

PAPER • OPEN ACCESS

Large-scale resource scheduling method using improved genetic algorithm combined with secondary coding in cloud computing environment

To cite this article: Gu Nan nan *et al* 2019 *IOP Conf. Ser.: Mater. Sci. Eng.* **563** 052016

View the [article online](#) for updates and enhancements.

Large-scale resource scheduling method using improved genetic algorithm combined with secondary coding in cloud computing environment

Gu Nan nan*, Yao Pei yang and Jiao Zhi qiang

School of Information and Navigation, Air Force Engineering University, Xi'an, Shaanxi 710038, China

E-mail: 2802896204@qq.com

Abstract. This paper proposes a large-scale resource scheduling method based on improved genetic algorithm combined with secondary coding. This method is used to solve the problem that traditional genetic algorithm can not meet the resource scheduling problem in large-scale cloud computing environment under multi-user. In the selective replication phase, a dual fitness function based on minimum task completion time and matching degree is used to screen the populations by double criteria. Next, the cross-mutation probability of the algorithm is adaptively optimized, and its adaptive ability is further improved, which ensures that the algorithm converges to the optimal solution as soon as possible. Finally, the improved genetic algorithm (IGA) is analyzed on the CloudSim platform. It shows that the improved genetic algorithm can be well applied to large-scale resource scheduling, and the result is better than the comparison algorithm.

1.Introduction

With the continuous rise of virtual technology, the previous stand-alone mode has been unable to meet the increasingly complex needs of users. As an important part of virtual technology, cloud computing has been applied to various fields, and cloud computing platforms can be based on constraints imposed by different users. Correspondingly, massive data and large-scale tasks are reasonably allocated to each resource for processing, thereby optimizing task processing. Cloud computing can be divided into management and service from its specific structure. From the service point of view, it is mainly to ensure the user's demand for various services in the cloud, which are specifically divided into the following three [1]. Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) [2].

2.Cloud resource scheduling related model

2.1 Cloud computing platform resource model

Considering the heterogeneity of cloud computing platform resources, that is, different cloud platforms have different capability focuses. Some cloud platform resources have strong computing power but small bandwidth, while some cloud platform resources have large memory but weak computing power. The ability is weak and so on. Based on the above characteristics, we can use the vector $abi = [C, Me, BW]$ to represent the resource capability vector of the cloud platform resource, where, C represents the computing power of the CPU (MIPS, the ability to process millions of



instructions per second), Me represents the size of the memory (MB), BW represents Total bandwidth (Kb/s).

2.2 Quality of Service Model

In the study of this paper, only two service quality evaluation parameters, computational power and bandwidth capability, are considered. Based on these two evaluation parameters, we can classify the received tasks into two categories: the first category is the computing capability task, and the second category is the bandwidth capability task, so the preferences of different user submitted tasks for different cloud platform resources. The degree can be expressed by the vector $pre_i = [pre_{cpu}, pre_{memory}, pre_{bandwidth}]$. Among them, pre_{cpu} , pre_{memory} and $pre_{bandwidth}$ represent the user's preference for CPU, Memory, and BandWidth. According to the literature [3] and [4], the preference vectors of the two types of tasks in this paper are given, as shown in Table 1:

Table 1: Preference Vectors

Task type	Preference vector
1	[0.6,0.3,0.1]
2	[0.3,0.2,0.5]

2.3 Task Requirements Model

Based on the quality of service model in 2.2, task priorities can be divided into two types: the first is a task with high computing power and memory requirements, that is, the computing capability task has priority execution rights, and the second is bandwidth capability type Task, which does not require high-performance computing and storage capabilities, so it has lower execution rights than the first task. Therefore, the task requirement vector can be expressed as $task = [ID, TP, CL, In, Out, Pre, AB, Pri]$, where ID represents the task ID, TP represents the task type (TP=1 is a computing capability task, TP=2 is a bandwidth capability task), CL represents the task instruction length, and In represents the input size. Out represents the output size, Pre represents the preference vector of different users for the cloud platform resource parameters, AB represents the actual required bandwidth of the task, and Pri represents the execution priority (pri=1 is the high priority, pri=2 is the low priority). For example, the task requirement vector is (3, 2, 100000, 1500, 500, 250, 2), where 3 represents the ID of the task, the first 2 represents the task category, 100000 represents the task instruction length, and 1500 represents the input size, 500. Represents the output size, 250 represents the actual bandwidth requirement, and the second 2 represents the task priority.

