Journal of Electronic Research and Application

Research Article

Research on Resource Scheduling of Cloud Computing Based on Improved Genetic Algorithm

Juanzhi Zhang, Fuli Xiong, Zhongxing Duan

School of Information and Control Engineering, Xi'an University of Architecture and Technology, Xian 710000, China

Funding: National Natural Science Foundation of China (61473216); Shaanxi Provincial Fund (2015JM6337)

Abstract: In order to solve the problem that the resource scheduling time of cloud data center is too long, this paper analyzes the two-stage resource scheduling mechanism of cloud data center. Aiming at the minimum task completion time, a mathematical model of resource scheduling in cloud data center is established. The two-stage resource scheduling optimization simulation is realized by using the conventional genetic algorithm. On the technology of the conventional genetic algorithm, an adaptive transformation operator is designed to improve the crossover and mutation of the genetic algorithm. The experimental results show that the improved genetic algorithm can significantly reduce the total completion time of the task, and has good convergence and global optimization ability.

Keywords: Cloud computing; resource scheduling; Genetic algorithms; Adaptive improvement operator

Publication date: March, 2020

Publication online: 31 March, 2020

Corresponding author: Juanzhi Zhang, wxcgxy@163.com

1 Introduction

After entering the 21st century, with the development of global Internet technology, the development of information technology, and the rise of Internet of Things and 5G technology, all kinds of data grow exponentially, which has higher requirements for Cloud Data Center (CDC) in high-speed network and high-performance computing system[1-2]. In the development of cloud computing, resource allocation mechanism is always an urgent problem to be solved in data-center efficiency optimization, and it is also a research hotspot in the field of cloud computing[3-4]. Therefore, it is necessary for CDC to schedule highly concurrent tasks reasonably and efficiently. In this case, the hybrid evolution of distributed computing and grid computing has formed a more mature cloud computing service model and business model[5-6]. At the same time, the resource scheduling and optimization of Cloud Data Center based on intelligent algorithms such as genetic algorithm (GA), particle swarm optimization (PSO) has also become a research hotspot[7]. For example, Mathiyalagan *et al*[8]. introduced ant colony optimization algorithm for grid computing programming to develop the ability of the system; Srikanth [9] introduced ant colony optimization into task scheduling for generating programs; keshanchi *et al*[10]. used priority queue to schedule tasks in cloud environment to improve genetic algorithm (GA) and adopt the elite technology with abnormal selection to avoid premature convergence.

Based on the process of task allocation, a general two-stage task-scheduling model for green cloud data center is established. It solves the general two-stage task-scheduling problem. Moreover, it also studies how to transform the task scheduling problem of cloud computing data center into a two-stage scheduling problem, and establishes a task scheduling mathematical model aiming at the completion time.

2 Resource scheduling model

The goal of task scheduling is to minimize the completion time. It involves two problems of allocation and sequencing. These problems are described as follows:

$$\begin{cases} M=\{M_1,\ M_2,\cdots,M_i,\cdots,M_n\} \\ J\ =\{J_1,\ J_2,\cdots,J_j,\cdots,J_m\} \end{cases} \quad (1)$$

Where M contains n identical parallel servers, J is a set of m tasks processed in two-level scheduling.

$$J_j=\{T_j^{(1)},T_j^{(2)}\},$$
$$(T_j^{(1)},T_j^{(2)}\geq 0;\ \text{for } j=1,2,\ldots,m) \quad (2)$$

The whole plan is defined as s, which includes two parts: allocation and sequencing. In order to represent the distribution relationship in s, an allocation matrix A=[A_{ij}] is introduced to represent the mapping from J to M:

$$A_{ij}=\begin{cases} 1, J_j \text{ was assigned to } M_i \\ 0, J_j \text{ was not assigned to } M_i \end{cases} \quad (3)$$

The assignment matrix can represent all possible mappings, and each mapping is unique. Because each task can only be processed by one server, it is shown as follows:

$$\sum_{i=1}^{n} A_{ij}=1 \quad (4)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij}=m \quad (5)$$

For example, the following is an assignment matrix of 15 tasks assigned to six servers. Each row is for a server, and each column is for a task.

