An Improved Differential Evolution Task Scheduling Algorithm Based on Cloud Computing

Li Jingmei ,Liu Jia ,Wang Jiaxiang
College of computer Science and Technology
Harbin Engineering University
Harbin, China
e-mail: liujia4652@163.com

***Abstract*—It is a key issue to handle many tasks efficiently in cloud computing at low cost. For the cloud computing scheduling problem, to efficiently and reasonably assign a large number of tasks submitted by users to cloud computing resources, a task scheduling algorithm (IDE) based on improved differential evolution is proposed to consider both task completion time and cost dual objectives. The algorithm introduces an immune operator into the traditional differential evolution algorithm. According to the vaccination probability, the population is vaccinated during the iterative process to speed up the convergence of the algorithm. Introducing the judgment mechanism on the selection strategy can shorten the running time of the algorithm and effectively improve the shortcomings of the standard differential evolution algorithm with slow convergence speed. The original fixed scaling factor F becomes adaptive, which helps to increase the diversity of the population. The simulation experiment of the proposed algorithm is performed on the cloud computing platform CloudSim. Comparing the IDE algorithm with the traditional differential evolution algorithm, genetic algorithm and Min-Min algorithm, the results show that IDE algorithm task completion time is short, which improves the utilization of cloud computing resource pools, and the cost of computing resources in a similar period of time is low.**

Keywords-cloud computing; task scheduling; differential evolution; vaccination

I. INTRODUCTION

The concept of cloud computing has become a research hotspot since its introduction. As an emerging technology, cloud computing can be regarded as the development of distributed computing, parallel computing, and grid computing, and is realized in the commercial field. In order to meet the needs of users, we can provide high-performance, stable computing and storage-related services for users. Some key technologies in cloud computing have emerged.

The core of cloud computing is the virtual resource pool, including high-performance processors and stable storage structures. The existing cloud computing model framework is commonly used in Google's cloud computing platform and cloud computing network applications, IBM's "Blue Cloud" platform products and Amazon's elastic computing cloud [1].Although these platforms can meet the basic needs of a large number of users, the efficient processing of massive cloud tasks is still the key and difficult point in cloud computing. At present, in order to meet the needs of a large number of users, taking into account the time and cost of the user to submit the task, the researchers proposed a variety of cloud computing task scheduling algorithms, but these algorithms are still not efficient enough when dealing with massive tasks [2,3,4]. Therefore, it is of great significance to optimize the existing cloud computing task scheduling algorithm.

Based on the above background, in order to make resource utilization higher in cloud computing, this paper proposes a cloud computing task scheduling algorithm (IDE) based on differential evolution algorithm which considers both time and cost. In the initial stage of the evolution of the algorithm, immune operators were added, and the traditional differential evolution algorithm was improved by the vaccination method. At the same time, the adaptive scaling factor F was designed to solve the problems of premature convergence, slow convergence, and difficulty of parameter setting of traditional differential evolution algorithms. The main goal is to shorten the task completion time as much as possible, and to minimize the cost of the computing node while satisfying the needs of the majority of users, so as to achieve dual goals of time and cost optimization. At the same time, through the experimental simulation, the proposed algorithm is compared with the basic differential evolution algorithm and other scheduling algorithms to verify the effectiveness of the proposed algorithm.

II. PROBLEM DESCRIPTION OF CLOUD COMPUTING TASK SCHEDULING

Nowadays, most cloud computing platforms use the Map/Reduce programming model proposed by Google to perform parallel computing and processing on large-scale data sets. Map/Reduce is mainly divided into two phases. First, the received large-scale data submitted by the user is divided into many small sub-tasks, and these sub-tasks are then allocated to the resource set on the cloud server through the corresponding scheduling method. This stage of splitting is called the Map stage. After the computing resources have processed these sub-tasks and then integrated through the Reduce stage, the final result after processing is fed back to the user [5]. By dividing the task by such a model and simplifying the complex problems by means of synchronous parallel computing, the execution efficiency of tasks is improved [6,7,8].