3.Improved genetic algorithm

3.1 Task—Second Real Encoding of Platform Resources

In view of the above problems, this paper improves the traditional coding method and divides it into primary coding and secondary coding .Set the set of tasks to m to $ID = [1, 2, 3, 4 \dots m]$, and the set of platform resources to n to $ID = [1, 2, 3, 4 \dots n]$. The specific flow is as follows:

$$[2, 2, 1, 4, 4, 6, 6, 6 \dots n, n, n]$$

(1) Initial coding: If a chromosomal gene string is randomly generated as $[2, 2, 1, 4, 4, 6, 6, 6 \dots n, n, n]$, where m is the total number of tasks, and n is the number of platform resources. It indicates that the task {1, 2} is assigned to the platform resource 2 execution, the task {3} is assigned to the platform resource 1 execution, and the task {m-2, m-1, m} is assigned to the platform resource n for execution.

(2) Secondary coding: Secondary encoding of the above chromosome, the chromosome length is n, and the specific value of each gene position is the total number of tasks assigned to the corresponding resource node and the task ID, as shown in Table 2:

Table 2: Secondary coding table

Resource ID	1	2	3	...	n
Number of tasks	1	2	5	...	3
Task ID	{3}	{1,2}	{8,15,25,31,56}	...	{m-2,m-1,m}

The above is the specific improved coding method. Due to the uncertainty and diversity of the population in the genetic algorithm, the initial population adopts a randomly generated method, and the population size is popsize.

3.2 double fitness function

3.2.1 Fitness function for minimum task completion time

When a cloud platform in a class processes a certain type of task, its individual task time can be expressed by:

$$TimeCost_i = \frac{CL}{MIPS} \quad (1)$$

Among them, CL represents the length of the task, and MIPS is the computing power of the cloud platform to process one million instructions per second.

Then the total time spent on the number of tasks assigned to a cloud platform can be expressed as:

$$TimeCost_{l,total} = \sum_{i=1}^{TaskNum} TimeCost_i \quad (2)$$

l is the lth cloud platform, TaskNum is the total number of tasks on the platform, and total is the total time.

The fitness function based on the minimum task completion time is:

$$f_1(j) = \max(TimeCost_{l,total}) \quad (3)$$

3.2.2 fitness function of optimal matching degree

Before matching degree was calculated, the performance parameters of the cloud platform resources should be normalized as shown in the following formula:

$$SP_l = \frac{cuP_l - \min P}{\max P - \min P} \quad (4)$$

Here SP_l is the normalized value, cuP_l represents the size of a certain type (CPU, Memory, Band With) parameter of the current cloud platform, $\max P$ and $\min P$ respectively represent the maximum and minimum values of certain types of parameter values in all cloud platforms. Value

We define the Euclidean distance between the cloud platform's preference vector and the cloud platform's capability vector for different types of tasks as the matching degree. The smaller the value, the more matching, as shown in the following equation:

$$D_i = \sqrt{\sum (SP_l - Pre_i)^2} \quad (5)$$

For a cloud platform, the matching degree is the sum of the matching degrees of all the tasks currently assigned, as shown in the following formula:

$$D_{l,total} = \sum_{i=1}^{i=TaskNum} D_i \quad (6)$$

So the fitness function based on the degree of matching can be expressed as:

$$f_2(j) = \frac{1}{\min_{l=1}^{PNum} D_{l,total}} \quad (7)$$

3.3 Select and Copy

The corresponding selection probability of the fitness function based on the completion time of the small task is:

$$P_1 = \frac{f_1(j)}{\sum_{j=1}^{popsize} f_1(j)} \quad (8)$$

The probability that each individual is selected based on the fitness function of the matching degree is as follows:

$$P_2 = \frac{f_2(j)}{\sum_{j=1}^{popsize} f_2(j)} \quad (9)$$

In the selective replication phase, in order to make the selected scheduling scheme have both a minimum scheduling schedule based on minimum task completion time and a scheduling scheme based on optimal matching degree, we use α and β to indicate the probability of selecting and respectively. Where $\alpha + \beta = 1$, here we define the maximum value word in $\alpha * P_1$ and $\beta * P_2$ as the selection probability of the scheduling scheme, and then randomly generate a sequence of random numbers between popsize $[0, 1]$, we use this random sequence with $\alpha * P_1$ and $\beta * P_2$. The sequence of selected maximum values is compared. If the random number is greater than it, the scheduling scheme is selected, otherwise it is not selected.