$$D=\begin{bmatrix} 0&1&0&0&1&0&0&0&0&0&0&0&1&0&0 \\ 1&0&0&0&0&0&1&0&0&1&0&0&0&0&1 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0&0&0&1&0&0&0&0 \end{bmatrix} \quad (6)$$

In fact, equations (4) and (5) mean that only one term in each column has a value of 1. $C(s)$ is the maximum value of the time to complete task execution and transmission, as follows:

$$\bar{C}(s)=max\{C_i(s)\}, i=1,2,\cdots,n \quad (7)$$

The goal of task scheduling is to find the scheduling that minimizes $C(s)$. For $J_j(j=1, 2, ..., m)$, suppose $T_{bj}^{(1)}$ and $T_{bj}^{(2)}$ as the start time of execution and transmission, and suppose $T_{ej}^{(1)}$ and $T_{ej}^{(2)}$ as the end time respectively, which are restricted by the following relations [11]:

$$t_{ej}^{(1)}=t_{bj}^{(1)}+T_j^{(1)} \quad (8)$$

$$t_{ej}^{(2)}=t_{bj}^{(2)}+T_j^{(2)} \quad (9)$$

Now, the scheduling optimization problem is:

$$min\bar{C}(s) \quad (10)$$

$$t_{bj}^{(2)}\geq t_{ej}^{(1)} \quad (11)$$

$$a_{ij}t_{ej}^{(1)}\leq a_{ik}t_{bk}^{(1)} \text{ or } a_{ik}t_{ek}^{(1)}\leq a_{ij}t_{bj}^{(1)} \quad (12)$$

$$a_{ik}t_{ek}^{(2)}\leq a_{ij}t_{bj}^{(2)} \text{ or } a_{ik}t_{ek}^{(2)}\leq a_{ij}t_{bj}^{(2)} \quad (13)$$

$$i=1,2\cdots,n;\ j,k=1\ 2\cdots m;\ j\neq k$$

In this case, (11) means that for each job, the transfer starts only after the execution is completed; constraints (12) and (13) ensure that the server executes and transfers. For all tasks assigned to the same server, for example, the execution (or transfer) of (J_k) in a new workflow starts only after the f execution (or transfer), or the previous work (J_j) has been completed. The analysis requires two assumptions: all servers are the same, have the same network knowledge and the same opportunity to receive requests [11].

3 Algorithm design of resource scheduling

3.1 Genetic algorithm design

Step1: Encoding

Scheduling optimization involves two problems: allocation and sequencing. This article only deals with allocation. Distribution matrix A=[A_{ij}], specifies the mapping of J to M. For the problems considered in this study, there is only one term with a value of 1 in each column. Since each task is assigned to only one server, we can use a one-dimensional matrix that lists the number of servers to which the task is assigned. In the example of distribution matrix (6), J_1 is assigned to M_2, so the first term of D^ is 2; J_2 is assigned to M_1, so the second term is 1; J_3 is assigned to M_5, so the third term is 5; And so on.

Step2: Decoding

For each physical machine resource M_i, it first extracts the terms in I with a value of i, and then generates the corresponding task list, so as to construct a task list L_i.

$$\boldsymbol{L}=[3\ \ 1\ \ 4\ \ 5\ \ 1\ \ 6\ \ 3\ \ 2\ \ 5\ \ 3\ \ 6\ \ 2\ \ 6\ \ 4\ \ 3] \quad (14)$$

The task list of M_2 first extracts all terms (i_1、i_7、i_{10}、i_{15}) with a value of 3, and then lists the corresponding tasks: L_3={J_1、J_7、J_{10}、J_{15}}, and so on.