The cloud computing platform mainly has two levels. The basic cloud computing platform model is shown in Figure 1. The requirements submitted by users classify

978-1-5386-7445-1/18/$31.00 ©2018 IEEE
DOI 10.1109/DCABES.2018.00018

tasks through the Map phase. The first level is to assign n tasks efficiently to m computing resources in a resource pool through a reasonable algorithm or strategy, and perform tasks on computing resources in a certain order of submission [9]. The second level of scheduling is the mapping between computing resources and physical machine entities. At this level, the main task accomplished is to optimize the virtual resource pool available to the cloud platform through the task scheduling strategy after the splitting of small child tasks. This article focuses on the scheduling problem at the first level.

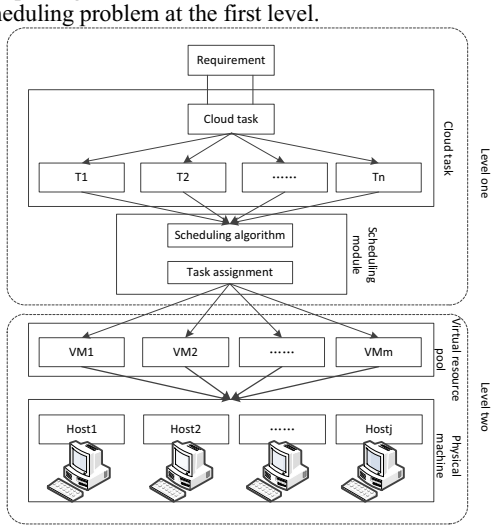

Fig 1 Cloud computing task scheduling model

Cloud computing needs to provide many users with different services at the same time. It needs to consider the response time of each user and also consider the cost of providing services for different users [10]. The existing algorithms only consider the shortest possible execution time of the task, and do not take into account the fact that some computing resource processing problems have a short time, but the cost is very high. Therefore, this paper proposes a task scheduling algorithm that reduces the task completion time while reducing the service cost of core computing resources.

This paper makes the following assumptions for cloud computing task scheduling:

(1) The completion time of a task on a computing resource node is known, the number of tasks is N, and the number of computing resources is M, then a matrix of N*M can be used to represent the completion time of a task on each computing resource, and is recorded as a time cost matrix.

(2) The cost required by each computing resource node per unit time is known. Then, an N*M matrix can be used to represent the cost per unit of time required for the task to execute on each computing resource, which is denoted as the cost overhead matrix.

(3) The number of cloud computing tasks is much larger than the number of computing resources, and the network delay and transmission time are ignored.

III. Improved Differential Evolution Task Scheduling Algorithm

A. Basic Differential Evolution Algorithm

Differential Evolution (DE) was first proposed by Storn and Price in 1995. It is mainly used to solve the real optimization problem [11]. It belongs to a group-based adaptive heuristic global search algorithm. Since DE's mutation operation is relatively simple, it can be widely applied in many fields such as data mining, neural network, and electromagnetics based on the rational modification of DE mapping with solving problems. On the discrete task scheduling problems such as cloud computing, the necessary improvements can also be effectively applied.

The core idea of DE is to perform a difference vector between two randomly generated populations, then generate a new intermediate generation after reasonable processing, and then use the greedy algorithm to select the best solution, and then iterate until the algorithm converges [12,13]. When the traditional DE algorithm cross-mutates, it is very likely that the global searching ability is poor because of the selection of the mutation operator. At the same time, the fine individuals produced by the new mutations may be exchanged during the crossover process, increasing the amount of calculations and iterations. To solve these drawbacks, this paper proposes an improved differential evolution algorithm that introduces immune operators to improve the convergence speed of the algorithm.

The general flow of the DE scheduling algorithm is as follows:

Step 1: Determine the parameters and variables in the algorithm based on the size of the problem.

Step 2: Randomly generate populations and initialize corresponding parameters.

Step 3: Calculate the initial population fitness value based on the goal of the problem.

Step 4: Determine if the resulting value satisfies the set termination condition or the maximum number of iterations. If yes, the algorithm is terminated, and conversely, execution continues.

Step 5: Individuals of the population undergo mutations and crossovers to produce intermediate populations.

Step 6: Calculate the fitness value of the middle population.

Step 7: According to the selection strategy, individuals satisfying conditions are selected among the intermediate population and the original population.

Step 8: Add one iteration, skip to step 4.

