srinivas et.al., [10] proposed an adaptive genetic algorithm (AGA). It took a dynamic automatic adjustment for crossover rate and mutation rate, which is based on the fitness value and the convergence level of the current group. This algorithm improved the convergence speed of the genetic algorithm, however, it is slightly slow at the beginning of the evolution process. Beyond that, only one fitness function is used to measure the adaptation of individual to different environment.

In order to improve the efficiency of task scheduling in the cloud rendering platform, an improved adaptive genetic algorithm based on probability model is proposed. The fitness function is established for the total time and total cost of the rendering tasks, which aims to achieve the minimum time and minimum cost. Then, the crossover rate function and mutation rate function is improved by applying the probability model, which effectively retains the excellent individuals in the evolution process of population and enhances the crossover rate and mutation rate of poor individuals.

The rest of this paper is as follows: In Section II, it describes the problem of tasks scheduling in cloud rendering environment, and gives the allocation pattern of tasks and the calculation method of total completion time and total cost of cloud tasks. In

DOI reference number: 10.18293/SEKE2017-193

Section III, the implementation and process of PMGA are introduced in details. In Section IV, PMGA and AGA are compared and analyzed under the same experimental environment. The experimental results proves that the performance of cloud rendering is improved by using PMGA for task scheduling. Finally, in Section V, we conclude this research and discuss future works.

II. Descriptions About the Cloud Rendering Task Scheduling

Cloud rendering task scheduling is to distribute the tasks to the compute nodes by applying appropriate scheduling strategies, which enables to minimize the time, lower the cost, and keep the load balancing of resources [11].

Assuming that the cloud rendering system user submits *T* tasks, the system has *R* resources. Then, the total subtask number of the task *t* is defined as $taskNum(t)$, the total subtask number is defined as $subTaskNum$. Then

$$subTaskNum = \sum_{t=1}^{T} taskNum(t) \tag{1}$$

Subtasks are numbered according to the sub-sequence numbering method. If the sequence number of *j*th subtask of *i*th task is *m*, then

$$m = \sum_{k=1}^{i\text{-}1} taskNum(k) + j \tag{2}$$

The ETC matrix [12] is used to record the execution time of each sub task, the ECC matrix is used to record the execution cost of each sub task. Consequently, the total time and total cost of all the sub tasks are as follows:

$$F_{time(x)} = \max_{r=1}^{R} \sum_{i=1}^{n} Time(r,i) \quad . \tag{3}$$

$$F_{cost(x)} = \max_{r=1}^{R} \sum_{i=1}^{n} Cost(r,i) \tag{4}$$

Where $Time(r,i)$ and $Cost(r,i)$ represents the execution time and cost of *i*th subtask on *r*th resource, *n* is the subtask number on *r*th resource. $Cost(r,i)$ is defined as follows:

$$\begin{aligned} Cost(r,i) = w_{cpu}Cost_{i\text{-}cpu}(r,i) + w_{menmory}Cost_{i\text{-}menmory}(r,i) + \\ w_{storage}Cost_{i\text{-}storage}(r,i) + w_{bandwidth}Cost_{i\text{-}bandwidth}(r,i) \end{aligned} \tag{5}$$

Where $Cost_{i-cpu}$, $Cost_{menmory}$, $Cost_{storage}$ and $Cost_{bandwidth}$ are the cost of *CPU*, *Memory*, *Storage* and *Bandwidth*, respectively. w_{cpu}, $w_{menmory}$, $w_{storage}$ and $w_{bandwidth}$ are weighing coefficients of the four kind of resource costs, respectively. They are both in [0,1], and satisfy the following equation:

$$w_{cpu} + w_{menmory} + w_{storage} + w_{bandwidth} = 1 \tag{6}$$

The value of weighing coefficient can be set differently to measure the user satisfaction of scheduling results. They are all fixed to 0.25 in our experiments to balance the resources costs.