Step 3: Initialization design of population

The initial population size is determined according to the number of tasks and the population size to be assigned, and the feasible solution of the specified number is generated according to the model constraints;

Step 4: Fitness function design

According to the objective function defined by the model, the congestion degree and energy consumption of each chromosome are calculated and weighted by preset weight. In this way, the objective function value of each chromosome can be obtained, and the reciprocal of the objective function value is taken as the fitness function value;

Step 5: Selection operator design

Roulette selection is one of the most frequently used selection operators in genetic algorithm. The probability of individuals being selected is directly proportional to their fitness value, which fully reflects the characteristics of genetic algorithm. Therefore, this paper uses roulette selection operator and the specific steps are as follows:

(1) calculate the fitness value $f(i=1, 2, ..., N)$ of each individual in the population , N is the population size;

(2) calculate the probability that each individual is passed on to the next generation;

$$P(x_i)=\frac{f(x_i)}{\sum_{j=1}^{N} f(x_i)} \tag{15}$$

(3) The cumulative probability of each individual is calculated;

$$q_i=\sum_{j=1}^{i} p(x_i) \tag{16}$$

q_i is called as cumulative probability of chromosome x_i $(i=1, 2, ..., n)$

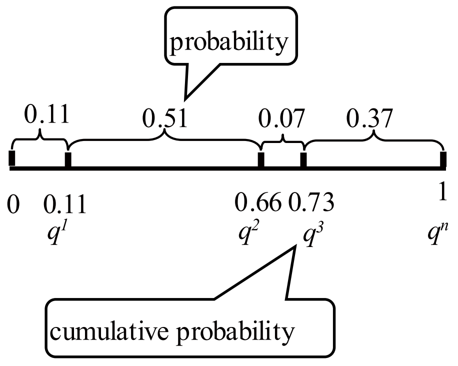

Figure 1. Schematic diagram of roulette selection

(4)A pseudorandom number r with uniform distribution is generated in the interval of [0,1];

(5)If $r<q^{[1]}$, then select individual 1, otherwise, select individual k, so that $q^{[k-1]}<r\leq q^{[k]}$ is established;

(6)N cumulative probability of repetition (4) and (5)

Step 6: Crossover operator design

The strategy of two-point crossing in chromosome is adopted, and the crossing operation is carried out according to the set crossing probability. Any two genes are selected in the chromosome, i.e. any two tasks in the task assignment sequence. Every aspect of the ability for each node after gene exchange is judged to meet the model constraints. If the constraint is satisfied, crossover operation will be performed; if not, crossover operation will not be performed;

Step7: Mutation operator design

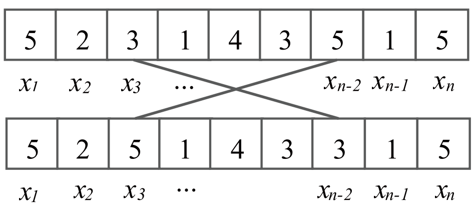
5 2 3 1 4 3 5 1 5
x_1 x_2 x_3 … x_{n-2} x_{n-1} x_n
5 2 5 1 4 3 3 1 5
x_1 x_2 x_3 … x_{n-2} x_{n-1} x_n

Figure 2. chematic diagram of double-point crossover operator

The strategy of single point crossing in chromosome is adopted, and the mutation operation is carried out according to the set mutation probability. Any gene is selected in the chromosome, i.e. any task in task assignment sequence, and it is mutated, i.e. randomly assign it to any resource node meeting its requirements;

3.2 Adaptive-genetic improved algorithm

In the early stage of genetic algorithm, if the crossover and mutation operations are too low, the probability that the population evolves into a better individual will be greatly reduced, and the convergence speed of the algorithm will be affected; in the late stage of genetic algorithm, if the crossover and mutation operations are too much, it is easy to destroy the structure of the good population and lead to the local convergence of the algorithm. It can be seen that the process of crossover and mutation in genetic algorithm directly affects the convergence speed and global optimization ability. The self-adaption of crossover process and mutation process can be divided into two levels: the self-adaption of crossover probability, mutation probability value, crossover operator and mutation operator. For these two levels, this paper proposes an adaptive adjustment strategy, which is shown as follows:

$$P_c=\begin{cases}\frac{P_{c\max}+P_{c\min}}{2}+\frac{(P_{c\max}-P_{c\min})}{2}\tanh(\frac{F_c-F_{avg}}{F_{\max}-F_{avg}})\pi & F_c\geq F_{avg}\\ P_{m\max}, F_m<F_{avg}\end{cases} \tag{17}$$