B. Population Coding

The traditional coding method in cloud computing task scheduling problem is based on the coding of computing resources. Each computing resource has a task sequence that needs to be executed.For example, there are 7 subtasks (N1, N2, N3, N4, N5, N6, N7) and 3 available resources (M1, M2, M3). The corresponding individual codes are [1, 3, 7, 0, 2, 5, 6, 0, 4] means that the N1, N3, and N7 tasks run on M1; N2, N5, and N6 tasks run on M2; N4 tasks run on M3; 0 indicates the identifier of the interval between computing resources symbol.

When using differential evolution algorithm for cross-mutation, this coding method has many disadvantages. For

example, when performing mathematical operations, it may exceed a reasonable range value. At the same time, it needs to determine whether it is an interval character, which increases the difficulty of parameter setting. Therefore, taking the above hypothesis as an example, the coding scheme corresponding to the encoding method used in this paper is [1,2,1,3,2,2,1], where the first one indicates that task N1 runs on M1; second A bit of 2 indicates that task N2 is running on M2; and so on, the seventh bit indicates that task N7 is running on M1. The corresponding decoding is: M1 runs on N1, N3, N7; M2 runs on N2, N5, N6; M3 runs on N4. This kind of coding method does not need to consider too many parameters in the cross-mutation. If there is a situation beyond a reasonable range, it can be rationalized by a unified method, which reduces the amount of calculation and makes it easier to obtain decoding results.

C. Initial Population Generation

Taking the initial population size as S, the number of tasks after all the tasks submitted by the user are properly handled is N, and the number of computing resources in the cloud computing is M, and the corresponding coding description according to the population initialization is: S sequences are randomly generated. Each code sequence is N in length, the range of each parameter in the sequence is [1, M].

D. Fitness Value Function

In order to make the improved algorithm suitable for cloud computing task scheduling problem, it is necessary to model the problem and design the corresponding fitness value function. The general algorithm only considers the execution time when it comes to cloud computing scheduling problems, but often ignores the cost requirements. Therefore, according to the characteristics of the cloud computing task scheduling problem and the needs of the majority of users, it is proposed that this algorithm is set to meet the dual-objective fitness value function of the calculation cost and task scheduling completion time.

According to the known time cost matrix, the completion time of each virtual resource execution task can be calculated, the set of cloud task assigned to the computing resource i is defined as Ri, and the task completion time of the computing resource is , then the completion time of all tasks is AT ,which is the maximum value of the task completion time for each computing resource, as shown in (1).

$$AT = \max\left(ET(i)\right) \tag{1}$$

Define the fitness value function that only considers the task completion time, as shown in (2).

$$F_{time} = AT \tag{2}$$

According to the cost-cost matrix, the cost of a task spent on different computing resources per unit time can be known as , and the completion time of the task on the computing resource is known, then the total cost of the task can be obtained by (3).

$$AC = \sum_{i=1}^{N} ET(i) * CN(i) \tag{3}$$

Because it is necessary to consider both time and cost goals, the cost function and time function vary greatly, and the difference may not be in the same order of magnitude. Therefore, the total cost of the task completion AC needs to be reasonably adjusted to FAC, as shown in (4).

$$FAC = \frac{AC}{10^{\left\lfloor \lg(\frac{AC}{AT}) \right\rfloor}} \tag{4}$$

It is intended to define a fitness value function that only considers costs, as shown in (5).

$$F_{\cos t} = FAC \tag{5}$$

Therefore, (2) and (5) define the comprehensive fitness value function that considers the task completion time and cost, as shown in (6).

$$Fitness = a \times F_{time} + b \times F_{\cos t} \tag{6}$$

In equation (6), a and b are used as the adjustment parameters, ranging from [0,1], and a+b=1, which is used to control and adjust the consideration of task execution time and cost. There are two extreme special values. When a=1 and b=0, the comprehensive fitness function becomes the fitness function that only considers time. When a=0 and b=1, the comprehensive fitness function becomes the fitness function that only considers the cost.