III. Task Scheduling of PMGA in Cloud Rendering

A. Encoding and Decoding

As for the solution of cloud task scheduling based on PMGA, it needs to establish a mapping relationship between the solution of the problem and the individual of the algorithm. The current encoding modes of chromosome always include direct encoding and indirect encoding. This paper adopts indirect encoding mode to encode the node occupied by each sub task, taking into account the division of the operation under cloud rendering. The length of the chromosome depends on the number of sub tasks, therefore, a chromosome corresponds to a task scheduling strategy. The length of the individual chromosome equals to the total number of tasks, the value of the chromosome gene is the cloud resource number occupied by the task at that location.

For example, if users submit 3 tasks in a certain time, the resource number of cloud rendering system is 3. Task 1 is divided into 1, 2, 3 three subtasks. Task 2 is divided into 4,5,6 three subtasks, Task 3 is divided into 7,8,9,10 four subtasks. The chromosome length is 10, the value of each gene is [1,3]. Then The chromosome {1,2,1,2,1,3,2,1,2,3} is a feasible scheduling strategy as shown in Table I.

TABLE I. Encoding mode of chromosomes

Subtask	1	2	3	4	5	6	7	8	9	10
Resource	1	2	1	2	1	3	2	1	2	3

It is necessary to decode them to obtain the distribution of the subtasks on each node. After that, the subtask performed on resource 1 is {1,3,5,8}, the subtask executed on resource 2 is {2,4,7,9}, the subtask executed on resource 3 is {6,10}.

B. Initialization of the Population

Assuming that the initial size of the population is $SCALE$, the total number of subtask is M, the number of resources is $RESOURCE$, the initialization of the population can be described as follows: the system randomly produce $SCALE$ chromosomes, the length of the chromosome is M, the values of genes are $\left[1, RESOURCE\right]$, the chromosomal gene is generated randomly within this range.

C. Double Fitness Function Model

The fitness function generates a fitness value of each chromosome. The value indicates how suitable for solving a task scheduling problem a chromosome is. The fitness function for SGA and AGA is only based on execution time of tasks. But, task scheduling problems in cloud rendering are different from general task scheduling problems because rendering services in cloud rendering environment are offered between cloud users and providers. Therefore, this paper defines $FIT_{time}(i)$ as the fitness function of the total completion time of the tasks, and $FIT_{cost}(i)$ as the fitness function of the total cost of the tasks. The calculation formula of this two function is as follows:

$$FIT_{time}(i) = \frac{1}{1 + Time_i(R_j)} \tag{7}$$

$$FIT_{cost}(i) = \frac{1}{1 + Cost_i(R_j)} \tag{8}$$

Where R_j is the *j*th resource of the *i*th individual, $Time_i R(j)$ is the completion time of the *i*th individual. $Cost_i R(j)$ is the cost of the *i*th individual. Formula (9) is used as the fitness function of the algorithm.

$$Fit(i) = w_{time}FIT_{time}(i) + w_{cost}FIT_{cost}(i) \tag{9}$$

In the formula, w_{time} and w_{cost} are weighing coefficients, and $w_{time} + w_{cost} = 1$, which can be set according to the user's needs. w_{time} and w_{cost} are both set to 0.5 in our experiments in order to balance the proportion of the total time and the total cost of the task.

D. Probability Model and Generic Operation

1) Individual Selection Based On Double Fitness Function Model

This paper adopts the roulette wheel selection method [13] to do the selection operation. The selection probability of each individual was calculated according to the fitness function (7) and fitness function (8), and give the equations:

$$P_1(i) = \frac{FIT_{time}(i)}{\sum_{j=1}^{SCALE} FIT_{time}(j)} \tag{10}$$

$$P_2(i) = \frac{FIT_{Cost}(i)}{\sum_{j=1}^{SCALE} FIT_{Cost}(j)} \tag{11}$$

When choosing the next generation of individuals, one of the probability is selected from P_1 and P_2 to choose populations with probability c_1 and c_2 respectively, and $c_1 + c_2 = 1$. In this way, the population both contains individual with shortest completion time and minimum cost, which provides good foundation for the next generation. c_1 and c_2 are both set to 0.5 in our experiments.