$$P_m=\begin{cases}\frac{P_{m\max}+P_{m\min}}{2}+\frac{(P_{m\max}-P_{m\min})}{2}\tanh(\frac{F_m-F_{avg}}{F_{\max}-F_{avg}}\pi\ ) \ , & F_m\geq F_{avg}\\ P_{m\max}, F_m<F_{avg}\end{cases} \tag{18}$$

$$Pc\_oper=\begin{cases}Crossover_{Single\text{-}Point}, F_c\geq F_{avg}\\ Crossover_{Partial\text{-}Mapped}, F_c<F_{avg}\end{cases} \tag{19}$$

$$Pm\_oper=\begin{cases}Mutation_{Single\text{-}Point}, F_m\geq F_{avg}\\ Mutation_{Multiple\text{-}Point}, F_m<F_{avg}\end{cases} \tag{20}$$

In formula (15), (16), (17) and (18), F_{max} is the maximum fitness of the population, F_{avg} is the average fitness of the population, F_c is the larger in the two individual fitness performing crossover operation, and F_m is the fitness of individuals performing mutation operation. Pc_{max} 、 Pc_{min} 、 Pm_{max} 、 Pm_{min} is the upper and lower limits of crossover probability and mutation probability, respectively. $Crossover_{Single\text{-}Point}$ is the single-point crossover operator, $Crossover_{Partial-Mapped}$ is the local crossover operator, $Mutation_{Single\text{-}Point}$ is the multi-point mutation operator, and $Mutation_{Multiple\text{-}Point}$ is the multi-point mutation operator.

In order to make the probability of crossover and mutation change smoothly at the pole, this paper uses hyperbolic tangent function to construct the adaptive law of crossover and mutation probability.

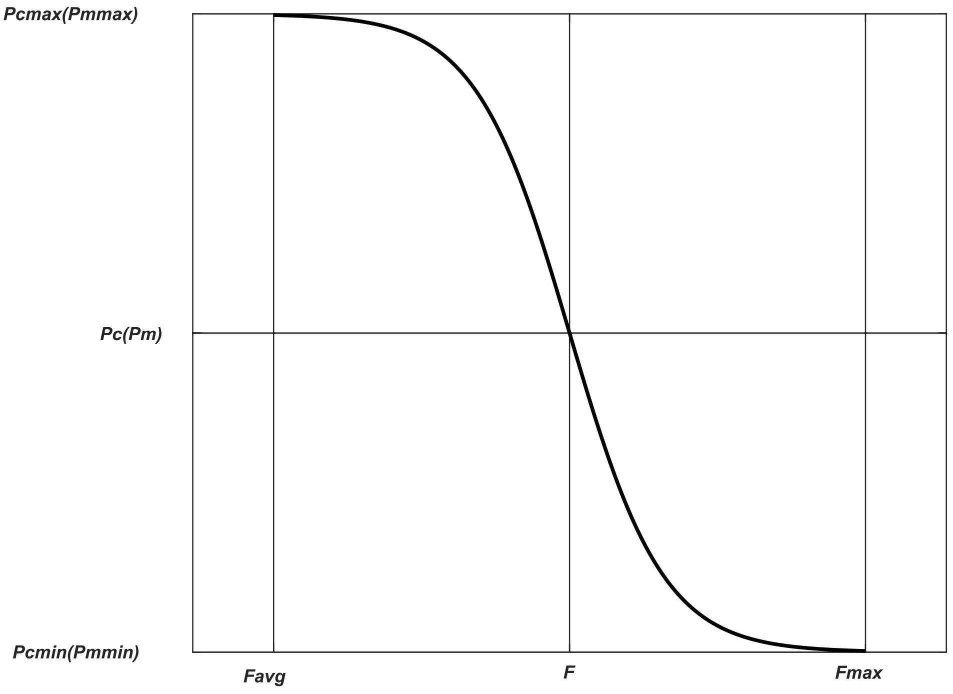

Figure 3. Adaptive law curve for genetic algorithm crossover and mutation

As shown in Figure 3, the crossover probability, mutation probability value, crossover operator and mutation operator of genetic algorithm are adjusted adaptively. When the fitness of crossover and mutation individuals tends to the maximum, the adaptive law can improve the probability of individual crossover and mutation, that is, increase the probability of evolution to a better individual; when the fitness of crossover and mutation individuals tends to the minimum, the adaptive law can reduce the probability of individual crossover and mutation, that is, keep the genes of the better individuals as much as possible.