E. Mutation Operation

The basic operation of generating descendants using differential evolution algorithms is mutation and crossover. There are multiple mutation strategies in the differential evolution algorithm. The general form of the mutation strategy chosen in this paper is represented as DE/x/y/z, and DE represents the differential evolution algorithm. x denotes whether the reference individuals participating in the variation process and the vector recombination are randomly generated individuals or optimal individuals in the current population; y indicates the number of difference vectors involved in the reorganization; z indicates whether the reorganization adopts a binomial reorganization method or an index reorganization method. The variation used in this paper is shown in (7), namely DE/rand/1/bin.

$$V_i = N_1 + F \times (N_2 - N_3) \tag{7}$$

Among them, N1~N3 represent any randomly generated individuals in the current parent population. N1 is a randomly generated reference entity and F is the scaling factor in the formula, which ranges from [0, 2]. By formula (7), it can be known that a smaller F is favorable for accelerating the convergence speed of the algorithm, and a larger F is favorable for a global search, so that the algorithm jumps out of the local optimum to prevent over-exploitation. Therefore, this paper proposes to set the adaptive F, as shown in (8).

$$F = \begin{cases} \dfrac{Cg}{Mg} & \dfrac{1}{2}Mg > Cg \\ \dfrac{Cg}{Mg} + 1 & \dfrac{1}{2}Mg < Cg \end{cases} \tag{8}$$

Cg represents the current population iteration number, and Mg represents the maximum number of iterations of the algorithm.

Unreasonable values will be generated during the running of the algorithm. Therefore, it is necessary to rationalize the unreasonable value. In this paper, (9) is used to rationalize the unreasonable value generated by the calculation.

$$V_i = \left\lceil |V_i \bmod (M-1)| \right\rceil + 1 \quad (9)$$

After the intermediate population generated by the mutation operation, the disadvantage of slow convergence rate of the differential evolution algorithm is carried out according to Section 3.6 to increase the number of excellent individuals in the population. According to the fitness value, the middle population is selected from the vaccinated individuals and the mutated individuals, and the next generation population is selected according to the selection strategy.

F. Introducing Immune Operators

Immune algorithm is an algorithm that simulates human immunity and evolution mechanism. The vaccination mechanism of immune algorithm can effectively improve the convergence speed of the algorithm without affecting the convergence of the algorithm [14]. Therefore, reasonable addition of immune algorithm for vaccination can optimize the convergence speed of the original algorithm. Aiming at the disadvantage of the slow convergence rate of differential evolution algorithm, the immunization algorithm is introduced into the vaccination process to improve the convergence speed of the task scheduling algorithm in this paper. The extraction of superior individuals is performed at the initial population to generate vaccine pool sets.

1) Extracting Vaccine

The vaccine pool is a collection of pairs of cloud tasks and computing resources. The number of pairs is determined in advance by the characteristics of cloud tasks and computing resources. A reasonable threshold is set in advance, and the cloud task-calculation resource pair whose task completion time is less than the set threshold is extracted and added to the vaccine library. For a better understanding of the following give a simple example. When considering the completion time of the cloud task, the time consumption matrix for the scheduling of the five tasks onto the three computing resources may be as shown in FIG. 2. The rows of the matrix represent the cloud tasks and the columns represent the computing resources.

$$\begin{bmatrix} 12 & 3 & 2 \\ 1 & 6 & 3 \\ 5 & 8 & 11 \\ 13 & 2 & 4 \\ 6 & 4 & 11 \end{bmatrix}$$

Fig 2 Time Consumption Matrix

According to the time cost matrix, the preset vaccine pool can be {(2,1), (4,2), (1,3)}.Then X=[3,1,2,2,2] can be updated for an individual in the population after X=[1,3,2,1,2] vaccination. Similarly, according to the cost consumption matrix, the corresponding number of cloud task-computing resources pairs can be extracted as a vaccine library, and this will not be enumerated again.Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1”, even at the beginning of a sentence.

2) Vaccination

Although the introduction of the vaccination mechanism in the differential evolution algorithm can improve the convergence rate of the algorithm to a certain extent, if the vaccination is improper, the algorithm will lead to a local optimum when it is over-exploited. Therefore, in order to introduce the immune operator reasonably, it is necessary to define the inoculation probability formula, as shown in (10).

$$P = e^{-Cg} \qquad Cg \in [0, Mg] \quad (10)$$

The probability of inoculation was calculated from the number of iterations of the algorithm, and the population was vaccinated with probability. Since the extracted vaccine is a simple pair, it does not change the diversity of the population on a large scale. Reasonable use of vaccine can effectively improve the convergence speed of the algorithm, and effectively improve the algorithm's balanced mining and search capabilities. At the end of the iteration, the individual's fitness value is examined. When the number of iterations increases to a certain value, the probability of vaccination is very small. When the population is detected after the population iteration is small, the vaccination can be selected.