2) Improved crossover operation and mutation operation based on the probability model

Crossover operation is to match the individuals from the population randomly in pairs, and exchange their partial gene with a probability by following a particular probability rule for each individual. The mutation operation is to change the genetic value of a certain locus on each individual in the population to other alleles at a certain probability.

The crossover rate function P_c and the mutation rate function P_m determine the speed of updating and searching for the population. In SGA, P_c and P_m are both constant values. SGA is inefficient and is easy to be precocious for complex multivariable optimization problem. AGA had made some improvements based on the fitness value [10]. P_c and P_m in AGA are defined as follows:

$$P_C = \begin{cases} \dfrac{k_1(f_{\max} - f')}{(f_{\max} - f_{avg})}, & f' \geq f_{avg} \\ k_3, & f' < f_{avg} \end{cases} \tag{12}$$

$$P_m = \begin{cases} \dfrac{k_2(f_{\max} - f)}{(f_{\max} - f_{avg})}, & f \geq f_{avg} \\ k_4, & f < f_{avg} \end{cases} \tag{13}$$

Where $f_{\max}$ is the maximum fitness of the population, f_{avg} is the average fitness of the population, f' is the greater fitness of the two individuals which are to have cross operation, f is the fitness value of the mutation individual that are to have mutation operation. k_1, k_2, k_3, k_4 are [0,1]. In literature [10], $k_1 = k_3$ =1.0, $k_2 = k_4$ =0.5.

P_c and P_{m} in AGA perform a linear change according to the average fitness and maximum fitness of individual as shown in Fig. 1 and Fig. 2 . P_c and P_m are equal to zero when the fitness value is relatively large. However, at the beginning of the algorithm, the individuals with higher fitness in the population may not be the optimal solution. Thus, it may lead to the occurrence of a local optimal solution.

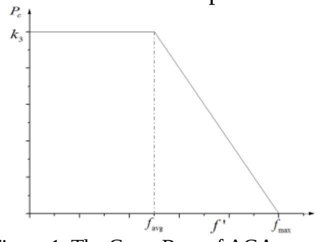

Figure 1. The Cross Rate of AGA

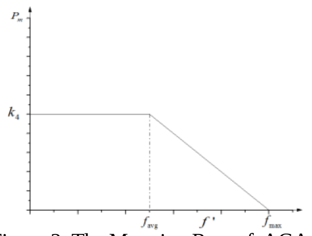

Figure 2. The Mutation Rate of AGA

The probability calculated by the probability model is applied to the crossover rate function and the mutation rate function in PMGA. The definition of the crossover rate and mutation rate function in PMGA are as follows:

$$P_C = \begin{cases} k_{c1} * \left(P_{ck} - \left|(P_{ck} - P_{pro-c})\right|^{\left(M_c + \frac{(f - f')}{(f_{\max} - f_{avg})}\right)} \right), & \dfrac{f_{avg} + f_{\max}}{2} < f' \leq f_{\max} \\ k_{c2} * \left(P_{ck} - \left|(P_{ck} - P_{pro-c})\right|^{\left(M_c + \frac{(f_{\max} - f')}{(f_{\max} - f_{avg})}\right)} \right), & f_{avg} < f' \leq \dfrac{f_{avg} + f_{\max}}{2} \\ P_{ck}, & f' \leq f_{avg} \end{cases} \tag{14}$$

$$P_m = \begin{cases} k_{m1} * \left(P_{mk} - \left|(P_{mk} - P_{pro-m})\right|^{\left(M_m + \frac{(f_{\max} - f)}{(f_{\max} - f_{avg})}\right)} \right), & \dfrac{f_{avg} + f_{\max}}{2} < f \leq f_{\max} \\ k_{m2} * \left(P_{mk} - \left|(P_{mk} - P_{pro-m})\right|^{\left(M_m + \frac{(f_{\max} - f)}{(f_{\max} - f_{avg})}\right)} \right), & f_{avg} < f \leq \dfrac{f_{avg} + f_{\max}}{2} \\ P_{mk}, & f \leq f_{avg} \end{cases} \tag{15}$$