3.4 Adaptive Crossover and Mutation Probability

In order to satisfy the crossover and mutation probability, we can adaptively adjust the size according to the population size and fitness value. We will express the crossover probability formula as follows:

$$p_{c(t)} = (F_e^2 + F_c^2)^{\frac{1}{2}} * (1 - \frac{D_{avg}}{D_{max}}) * (1 - \frac{t}{T})$$

$$F_e = (f_{max} - f_{avg}) / (f_{max} - f_{min})$$

$$F_c = (f_{max} - f_{(x)}) / (f_{max} - f_{min}) \quad (10)$$

The probability of variation is:

$$p_{m(t)} = \frac{F_m}{1 + F} * \frac{D_{avg}}{D_{max}} * \frac{\arctan(t^{\frac{1}{2}})}{\pi}$$

$$F_m = (f_{max} - f_{(x)}) / (f_{max} - f_{min}) \quad (11)$$

In equation (10), t represents algebra, $P_{c(t)}$ represents the crossover probability of the t th generation, D_{max} represents the maximum value of the Euclidean distance between contemporary populations, D_{avg} represents the average of the Euclidean distances between contemporary populations, and f_{max} , f_{min} and f_{avg} represent contemporary The maximum, minimum, and average values of the fitness values, and $f_{(x)}$ represents the one with the larger fitness value among the two parent crosses. $P_{m(t)}$ in the formula (11) represents the mutation probability of the t -th generation.

3.5 Convergence termination condition

The specific expression is as follows:

$$sd = \left(\frac{\sum_{i=1}^{popsize} (f_1(i) - f_{avg})^2}{popsize} \right)^{\frac{1}{2}} < \sigma \quad (12)$$

The expression $f_1(i)$ represents the fitness value of the i -th individual, f_{avg} represents the average value of the fitness values of all individuals in the contemporary population, and $popsize$ is the size of the population. σ is the convergence threshold and σ satisfies $\sigma \in (0,1)$. Here we take σ as 0.1, that is, when the standard deviation of the fitness function value of the minimum task completion time is less than 0.1, the iteration is terminated, and at the same time, the individual with the largest fitness function value is output. And it is decoded to get an approximate optimal resource scheduling scheme.

4.algorithm performance test

This section uses the environment of Cloudsim to simulate the cloud computing platform to verify the algorithm. The improved genetic algorithm is compared with the cuckoo search algorithm (CS) in the literature[5], the particle swarm optimization (PSO) in the literature [6], the traditional genetic algorithm (GA) and the Sufferage algorithm.

(1) Small-scale resource scheduling simulation: Here, only five cloud resource platforms can be set to be called, and the corresponding number of tasks to be executed is increased from 10 to 100, and the time used to generate a complete scheduling scheme is recorded, such as Figure 1. From Figure 1, we can see that the performance of the five algorithms does not differ much when dealing with small-scale resource scheduling problems.

(2) Large-scale resource scheduling simulation: Here we set the number of cloud platform resources to a fixed value of 30, and change the number of tasks from 1000 to 3500, respectively, and simulate and record the time used to generate a complete scheduling scheme. ,as shown in picture 2:As can be seen from Figure 2, as the number of tasks increases sharply, the performance difference between IGA and the other four algorithms becomes more and more obvious, especially the running time of the Sufferage algorithm increases significantly. It can be clearly seen that the performance of IGA is significantly better than the other four algorithms when dealing with large-scale resource scheduling problems.

(3) Now we have a fixed number of tasks of 3,500, which makes the number of cloud platform resources dynamically change from 10 to 35, and the same record takes the time required to generate a complete scheduling solution. As shown in Figure 3:When the number of tasks is fixed at 3,500 and

the number of cloud platform resources is 10, the performance of IGA is significantly better than the other four algorithms. However, as the number of cloud platform resources increases, the running time of the five algorithms is significantly shortened, slowly tend to be equal. When we increase the number of cloud platform resources to 35, the running time of the five algorithms is very small, because the cloud platform resources have the ability to handle all the tasks. The task needs, there is basically no phenomenon of robbing resource nodes or prioritizing execution.

