# Survey On Task Scheduling Algorithm In Cloud Computing

**Dr. Aishwarya P[1], DeepakV[2]**
[1, 2] Dept of Computer Science
[1, 2] Atria Institute of Technology, India, Bangalore-560024

***Abstract-*** *As cloud computing is growing rapidly, efficient task scheduling algorithm plays a vital role to improve the resource utilization and enhance overall performance of the cloud computing environment. However, task scheduling is the severe challenge needed to solve urgently in cloud computing. Therefore, the simulated annealing multi-population genetic algorithm (SAMPGA) is proposed for task scheduling in cloud computing, which is the combination of simulated annealing algorithm (SA) and multi-population genetic algorithm (MPGA) in this paper. In population initialization, SAMPGA adopts max-min algorithm to enhance the search efficiency. SA incorporated into SAMPGA is employed to avoid local optimum and improve the performance of global optimum, while a family evolution strategy based on adaptive mechanism in MPGA is proposed to find better solution and improve convergence speed. Finally, experiments are conducted to evaluate the efficiency of the proposed method in MATLAB. Compared with MPGA, SA and simulated annealing genetic algorithm (SAGA), the results of simulation show that the SAMPGA has more excellent performance in terms of the completion time, completion cost, convergence speed and degree of load imbalance.*

***Keywords***- Cloud computing, task scheduling, multi-population genetic algorithm, simulated annealing algorithm

## I. INTRODUCTION

is obviously important in the cloud environment. Classical task scheduling algorithms may perform well sometimes, but there are always some shortcomings. The Min-Min algorithm mentioned in [3] can easily result in serious load imbalance of computing resource. The Round-Robin algorithm referred to in [3] is not suitable for resources with different computational capabilities. Genetic Simulated Annealing algorithm proposed in [4] considering the completion time and cost, has extremely limited improvement. Besides, there are some other algorithms, such as Ant Colony Optimization algorithm (ACO) and Particle Swarm Optimization algorithm (PSO) [5], however, the convergence speed of ACO is quite slow and PSO has a poor capacity in local search. Computing resources in cloud computing environment have heterogeneous processing capacities. Therefore, task scheduling algorithms try to efficiently balance the load of the system taking into consideration total execution time and cost. In this paper, SAMPGA combining SA and MPGA which is the improvement of SAGA, is proposed to reduce the completion time, completion cost and load imbalance. The MPGA enhances the exploration and convergence capability of SAMPGA, and the SA equip SAMPGA local search ability.

## II. DESCRIPTION OF TASK SCHEDULING PROBLEM

At present, the vast majority of distributed computing stages receive Google's Map/Reduce programming model [6] to figure it out parallel figuring of substantial scale informational collections. After client presents the undertaking, the stage partitions the errand into a few subtasks, at that point allots these subtasks to various processing assets to acquire the last outcome through Map and Reduce methodology. Hence, the ideal undertaking planning straightforwardly influences usage of registering assets. Errand planning for distributed computing is to apportion a lot of offered undertakings to a few processing assets subject to a few obliges of enhancing execution [7]. In this paper, N is the quantity of subtasks, M is the quantity of figuring assets, and Expect Time to Complete (ETC) lattice is used to ascertain time for finishing subtasks on each figuring asset, where ETC(i, j) signifies the execution time of the subtask j on the figuring asset I. RCU(i) (Asset Cost per Unit) means the expense of running the subtask per unit time of each figuring asset. In perspective on above conditions, the objective of errand planning in distributed computing is the manner by which to allot different subtasks to figuring assets sensibly, which results in shorter the consummation time, lower culmination cost, and adjusting load of registering assets.

## III. CLOUD COMPUTING

Cloud computing is the on demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. The term is generally used to describe data center’s available too many users over

the Internet. Large clouds, predominant today, often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close, it may be designated an edge server.Clouds may be limited to a single organization (enterprise clouds[1][2],) be available to many organizations (public cloud,) or a combination of both (hybrid cloud). The largest public cloud is Amazon AWSCloud computing relies on sharing of resources to achieve coherence and economies of scale.