Where $f_{\max}$, f_{avg}, f', f are the same as them in equation (12) and equation (13). k_{c1}, k_{c2}, k_{m1}, k_{m2} are constant values in [0,1], they are used to control the control the changing speed of P_c and P_m. $P_{\mathrm{pro}-c}$ and $P_{\mathrm{pro}-m}$ represents the value of the cross operation probability and the mutation operation probability calculated by the probability model. M_{c} and M_m are constant values in $[1,+\infty]$. The larger value they are, the higher crossover rate and the mutation rate the optimal individual has. When the fitness value is smaller than f_{avg}, the crossover rate is P_{ck}, the mutation rate is P_{mk}. The value of k_{c1}, k_{c2}, k_{m1}, k_{m2}, P_{ck} and P_{mk} can be adjusted according to the practical problems. The graph of P_c and P_m in PMGA are shown in Fig. 3 and Fig. 4, respectively.

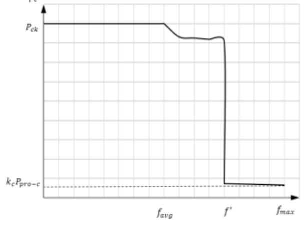

Figure 3. Crossover rate of PMGA

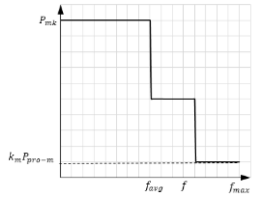

Figure 4. Crossover rate of PMGA

When $f \le f_{avg}$, P_c and P_m in PMGA is a relatively large fixed value, enabling the individuals to evolve faster. When $f_{avg} < f \le \frac{f_{avg}+f_{\max}}{2}$, P_c and P_m decrease slowly with the increase of fitness value. When $\frac{f_{avg}+f_{\max}}{2} < f \le f_{\max}$, P_c and P_m become smaller rapidly and decrease with the increase of fitness value, when the fitness reaches its maximum value, the value of P_c and P_m are equals to $K_c P_{pro-c}$ and $K_m P_{pro-m}$, respectively, which are very close to zero. Therefore, the crossover rate and mutation rate in PGMA both have the ability to find the best solution quickly and the randomness property. When the fitness is relatively large, the value of P_c and P_m are not an absolute zero value, so that the algorithm will not be in a low searching speed in the early process of evolution.

3) Generating and Computing of Probability Model

Probability model refers to a type of random system models with Markov characteristics, which is proposed by the Russian scientist Markov in 1970. According to the state space and the nature of the parameters, it can be divided into discrete time Markov chain, continuous time Markov chain, Markov decision process.

Discrete time Markov chain is a kind of Markov random process with discrete state space and discrete time. It can be defined as: $D=\{S,S_{init},P,L\}$, S is a set of finite nonempty states, indicates all possible configuration status of the modeled system. $P\times S\rightarrow[0,1]$ is the migration probability function matrix, represents the probability of the migration between the modeled system states, $\forall s\in S, \sum_{s'\in S}P(s',s)=1$. $L:S\rightarrow 2^{AP}$ is a label function with atomic propositions.

The process of PMGA is a continuous random process of selection, crossover and mutation operations. It is related to the state of the current population, regardless of the previous population status. In addition, PMGA is homogeneous, there exists a limit probability distribution in PMGA, and the transition probability is independent of time. As a result, discrete time Markov chain (DTMC) is chosen to model it.

The adaptive genetic algorithm based on probability model is described as follows:

1) Initialize computing node of cloud rendering system rendering, go to 2)

2) Initialize parameters of cloud rendering task scheduling, then go to 3).

3) Encode parameters into chromosomes.

4) Initialize the population, go to 6).

5) The cloud rendering system error or program exception.

6) Calculate the fitness of each individual and determine whether the individual satisfies the optimal condition. Then go to step 12), if not, go to 7).

7) Do select operation, that is, according to the individual selection rules, select some excellent individuals from the current population as the new population, then go to 8).

8) Calculate the crossover rate P_c, do cross operation with probability P_c, go to 9), skip cross operation with probability 1-P_c go to 10).