4 Experimental analysis

4.1 Setting of task execution time(Table 1)

Table 1. Timetable for mandate implementation

Times(*s*)	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
$T_j^{(1)}$	92	54	19	78	80	59	114	92
$T_j^{(2)}$	103	48	36	88	72	39	84	79
Times(*s*)	J_9	J_{10}	J_{11}	J_{12}	J_{13}	J_{14}	J_{15}	
$T_j^{(1)}$	42	38	37	76	29	88	51	
$T_j^{(2)}$	54	45	58	65	34	80	43	

4.2 Algorithm setting

For meta-heuristics algorithm, the selected parameters have a great influence on the experimental results. GA algorithm has three parameters: crossover probability (P_c), crossover probability (P_m) and population number (Pop), and the adaptive genetic algorithm (AGA) proposed in this paper has the same parameters as GA algorithm;

In this paper, $L_9(3^4)$direct intersection method is adopted to design GA algorithm and AGA-algorithm parameter experiment method. They respectively obtain the optimal-algorithm parameter combination as shown in Table 2:

Table 2. Algorithm parameter table

GA		AGA	
parameter	*value*	*parameter*	*value*
Pc	0.8	*Pc*	0.9
Pm	0.1	*Pm*	0.2
Pop	50	*Pop*	50

4.3 Algorithm analysis and comparison

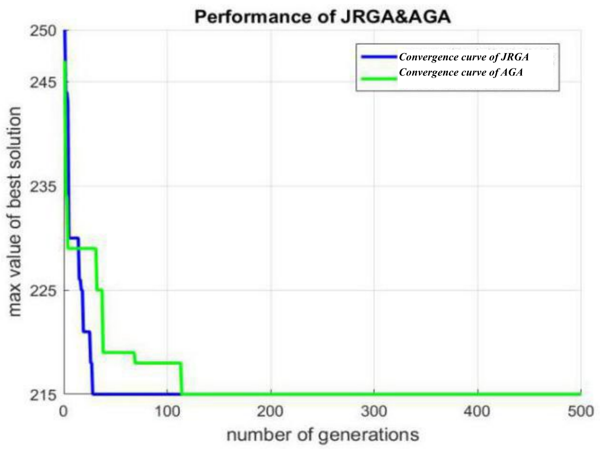

Figure 4. Gorithm comparison under resource allocation 1

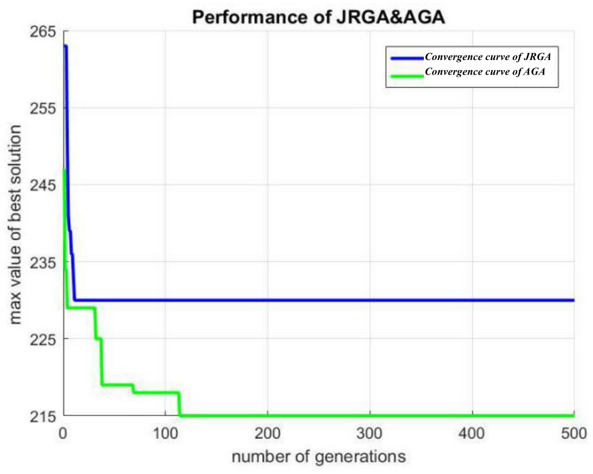

Figure 5. Comparison of under Resource Allocation 2

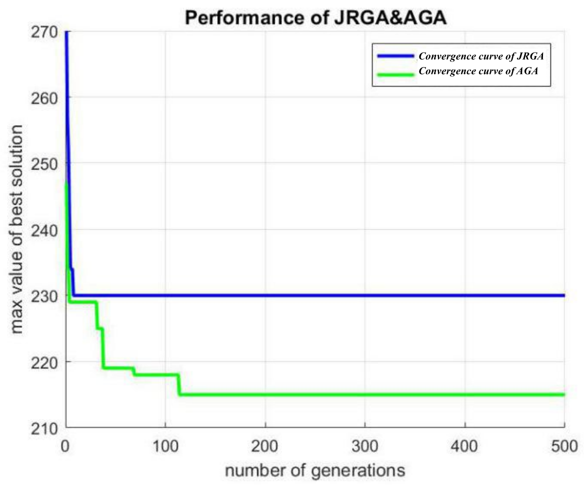

Figure 6. Algorithm Comparison under Resource Allocation 3

Figure4, Figure5 and Figure6 are the comparison charts of simulation convergence curves based on conventional genetic algorithm and adaptive improved genetic algorithm when the number of servers is 6 and the number of tasks is 15, 30 and 45 respectively.

It can be seen from the figure that under configuration conditions of different resource and task corresponding, the curve of adaptive improved genetic algorithm basically needs to be iterated about 100 times and then tends to be stable. The number of iterations is a little rising in comparison with that of conventional genetic algorithm (10 times). However, the minimum completion time of the whole resource scheduling is 215s. Compared with the minimum completion time (230s) based on Johnson genetic algorithm, it is improved by 15s, and the effect is obvious; the adaptive improved genetic algorithm is also better in stability, and the minimum completion time does not increase dramatically due to the increase of the number of tasks. Although the convergence speed of the curve is reduced, the adaptive improvement based on the conventional genetic algorithm meets the expected result of decreasing the minimum completion time.