G. Selecting a Strategy

Most of the traditional differential evolution algorithm selection strategy is to choose greedy selection strategy, that is, individuals with better fitness values are selected in the middle population and the original population after the mutation crossover, and the good individuals are retained to form a new parent to enter the next iteration. Thus reciprocating the population continuously evolves towards a better target fitness value.

However, there are two shortcomings in the above selection strategy. First, the survival of the fittest during the iterative process is to select the best iteration to enter the new one at a time, which results in a poor global search capability leading to a local optimum. Second, good individuals generated after mutation operations may be destroyed by crossover operations and become individuals with poor fitness, which increases the number of iterations. In view of the above two insufficiency, in order to improve the search capability, we choose to use roulette to select and enter the next generation, which can appropriately improve the overall search scope and increase the diversity of the population. While improving the selection strategy, a judgment mechanism is added after the vaccination operation. That is, after the vaccination operation, it is judged whether the new individual's fitness value is better than the original individual. If the evolution of the population directly into the next generation is better than that of the original individual, and the fitness value of the original individual is not good, then the original algorithm is used for cross-operation and then the roulette strategy is used to select the next generation population.

H. The overall process of the algorithm

The flow of the improved differential evolution task scheduling algorithm that introduces immune operators is:

Step 1: According to the cloud computing problem model, initialize the parameters, make the current iteration number 0, set the maximum number of iterations and other variables.

Step 2: Randomly generate populations based on the set number of cloud tasks and resources.

Step 3: Calculate the fitness value of the initial population based on the dual-target demand set by the problem and extract the vaccine pool.

Step 4: Determine if the result meets the set termination condition or the maximum number of iterations. If yes, the algorithm is terminated, and conversely, execution continues.

Step 5: Randomly select individuals in the population to perform mutation operations to generate the mutated individual Vi, vaccinate Vi and select excellent individuals as intermediate populations. Determine whether the middle population is better than the current parent population. If yes, skip step 6; otherwise, perform the sequence.

Step 6: Crossing the current parent population and the intermediate population according to the crossover probability generates a new population.

Step 7: Select the next generation population according to the corresponding selection strategy.

Step 8: Add one iteration, skip to step 4.

The specific flow chart is shown in Figure 3.

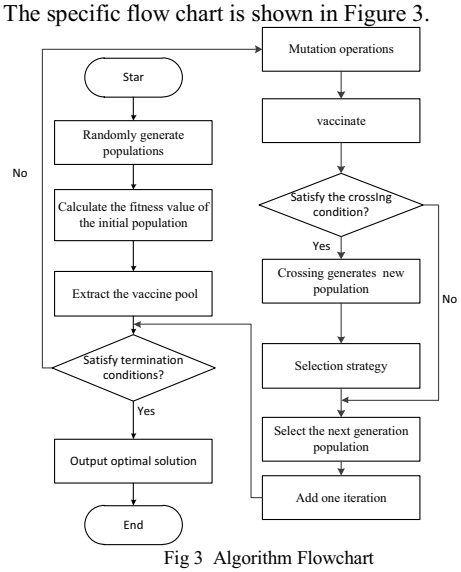

Fig 3 Algorithm Flowchart

IV. EXPERIMENTS AND RESULTS

In order to better evaluate and analyze the IDE proposed in this paper, cloud simulation platform CloudSim was used for simulation. CloudSim is the cloud computing simulation software announced by the Grid Lab and Gridbus project at the University of Melbourne, Australia. CloudSim can provide virtualization engines to establish and manage the virtualization services required by cloud computing. CloudSim's component tools are easy to use for open source architectures and meet cloud computing task scheduling requirements.

In practical problems, the tasks of cloud computing are often massive. When the number of processing tasks increases, the performance of the algorithm will be affected. According to the actual characteristics of cloud computing's computing resources, it is necessary to consider both time and cost. Therefore, this paper designs two sets of experiments. The first set of experiments keeps the number of computing resources constant and selects multiple sets of cloud tasks. When evaluating the number of computing resources is constant, as the cloud task grows, the performance of the differential evolution algorithm and other algorithms is improved. The experimental comparison algorithm used traditional differential evolution algorithm (DE), genetic algorithm (GA) [15] and Min-Min [16] algorithm. The second set of experiments keeps the number of cloud tasks and computing resources unchanged, and sets multiple sets of different a and b parameters. Evaluate the effectiveness of the differential evolution algorithm when the time and cost are taken into account while the cloud tasks and computing resources are unchanged. The experimental comparison uses parameter settings that only consider the time and only consider the cost.