## IV. SAMPGA

Genetic Algorithm (GA) is a global search method which has strong global search ability, but it is prone to premature convergence and fall into the local optimal solution. SA is a Proceedings of the 36th Chinese Control Conference July 26-28, 2017, Dalian, China 5633 global optimal algorithm, which has ability to jump out of the local optimal range and great local search ability, but it does not learn a lot about the entire search space, so the search efficiency is not satisfying. Therefore, this paper proposes to combine MPGA which is an improved genetic algorithm with SA to play the advantages of each algorithm and make up for each other's shortcomings. In the process of cycle optimization, MPGA takes advantage of the local search ability and avoiding local minimum of SA, meanwhile, the probability of crossover and mutation adopt adaptive mechanism, and multiple populations are divided into different evolution families with different strategies to improve the efficiency and convergence precision of the algorithm.

### *4.1 MPGA*

]GA [8] is a kind of adaptive probability optimization technique which is applied to the optimization of complex systems. However, the traditional genetic algorithm has the problem of premature convergence and low search efficiency in the late period of evolution. MPGA is a excellent method to improve the performance of GA. The basic idea of MPGA is that multiple populations instead of single population are adopted, and immigration operation between populations is employed to promote evolution and communication between the various populations. MPGA can improve search speed and accuracy without negative impact on search time by increasing number of population simply [9].

#### *4.1.1 Chromosome encoding*

In this paper, resource-task scheme is used to encode. The length of the chromosome represents the number of subtasks, and gene value denotes computing resource. In the population initialization, each chromosome is randomly generated. Suppose that there are 10 subtasks and 3 computing resources, chromosome encoding is generated randomly as follows:

{3, 1, 3, 2, 2, 1, 3, 1, 2, 3}

The chromosome indicates that the first subtask is running on resource 3, the second subtask is on resource 1, and so on. After chromosome encoding, it needs to be decoded to obtain the distribution of subtasks on the computing resource. The corresponding relationship between computing resource and subtask is shown in Table 1.

Table 1: Chromosome decoding

| Computing resource | Subtask |
|---|---|
| 1 | 2,6,8 |
| 2 | 4,5,9 |
| 3 | 1,3,7,10 |

A sequence of subtasks assigned to each computing resource is obtained by decoding. It is easy to calculate the time of each computing resource to complete the subtask sequence by using ETC matrix.

$$sumTime(i) = \sum_{j=1}^{n} Time(i,j), i \in [1, M] \quad (1)$$

In the formula (1), Time(i, j) denotes the execution time of the subtask j assigned to the computing resource i. Computing resources work in parallel, so taking maximum of above calculation results as the completion time.

$$completeTime = \max(sumTime(i)) \quad (2)$$

The completion cost is defined as

$$completeCost = \sum_{i=1}^{M} sumTime(i) \times RCU(i) \quad (3)$$

The load imbalance value is defined as

$$load = \sqrt{\frac{1}{M-1}\sum_{i=1}^{M}(sumTime(i) - avgTime)^2} \quad (4)$$

where avg Time is the average running time of computing resource.

#### 4.1.2 Population initialization

The number of population is denoted as MP, the scale of each population is set as NIND, the number of computing resource is denoted as M, and the number of subtasks is set as

N. The max-min algorithm is adopted to initialize populations. The process of initialization operates as follows: MP populations are randomly generated by the system, and each population contains NIND chromosomes with length of N and gene value in the range of [1, M].

**4.1.3 Fitness function**

Fitness function is extremely important in MPGA, which influences the convergence speed and search of optimal solution directly. Task scheduling needs to take into account execution time and cost of all subtasks. Fitness function of time is defined as:

$$F_{time} = \frac{1}{completeTime} \quad (5)$$

Fitness function of cost is defined as

$$F_{cost} = \frac{1}{completeCost} \quad (6)$$

The total fitness function is defined as

$$Fitness = \alpha \times F_{time} + \beta \times F_{cost} \quad (7)$$

where and denote the weight of time and cost respectively, and $\alpha + \beta = 1$. When$\alpha = 0$, $\beta = 1$ it means that the fitness function only considers cost. On the contrast, the fitness function only considers time. When neither of them is 0, it means that both of time and cost are taken into account. In this paper, $\alpha = 0.3$,
$\beta = 0.7$.

**4.1.4 Genetic operation**

Genetic operation in MPGA includes selection, crossover, mutation, and migration. We propose to divide multiple populations into three families and each family adopts different strategy in genetic operation. Family 1 known as exploration family with a large probability of crossover and mutation, is used to provide new hyper planes in the process of evolution. Family 2 with small probability of crossover and mutation, called development family, is used to search for outstanding individual within local context. Family 3 adopts probability of crossover and mutation between the above two, called exploration and development family, which has property of the above two families.