5 Conclusion

Based on the process of task allocation, this paper establishes a task-scheduling mathematical model aiming at the execution time of task queue. Because the conventional GA algorithm is easy to fall into the local search, difficult to realize the global optimization and the task execution time is too long, this paper proposes an adaptive modification operator based on tangent hyperbolic function to modify the conventional GA algorithm. Experiments show that the adaptive improved genetic algorithm can significantly reduce the total completion time of task queue, and has good convergence and global optimization ability.

References

[1] Zhu X. Research on key technologies of energy consumption and resource scheduling optimization in green cloud computing data center[J]. Wireless Internet Technology, 2019,16 (11): 118-119.

[2] Luo HL. Algorithm research on cloud computing resource scheduling based on Wi-Fi and Web [J]. Computer Measurement and Control, 2017, 25 (12): 150-152 + 176.

[3] Zhang L, Shang YL. Design and implementation of resource scheduling system in cloud computing environment[J]. Computer Measurement and Control, 2017, 25 (1): 131-134.

[4] Hao L. Algorithm research on cloud computing resource scheduling for energy consumption optimization . Harbin Industrial University. Doctoral Dissertation.

[5] Huang WJ, Guo F. Cloud computing multi-objective task scheduling based on fireworks algorithm[J]. Computer Application Research, 2017, 34 (6): 1718-1720 + 1731.

[6] Zhong M. Research and improvement of cloud computing resource scheduling[D]. Jiangxi University of Science and Technology. Master Dissertation.

[7] Cai XL, Qian C. Cloud resource scheduling strategy based on improved particle swarm algorithm [J]. Microelectronics and Computer, 2018,35 (6): 28-30.

[8] Mathiyalagan P, Dhepthie UR, Sivanandam SN. Grid scheduling using enhancedant colony algorithm. ICTACT Journal on Soft Computing. 2 (2010).

[9] Srikanth U, Maheswari VU, Shanthi P, Siromoney A. Tasks scheduling using Ant Colony Optimization[J]. Journal of Computer Science, 2012, 8: 1314-1320.

[10] Keshanchi B, Souri A, Navimipour NJ. Animproved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing[J]. Journal of Systems and Software, 2017, 124: 1-21.

[11] Xiong YH, Huang SZ, Wu M. A Johnson's-Rule-Based Genetic Algorithm for Two-Stage-Task Scheduling Problem in Data-Centers of Cloud Computing[J]. Journal of Latex Class Files, 13(9): 2014.