Initial conditions of experiment 1 algorithm: Randomly generate six groups of cloud tasks with different numbers 100, 200, 400, 600, 800, and 1000. Since the computing resources have different processing time for different tasks, the latter task needs to include the previous tasks. The calculation resource M is set to 10, and when comparing the algorithms, the task completion time and the task total cost are respectively considered.

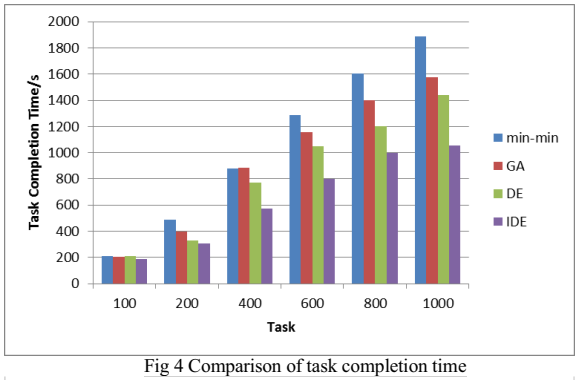

Fig 4 Comparison of task completion time

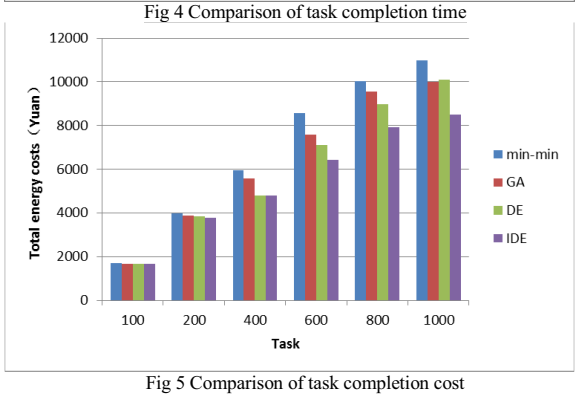

Fig 5 Comparison of task completion cost

Analysis of experimental results: It can be seen from Figure 4 and Figure 5. When the number of tasks is small, the performance of the algorithm is not much different. However, it can be seen that when the number of tasks is large, the IDE algorithm is significantly shorter than other algorithms and the cost is low. It shows that the IDE algorithm proposed in this paper has good performance in handling massive cloud tasks.

Initial conditions of experiment 2 algorithm: Randomly generate 100 tasks, computing resource M is set to 10, and the maximum number of iterations is set to 100. Set (a = 1,

b = 0); (a = 0, b = 1); (a = 0.5, b = 0.5) three different pairs a, b values.

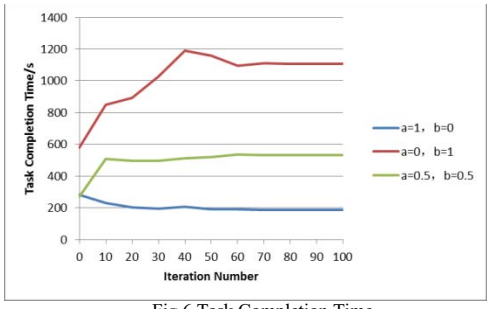

Fig 6 Task Completion Time

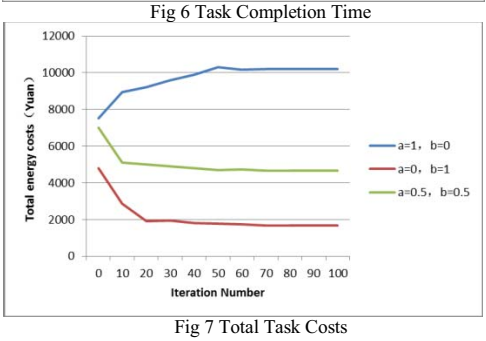

Fig 7 Total Task Costs

Analysis of experimental results: It can be seen from Fig. 6 and Fig. 7 that the performance of the algorithm is conflicting when considering only time and only cost. The IDE algorithm proposed in this paper, which considers both time and cost, is suitable for a wide range of users who balance time and cost. When dealing with practical problems, the IDE algorithm can adjust the values of parameters a, b to meet the needs of different users.