9) Do crossover operation, and generate a new population, then go to 10).

10) Calculate the mutation rate P_m, do mutation operation with probability P_m, go to 11), skip mutation operation with probability 1-P_m, go to 6).

11) Do mutation operation, and generate a new population, then go to 6).

12) Decode the chromosome and get the optimal solution.

According to the above process description, the state transition diagram of PMGA is shown in Fig. 5.

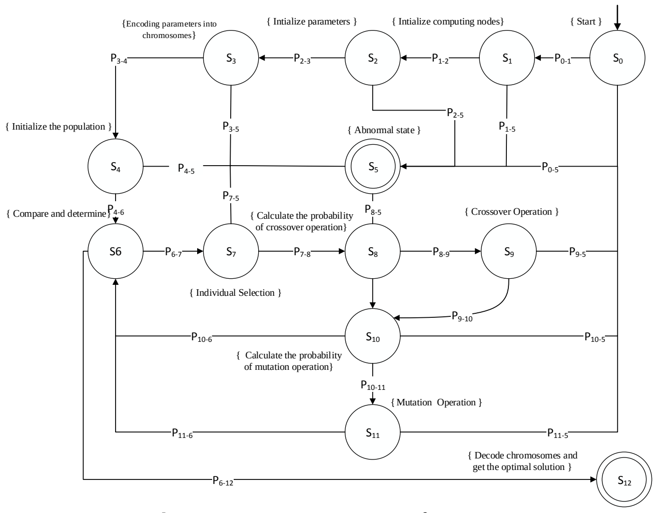

Figure 5. The State Transition Diagram of PMGA

PMGA has the properties of discrete time Markov chain. It is defined as a discrete time Markov chain D, $D=(S,S_{init},P,L)$, each element is described as follows:

- $S=\{S_0,S_1,S_2,S_3,S_4,S_5,S_6,S_7,S_8,S_9,S_{10},S_{11},S_{12},\}$.
- $S_{init}=S_0$
- $$P=\left\{\begin{matrix}
0 & P_{0-1} & 0 & 0 & 0 & P_{0-5} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & P_{1-2} & 0 & 0 & P_{1-5} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & P_{2-3} & 0 & P_{2-5} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & P_{3-4} & P_{3-5} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & P_{4-5} & P_{4-6} & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & P_{6-7} & 0 & 0 & 0 & 0 & P_{6-12}\\
0 & 0 & 0 & 0 & 0 & P_{7-5} & 0 & 0 & P_{7-8} & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & P_{8-5} & 0 & 0 & 0 & P_{8-9} & P_{8-10} & 0 & 0\\
0 & 0 & 0 & 0 & 0 & P_{9-5} & 0 & 0 & 0 & 0 & P_{9-10} & 0 & 0\\
0 & 0 & 0 & 0 & 0 & P_{10-5} & P_{10-6} & 0 & 0 & 0 & 0 & P_{10-11} & 0\\
0 & 0 & 0 & 0 & 0 & P_{11-5} & P_{11-6} & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}\right\} and \left\{\begin{array}{l}
P_{0-1}+P_{0-5}=1\\
P_{1-2}+P_{1-5}=1\\
P_{2-3}+P_{2-5}=1\\
P_{3-4}+P_{3-5}=1\\
P_{4-5}+P_{4-6}=1\\
P_{6-7}+P_{6-12}=1\\
P_{7-5}+P_{7-8}=1\\
P_{8-6}+P_{8-5}+P_{8-10}=1\\
P_{9-5}+P_{9-10}=1\\
P_{10-5}+P_{10-6}+P_{10-11}=1\\
P_{11-5}+P_{11-6}=1
\end{array}\right.$$
- $$\begin{cases}
L(S_0)=\{Start\}\\
L(S_1)=\{Intialize\ computing\ nodes\}\\
L(S_2)=\{Intialize\ parameters\}\\
L(S_3)=\{Encoding\ parameters\ \inf o\ chromosome\}\\
L(S_4)=\{Initialize\ the\ population\}\\
L(S_5)=\{Abnormal\ state\}\\
L(S_6)=\{Compare\ the\ fitness\ values\ and\ determine\ whether\ get\ the\ optimal\ solution\}\\
L(S_7)=\{Individual\ Selection\}\\
L(S_8)=\{Calculate\ the\ probability\ of\ crossover\ operation\}\\
L(S_9)=\{Crossover\ Operation\}\\
L(S_{10})=\{Calculate\ the\ probability\ of\ mutation\ operation\}\\
L(S_{11})=\{Mutation\ Operation\}\\
L(S_{12})=\{Decode\ chromosomes\ and\ get\ the\ optimal\ solution\}
\end{cases}$$