**4.1.4.1 Select operation**

If the fitness value of an individual is bigger, the individual will appear more in the next generation, hence, the population will approach the optimal solution. This paper adopts the following formula for individual selection:

$$N_i = NIND \times \frac{F_i}{\sum F_i} \quad (8)$$

where NIND is the scale of each population and Fi denotes the fitness value of the $i^{th}$ individual. Therefore, $\sum_1^{NIND}$ [Ni] individuals in the next generation can be determined. Then sort individuals in descending order according to the fraction part of Ni and add the first NIND - $\sum_1^{NIND}$[Ni] individuals to the next generation. By far, NIND individuals have been selected.

**4.1.4.2 Crossover and mutation operation**

Crossover and mutation operations are important parts of genetic operation. According to the adaptive genetic algorithm (AGA) proposed in [10], the probability of crossover is defined as:

$$P_c = \begin{cases} k1 * \dfrac{f_{\max} - f'}{completeTime}, & f' \geq f_{avg} \\ k2, & f' < f_{avg} \end{cases} \quad (9)$$

The definition of mutation probability is:

$$P_m = \begin{cases} k3 * \dfrac{f_{\max} - f}{completeTime}, & f \geq f_{avg} \\ k4, & f < f_{avg} \end{cases} \quad (10)$$

In the above two formulas, $f_{max}$ denotes the maximum fitness value of each population, $f_{avg}$ denotes the average fitness value of each population, f denotes the higher fitness value between the two individuals of crossover, f denotes the fitness value of the individual of mutation. The k1, k2, k3 and k4 are different in different families because of different evolution strategies. After the crossover and mutation operations are completed, the Boltzmann mechanism in the literature is adopted to judge whether accept the new individual or not.

**4.1.4.3 Immigration operation**

Immigration is a key feature of MPGA. When immigration operates, each population immigrates a copy of its best individual to its neighboring population, replacing the individual with lowest fitness value in the neighboring population [9]. Immigration operation between multiple

populations enhances global search capability and convergence speed.

### 4.2 MPGA

SA was proposed by Metropolis in 1953. Kirkpatrick et al. [12] succeeded in introducing SA to combinatorial optimization in 1983. SA employs the temperature control parameters and the cooling schedule to avoid local optimum and find solutions that are closer to the global optimum [13]. In the process of searching for the optimal solution, SA not only accepts better solution, but also accepts the poor solution conditionally, which makes the algorithm jump from the local region of the solution and get the global optimal solution [4]. The basic idea is: if the new solution has better performance, it will be accepted as current optimal solution. But if the new solution gets worse, then accept it as the current solution with the Metropolis criterion to ensure the algorithm to find the global optimal solution. Therefore, SA introduced in this paper enhances the local search ability of MPGA.

When exploring the solution space, as the temperature decreases, the probability that the algorithm accept the poor solution will decrease [14]. The temperature mainly affects search performance of SA. Since the amount of calculation and feasibility, formula (11) is generally chosen as a function of SA to control temperature:

$$T(k+1)=\lambda \times T(k) \qquad (11)$$

where: is the cooling coefficient, and slightly lower than 1; k is the number of cooling.

## V. *EXPERIMENT*

In this paper, simulation of task scheduling in cloud computing environment is conducted in MATLAB. The RCU and ETC matrices are randomly initialized in MATLAB. In the SA, the initial temperature is set to 200. In the experiments, set up 50 subtasks and 5 computing resources, the scale of population is 40, the number of population of SAMPGA and MPGA are both 10, and the same fitness function is used for each algorithm. The maximum number of generation is 150, experiments are carried out 100 times consecutively, and the average of 100 experimental data is taken as the measurement. The parameters of experiments are set in Table 2, and the average values of the experimental data are recorded in Table 4. 4.1

### *5.1 Comparison of SAMPGA and SAGA*

| Algorithm | Parameters | Value |
|---|---|---|
| SAGA | NIND | 40 |
| | k1 | 0.6 |
| | k2 | 0.8 |
| | k3 | 0.1 |
| | k4 | 0.05 |
| | M | 5 |
| | N | 50 |
| | Gamma | 0.95 |
| SAMPGA | MP | 10 |
| | NIND | 40 |
| | k1(Family one) | 0.8 |
| | k1(Family two) | 0.6 |
| | k1(Family three) | 0.4 |
| | k2(Family one) | 0.9 |
| | k2(Family two) | 0.8 |
| | k2(Family three) | 0.5 |
| | k3(Family one) | 0.3 |
| | k3(Family two) | 0.1 |
| | k3(Family three) | 0.05 |
| | k4(Family one) | 0.2 |
| | k4(Family two) | 0.05 |
| | k4(Family three) | 0.01 |
| | M | 5 |
| | N | 50 |