V. CONCLUSION

According to the characteristics of cloud computing task scheduling problem, taking into account the dual goals of time and cost, a dual-objective improved differential evolution algorithm is proposed.A simple coding method is used to model the cloud computing task scheduling, and a judgment mechanism is introduced before the crossover operation. At the same time, the immune operator is added after the mutation crossover, which accelerates the convergence speed of the algorithm better and adopts an adaptive scaling factor F. Increased population diversity prevents the algorithm from falling into a local optimal solution. Through simulation experiments, the proposed algorithm is compared with the traditional cloud computing task scheduling algorithm. The algorithm proposed in this paper has a faster convergence speed and lower cost. However, there are still some shortcomings in the research of this algorithm for cloud computing task scheduling. The next step will be to consider that during the operation of the algorithm, network delays and interruptions need to be added to the task memory function in order to better meet the cloud computing task scheduling requirements.

ACKNOWLEDGMENT

This work was supported by National Key Research and Development Plan of China under Grant No.2016YFB0801004.

REFERENCES

[1] Armbrust, Michael, Fox, et al. Above the Clouds: A Berkeley View of Cloud Computing[J]. Eecs Department University of California Berkeley, 2009, 53(4):50-58.

[2] Abdullahi M, Ngadi M A, Abdulhamid S M. Symbiotic Organism Search optimization based task scheduling in cloud computing environment[J]. Future Generation Computer Systems, 2016, 56:640-650.

[3] Dutta D, Joshi R C. A genetic: algorithm approach to cost-based multi-QoS job scheduling in cloud computing environment[C]// International Conference & Workshop on Emerging Trends in Technology. ACM, 2011:422-427.

[4] Razaque A, Vennapusa N R, Soni N, et al. Task scheduling in Cloud computing[C]// IEEE Long Island Systems, Applications and Technology Conference. IEEE, 2016:1-5.

[5] Chang Y J, Chen C C, Chen C L, et al. A de novo next generation genomic sequence assembler based on string graph and MapReduce cloud computing framework.[J]. Bmc Genomics, 2012, 13 Suppl 7(S7):S28.

[6] Panda S K, Jana P K. Efficient task scheduling algorithms for heterogeneous multi-cloud environment[J]. The Journal of Supercomputing, 2015, 71(4):1505-1533.

[7] Lin R, Li Q. Task scheduling algorithm based on Pre-allocation strategy in cloud computing[C]// IEEE International Conference on Cloud Computing and Big Data Analysis. IEEE, 2016:227-232.

[8] Singh S, Chana I. QRSF: QoS-aware resource scheduling framework in cloud computing[J]. Journal of Supercomputing, 2015, 71(1):241-292.

[9] 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.

[10] Jiehui J U, Bao W Z, Wang Z Y, et al. Research for the Task Scheduling Algorithm Optimization based on Hybrid PSO and ACO for Cloud Computing[J]. International Journal of Grid & Distributed Computing, 2014, 7(25):217-218.

[11] Price K, Price K. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces[M]. Kluwer Academic Publishers, 1997.

[12] Ge J, He Q, Fang Y. Cloud computing task scheduling strategy based on improved differential evolution algorithm[C]// International Conference on Computer-aided Design. AIP Publishing LLC, 2017:1-35.

[13] Zheng Z, Xie K, He S, et al. A Multi-objective Optimization Scheduling Method Based on the Improved Differential Evolution Algorithm in Cloud Computing[C]// International Conference on Cloud Computing and Security. Springer, Cham, 2017:226-238.

[14] Jiao L, Wang L. A novel genetic algorithm based on immunity[J]. Systems Man & Cybernetics Part A Systems & Humans IEEE Transactions on, 2000, 30(5):552-561.

[15] Goldberg D E. Genetic Algorithm in Search Optimization and Machine Learning[J]. Addison Wesley, 1989, xiii(7):2104–2116.

[16] Singh S, Singh V. A Genetic based Improved Load Balanced Min-Min Task Scheduling Algorithm for Load Balancing in Cloud Computing[C]// International Conference on Computational Intelligence and Communication Networks. 2016.