The probability of the boundary reachability in the Markov chain is defined as the probability of reaching the state set T from state S within K step transition. It is expressed as: $ProbReach^{\le k}(S,T)$, The formula is expressed as follow:

$$ProbReach^{\le k}(S,T)=\begin{cases}1, k\in T\\ 0\ , k\ =0\ \&\ s\notin T\\ \sum_{s'\in S}P(s,s')*ProbReach^{\le k-1}(s',T), k\ >0\ \&\ s\notin T\end{cases} \quad (16)$$

In probability model, $P_{\text{pro}-c}$ is the probability from state S_0 to S_9, $P_{\text{pro}-m}$ is the probability from state S_0 to S_{11}. The probability between states in the model can be obtained by the running log of the cloud rendering system. The formula of $P_{\text{pro}-c}$ and $P_{\text{pro}-m}$ is:

$$P_{pro-c}=ProbReach^{\le k}\left(S_0,T\{S_7\}\right) \quad (17)$$

$$P_{pro-m}=ProbReach^{\le k}\left(S_0,T\{S_{11}\}\right) \quad (18)$$

IV. Experiment And Analysis

A. Experimental Environment

The experiments are based on the comprehensive cloud computing service platform, which is made up of five clusters, consisting of 25 compute nodes. The tasks scheduled in the experiment are mainly the rendering of pictures and video key frames. The configuration of cloud rendering system compute node is shown in Table II.

TABLE II. COMPUTE NODE CONFIGURATION PARAMETERS OF CLOUD RENDERING SYSTEM

Cluster	The number of nodes	Operating system	CPU	GPU	Memory
1	6	Cent OS	i3-4160	GT720	4GB
2	6	Cent OS	i5-4460	GT720	8GB
3	5	Cent OS	i5-4590	GT720	8GB
4	5	Windows	i5-6500	GT720	4GB
5	3	Windows	i5-7500	GT960	8GB

B. Experimental Results and Performance Analysis

In experiment 1, the total time and total completion cost using AGA and PMGA for task scheduling are compared. Then, the average fitness value of the current new population in AGA and PMGA are compared.

The initial conditions of experiments are shown as below: (1) The number of resource is 25, the number of task is 100. (2) Each task is divided into *n* subtask, *n* is in [20,80].

The termination conditions of experiment are as follows: (1) If the number of iterations reaches the maximum G_{max} (G_{max} = 220), then the algorithm is considered to be terminated. (2) If the completion time and the total cost of continuous 40-generation are not changed, then the algorithm will be terminated.

The main parameters of PMGA and AGA in the experiment is shown in Table III.

TABLE III. MAIN PARAMETERS OF THE PMGA AND AGA

PMGA		AGA	
w_{cost}	0.50	k_1	1.00
w_{time}	0.50	k_2	0.50
k_{c1}	0.10	k_3	1.00
k_{c2}	0.50	k_4	0.50
k_{m1}	0.10	-	-
k_{m2}	0.50	-	-
$P_{\text{pro}-c}$	0.40	-	-
$P_{\text{pro}-m}$	0.08	-	-
P_{ck}	0.90	-	-
P_{mk}	0.10	-	-
M_{c}	1.00	-	-
M_m	1.00	-	-