The result of comparative experiment between SAMPGA and SAGA is shown in Fig. 1, SAMPGA evolution is shown in Fig. 2. The dotted line in the figure indicates the evolution of SAMPGA and the solid line represents the evolution of SAGA.

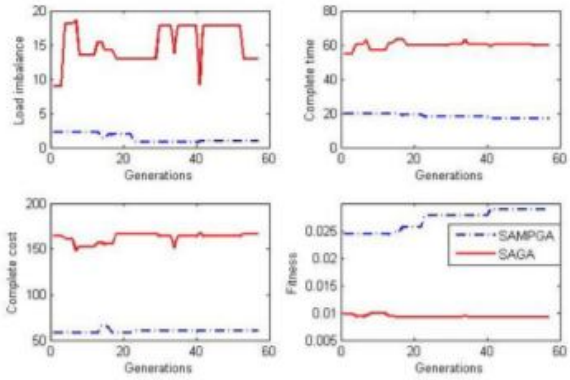


Fig. 1: Comparison between SAMPGA and SAGA

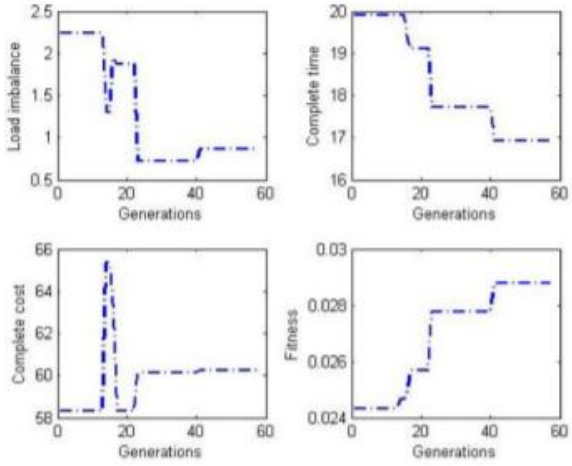


Fig. 2: Evolution process of SAMPGA

Observing Fig. 1, SAMPGA is significantly better than SAGA in terms of the completion time, completion cost, and load imbalance.

Table 3: Results of experiments

| Algorithm | Load | Time | Cost | Generations |
|---|---|---|---|---|
| SAMPGA | 1.4786 | 19.5664 | 44.3245 | 42 |
| SAGA | 13.6604 | 57.3239 | 96.1194 | 46 |
| MPGA | 5.4507 | 40.5734 | 86.1447 | 66 |
| SA | 10.6293 | 50.0481 | 92.0334 | 62 |

From the data shown in Table 3, compared with SAGA, the completion time of SAMPGA decreases by 65.87%, completion cost of SAMPGA decreases by 53.89%, load imbalance of SAMPGA decreases by as high as 89.18 %, the final number of convergence generation is 42 in SAMPGA, and the number of SAGA is 46. This means that SAMPGA achieves a global optimal search at faster convergence speed, and the optimal result is much better than that of SAGA, and the comprehensive capability is more outstanding.

### 5.2 Comparison of SAMPGA and MPGA, SA

Fig. 3 and Fig. 4 show comparative experiments between SAMPGA and MPGA, SA. The dotted lines in Fig. 3 and Fig. 4 indicate the evolution process of SAMPGA, and the solid lines represent the evolution of MPGA and SA, respectively.