(4) The degree of matching is reflected in the occupancy rate of the task bandwidth. For the asymmetric digital subscriber line ADSL, when the provider can provide downlink and uplink speeds of 1.5Mb/s and 256 kb/s-1Mb/s respectively. At the time, the occupancy rate of the downlink bandwidth is actually only 60 percent to 85.7 percent. Therefore, the proportion of the actual bandwidth of the downlink bandwidth is used as the evaluation criterion of the matching degree in this experiment, as shown in Figure 4: As can be seen from Figure 4, IGA's downlink bandwidth occupancy rate is higher than 50% in 60%-85.7% and is superior to CS, PSO, GA, and Suffrage. This shows that it is effective to add a fitness function based on the degree of matching.

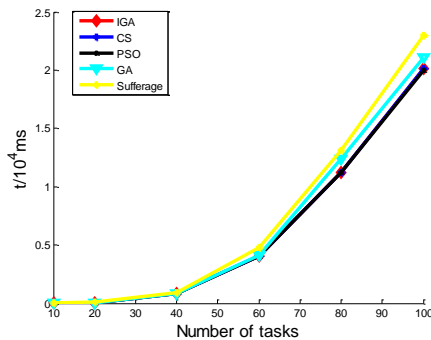


Figure 1: Comparison of generation time of small-scale scheduling schemes

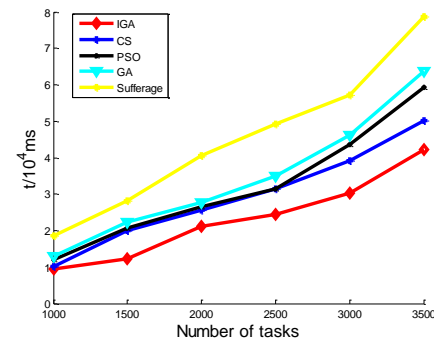


Figure 2: Scheduling scheme generation schedule with variable number of tasks

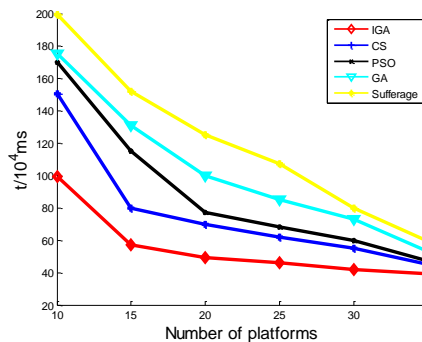


Figure 3: Scheduling scheme generation schedule with variable number of platforms

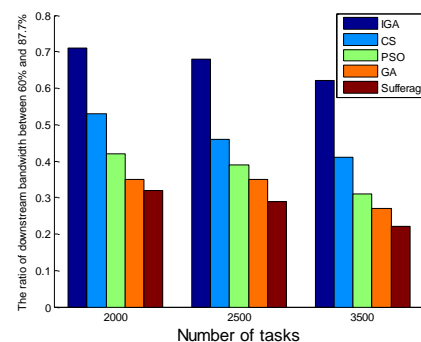


Figure 4: Matching degree comparison chart

5. Conclusion

In summary, a task-resource quadratic real number coding method designed in this paper can solve large-scale resource scheduling problems better, and introduces a dual fitness function based on minimum task completion time and optimal matching degree to make scheduling. The scheme is more comprehensive, and the cross mutation probability is optimized so that the algorithm can dynamically adjust the search space to converge to the optimal solution as soon as possible. Finally, the standard deviation based on the minimum task completion time fitness value is used as the termination condition, which allows the algorithm to jump out in time. The cycle saves time. Through simulation verification and comparison analysis, it is proved that the improved method is feasible and can be well applied to the resource scheduling problem under the multi-user large-scale cloud computing task.

References

- [1] Tao Yong, Shen Jinan. Cloud workflow scheduling algorithm based on dynamic critical path[J]. Research on Computer Applications, 2018, 35(5):1500-1505.
- [2] Zhu Jiazhen, Xiao Dan, Wang Fei. Multi-dimensional QoS constraint task scheduling mechanism for load balancing under cloud computing[J]. Computer Engineering and Applications, 2013, 49(9): 85-89.
- [3] Maheswaran M , Ali S , Siegal H J , et al. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems[C]// Heterogeneous Computing Workshop. IEEE, 1999.
- [4] Feng Longhua. Research and implementation of job scheduling algorithm based on greedy model in cloud environment[D]. Chongqing University, 2014.
- [5] Zhao Li. Cloud computing resource scheduling based on improved cuckoo search algorithm[J]. Journal of Nanjing University of Science and Technology(Natural Science), 2016, 40(4): 472-476.
- [6] Meshkati J, Safi-Esfahani F. Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing[J]. The Journal of Supercomputing, 2018, 26(9): 1-42.