Fig. 6 and Fig.7 show the iterative process of optimal total time and lowest total cost by using AGA and PMGA. In Fig. 6 and Fig. 7, the horizontal axis represents the number of iterations. In Fig. 6, the vertical axis represents the task total time. In Fig. 7, the vertical axis represents the total cost. In the early evolution, the convergence speed of AGA is relatively slow, with the further evolution, AGA only focuses on the total time, resulting in the loss of some potentially good genes. However, PMGA has a larger crossover rate and mutation rate for poor individuals, and the better individuals still have a small crossover rate and mutation rate. At the later stage of evolution, PMGA continues to evolve and eventually converges to the global optimal solution. While AGA still converges to the local optimum. The task completion time of PMGA is smaller than AGA, and the cost is significantly less than AGA. The experimental results show that PMGA reduces the total time and lowers the cost of the scheduling, which is an effective algorithm for the cloud rendering task scheduling.

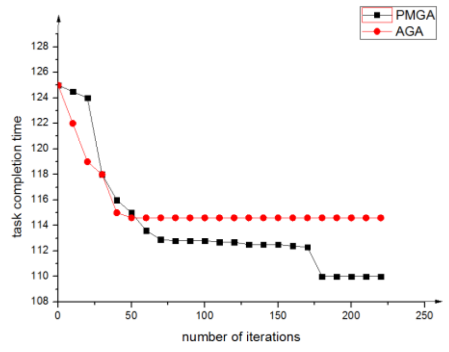

Figure 6. The Total Time for Task Completion

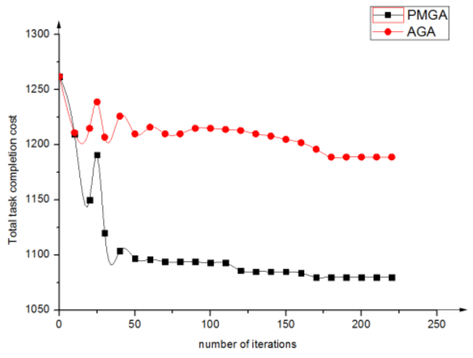

Figure 7. The Total Cost for Task Completion

Figure 8 shows the comparison of the average fitness value of the new population during the process of iteration. The horizontal axis represents the number of evolution times, and the vertical axis represents the average fitness value. AGA starts to premature convergence at about 60th generation, while PMGA continues to evolve, because when the average fitness of individuals is close to the maximum fitness, the crossover rate and mutation rate are not a zero value. Eventually, PMGA has a larger fitness value than AGA. The experimental results show that PMGA has a better ability to do the global searching.

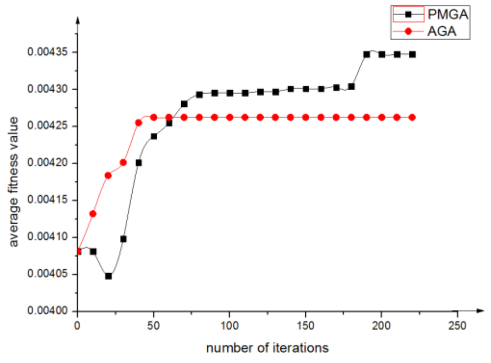

Figure 8. The Average Fitness Values of Population

In experiment 2, the number of convergence using PMGA and AGA for task scheduling are compared. The algorithm parameters are consistent with those in Table III. PMGA and AGA are compared for 50 times under the same iterations. The average value of each iterations is regarded as the ultimate result. Fig. 9 is the comparison chart of the number of convergence times using AGA and PMGA.

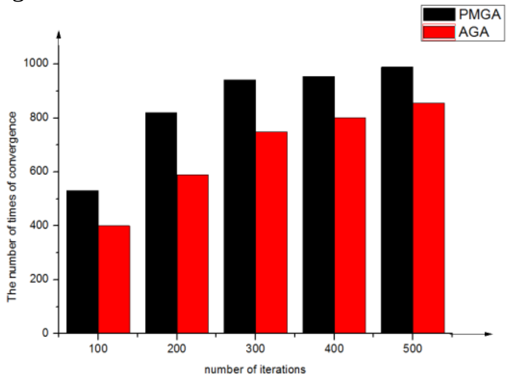

Figure 9. The Number of Convergence Using PMGA and AGA

In Fig. 9, the horizontal axis represents evolution times, the vertical axis represents the convergence times. It can be seen from the figure that the convergence number of PMGA is much more than AGA under the same number of iterations. The experimental results show that PMGA can improve the performance and robustness of the cloud rendering task scheduling.