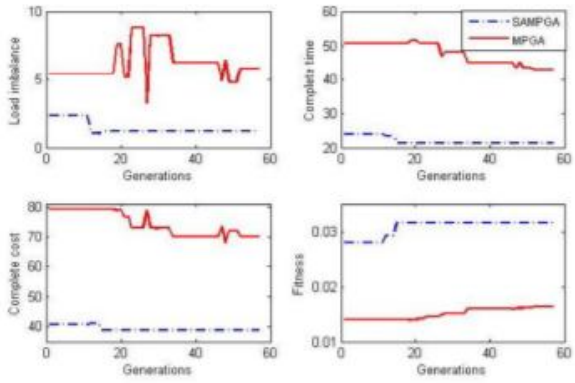


Fig. 3: Comparison between SAMPGA and MPGA

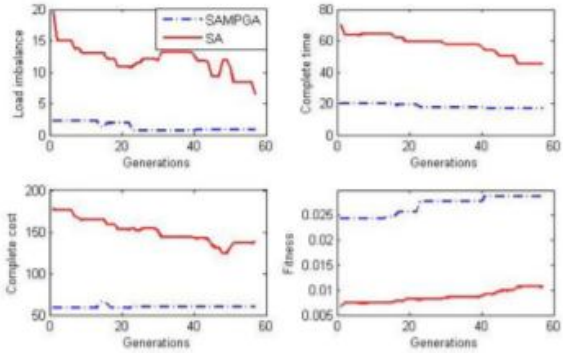


Fig. 4: Comparison between SAMPGA and SA

From Fig. 3 and Fig. 4, compared with MPGA and SA, it is obvious that SAMPGA performs much better, and enhances the efficiency of searching for the global optimal solution. The completion cost is much lower, the completion time is shorter and the load imbalance is smaller. As data shown in Table 3, the completion time, completion cost, load imbalance of SAMPGA compared to MPGA decrease by 51.78%, 48.55% and 72.87%, respectively. Likewise, these indices of SAMPGA compared to SA decrease by 60.9%, 48.55% and 86.09%, respectively. The final number of generation in SAMPGA is 42, and the number of convergence generation in MPGA and SA are 66 and 62 respectively. This means that compared with MPGA and SA, SAMPGA not only can significantly improve the convergence speed, but also dramatically enhances the performance of load imbalance, the completion time, the completion cost.

## VI.CONCLUSION

In this paper, the SAMPGA is proposed to obtain optimal schedule of tasks in cloud computing environment. The proposed algorithm takes advantage of local search capacity of SA to improve the convergence speed and employs MPGA adopting family evolution strategy based on adaptive mechanism to find better global optimal solution and improve

global search ability. And simulation experiments are carried on in MATLAB. The results of experiments show that the SAMPGA outperforms SAGA, MPGA, and SA in terms of the completion time, the completion cost, load imbalance and convergence speed. Therefore, the SAMPGA is more suitable for task scheduling in cloud computing.

## REFERENCES


[1] S.Selvarani, GS Sadhasivam. Improved cost-based algorithm for task scheduling in cloud computing. IEEE International Conference on Computational Intelligence and Computing Research, 2010: 1-5.

[2] JF Li, J Peng. Task scheduling algorithm based on improved genetic algorithm in cloud computing environment. Journal of Computer Applications, 2011, 31(1):184‡186.

[3] Pinal Salot. A survey of various scheduling algorithm in cloud computing environment. International Journal of Research in Engineering and Technology, 2013, 2(2): 131-135.

[4] Guo-ning G, Ting-lei H, Shuai G. Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. 2010 International Conference on Intelligent Computing and Integrated Systems.

[5] Zhan Z H, Liu X F, Gong Y J, et al. Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Computing Surveys, 2015, 47(4): 63.

[6] Jeffrey Dean, Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008, 51(1): 107-113.

[7] CW Tsai, JJPC Rodrigues. Metaheuristic scheduling for cloud: a survey. IEEE Systems Journal, 2014, 8(1):279-291.

[8] JH Holland, Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. Ann Arbor, MI, USA: Univ. of Michigan Press, 1975.

[9] T Starkweather, D Whitley, K Mathias. Optimization using distributed genetic algorithms. International Conference on Parallel Problem Solving from Nature, 1990: 176-185.

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

[11] Liu C, Yin X, Wang B. Stochastic inversion of elastic impedance based on simulated annealing genetic algorithm. SEG International Exposition and 86th Annual Meeting, 2016: 2896-2900.

[12] S Kirkpatrick, CD Gelatt, MP Vecchi. Optimization by simulated annealing. Science, 1983, 220(4598): 671–680.

[13] JOHAN J, ERIC N. Investigating a genetic algorithm simulated annealing hybrid applied to university course timetabling problem. KTH, 2016.

[14] Nehra Pooja, Ahuja Sunil. Efficient scheduling by genetic algorithm and simulated annealing. International Journal of Advance Research, Ideas and Innovations in Technology. 2016,2(3).