V. CONCLUSION

This paper aims to reduce the total time and total cost of task scheduling in cloud rendering system, which applies probability model to genetic algorithm for rendering task scheduling. The algorithm evaluates the excellent degree of the new solution by establishing the fitness function considering the total time and total cost of the task. As a result, the better individuals will be returned, and the diversity of the population will be improved. The crossover rate function and mutation rate function are improved, which enhances the local search ability and avoids the occurring of local convergence. The experimental results show that PMGA reduces the total time and minimize the total cost. Thus, it is an effective algorithm for scheduling tasks in cloud rendering system.

In the future work, the Markov characteristic of PMGA will be studied in order to verify the scientific and rationality of scheduling algorithm using PMGA. Then, we will focus on the load balancing of dynamic task scheduling under cloud rendering system environment, and consider the influence of data distribution, service quality and other impact factors on the task scheduling results.

ACKNOWLEDGMENT

This paper is supported by National Natural Science Foundation of China under Grant No. 61502294, Natural Science Foundation of Shanghai under Grant No.15ZR1415200, CERNET Innovation Project under Grant No.NGII20160325, and Foundation of Science and Technology Commission of Shanghai Municipality under Grant No.14590500500.

REFERENCES

[1] Gupta R. Above the Clouds: A View of Cloud Computing[J]. Eecs Department University of California Berkeley, 2012, 53(4):50-58.

[2] Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, ‖ Software: Practice and Experience[J]. Software Practice & Experience, 2010, 41(1):23–50.

[3] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters.[J]. In Proceedings of Operating Systems Design and Implementation (OSDI, 2004, 51(1):107-113.

[4] Ullman J D. NP-complete scheduling problems. J Comput Syst Sci[J]. Journal of Computer & System Sciences, 1975, 10(3):384-393.

[5] Cao Q, Wei Z B, Gong W M. An Optimized Algorithm for Task Scheduling Based on Activity Based Costing in Cloud Computing[C]// International Conference on Bioinformatics and Biomedical Engineering. IEEE, 2009:1-3.

[6] Xu H, Kang F, Li L. Task scheduling strategy based on multi fitness particle swarm optimization in cloud computing[J]. Icic Express Letters, 2014, 8(11):3165-3170.

[7] Jie X U, Zhu J C, Ke L U. Task Scheduling Algorithm Based on Dual Fitness Genetic Annealing Algorithm in Cloud Computing Environment[J]. Dianzi Keji Daxue Xuebao/journal of the University of Electronic Science & Technology of China, 2013, 42(6):900-904.

[8] Liu C Y, Zou C M, Wu P. A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization in Cloud Computing[C]// International Symposium on Distributed Computing and Applications To Business, Engineering and Science. IEEE, 2014:68-72..

[9] Holland J H. Adoption in Natural and Artificial System[M]// Adaptation in natural and artificial systems. MIT Press, 1975:126–137.

[10] Srinivas M, Patnaik L M. Adaptive probabilities of crossover and mutation in genetic algorithms[J]. IEEE Transactions on Systems Man & Cybernetics, 1994, 24(4):656-667.

[11] Iosup A, Ostermann S, Yigitbasi N, et al. Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing[J]. IEEE Transactions on Parallel & Distributed Systems, 2011, 22(6):931-945.

[12] Ali S, Siegel H J, Maheswaran M, et al. Representing Task and Machine Heterogeneities for Heterogeneous[J]. Tamkang Journal of Science & Engineering, 2003, 3(3)

[13] Lipowski A, Lipowska D. Roulette-wheel selection via stochastic acceptance[J]. Physica A Statistical Mechanics & Its Applications, 2012, 391(6):2193-2196