# A MULTI-TASK ASSIGNMENT METHOD IN CLOUD-BASED SIMULATION

**Lei Ren[(a)], Hejian Ou[(b)], Jin Cui[(a)], Bowen Li[(a)], Baocun Hou[(c)]**

[(a)]School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China;
[(b)] The High-Tech Research & Development Center Ministry of Science & Technology, P.R.C, Beijing 100044, China;
[(c)]Beijing Simulation Center, Beijing 100854, China

[(a)]renlei@buaa.edu.cn, [(b)]ohj@sina.com, [(c)]baocun_hou@163.com

**ABSTRACT**
Cloud-based Simulation (Cloud Simulation) can significantly improve the capacity of existing networked modeling and simulation systems. The performance of the Cloud Simulation Platform tightly couples with the partitioning and allocation of simulation modules among different hosts. This paper proposed a Cloud Simulation resources scheduling strategy for initial partitioning based on Genetic Algorithm (GA). The Cloud Simulation resources scheduling model was given by considering the computing resources of involved hosts as well as the predicted computing load of simulation modules. The improved strategy of GA is designed according to the characteristics of the Cloud Simulation multi-task assignment problem. The simulation results demonstrate the feasibility and efficiency of the proposed method.

Keywords: Cloud Computing, Cloud Simulation, Genetic Algorithm, Multi-task assignment

## 1. INTRODUCTION

Cloud-based Simulation is a new network-based and service-oriented simulation model which enables users to acquire simulation resources and capacity on demand through network and Cloud Simulation Platform (Ren, and Zhang 2011; Li, Chai, and Zhang 2012). Cloud Simulation can improve the capacity of existing networked modeling and simulation systems significantly by supporting fine-granularity resource sharing, multi-user, cooperation, fault tolerance and security application mechanism (Li, Chai, and Hou 2009).

The performance of Cloud Simulation Platform has a close relationship with the distribution of simulation modules on involved hosts. A reasonable partitioning can lead to a reduction of simulation run-time. In contrast, a poor performance can emerge due to an inadequate distribution of simulation modules. Therefore, optimal allocation mechanism & algorithm for Cloud Simulation services is one of the key operation technologies for virtual simulation environment.

A lot of strategies for resource scheduling have been proposed in order to improve the performance of certain system.

In the context of computational Grids, researchers employ heuristic algorithms such as Simulated Annealing, Tabu Search and Genetic Algorithm for job scheduling (Abraham, Buyya, and Nath 2000). In addition to optimizing execution time, the cost arising from data transfers between resources as well as execution costs must also be taken into account in Cloud computing environment. Suraj Pandey et al. utilize a particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments (Pandey, Wu, and Guru 2010).In simulation domain, both dynamic and static solutions are presented for partitioning simulation modules of distributed simulation system. Statistic partitioning methods or heuristic assignment mechanisms are widely used in simulation environment. Compared with dynamic partitioning algorithms (Lui and Chan 2002, Low 2002, Deelman and Szymanski 1998), static partitioning approaches are usually employed with regard to specific applications (Peschlow, Honecker, and Martini 2007). However, there is little literature on Cloud Simulation oriented simulation module assignment. Allocation algorithms aimed to improve the efficiency of Cloud Simulation Platform are highly required. In this paper, a multi-task assignment method in Cloud Simulation is proposed based on Genetic Algorithm. A relatively accuracy model and the evaluation function are given. Compared with the experiment result of greedy algorithm, our strategy could reduce the simulation run-time efficiently.

The remainder of the paper is organized as follows. In next section, we exhibit the model of simulation resources assignment problem in Cloud Simulation Platform. In Section 3, we developed a simulation module partitioning mechanism based on Genetic Algorithm. Details of the algorithm are given. Simulation results and discussions are presented in Section 4. Finally, conclusions and suggestions for future work are given in Section 5.

## 2. PROBLEM DEFINITION

Simulation modules assignment mechanism is one of the key issues in Cloud Simulation Platform. Cloud Simulation multi-task assignment problem can be described as a quadruple $\{M, E, T, C\}$. $M = \{M_1, M_2, \cdots, M_n\}$ refers to the simulation task which consists of simulation modules $M_i$ $(i = 1, 2, \cdots, n)$. $E = \{N_1, N_2, \cdots, N_m\}$ is the collection of involved hosts to execute the simulation task M. E consists of computing nodes $N_i$ $(i = 1, 2, \cdots, m)$. T is the time matrix. $t_{ij}$ represents the time which computing node i needs to execute simulation module j $(i = 1, 2, \cdots, m; j = 1, 2, \cdots, n)$. The time matrix is a $m \times n$ dimensional matrix.

$$T = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ t_{m1} & t_{m2} & \cdots & t_{mn} \end{bmatrix} \quad (1)$$

While C refers to constrains in the simulation modules partitioning problem in Cloud Simulation Platform.

### 2.1. Related Constrains

The Cloud Simulation Multi-task assignment problem is a problem with timing constrains and assignment constrains.

#### 2.1.1. Timing Constrains

If simulation module $M_i$ is required to be finished before simulation module $M_j$, then $M_i$ is the previous task of simulation module $M_j$, $M_i$ can be presented as Prev$(M_j)$. Similarly, $M_j$ is the next task of simulation module $M_i$, $M_j$ can be expressed as Next$(M_i)(i = 1, 2, \cdots, n;\ j = 1, 2, \cdots, n; i \neq j)$ (Su, Chen, and Shen 2008). The timing constrain among simulation modules is given by formula (2).

$$\begin{cases} Enforce[\{Prev(M_i), M_i\}, \prec] \\ Enforce[\{M_i, Next(M_i)\}, \prec] \\ \qquad i = 1, 2, \cdots, n \end{cases} \quad (2)$$

#### 2.1.2. Assignment Constrains

As to assignment constrains, decision variable $x_{ij}$ is 1 when $M_j$ is assigned to $N_i$, otherwise $x_{ij}$ is 0 $(i = 1, 2, \cdots, m; j = 1, 2, \cdots, n)$.

$$\sum_{i=1}^{m} x_{ij} = 1, x_{ij} \in \{0, 1\} (j = 1, 2, \cdots, n) \quad (3)$$

Formula (3) guarantees that each simulation module of simulation task is assigned to only one computing node of the Cloud Simulation Platform.

### 2.2. Related Constrains

For a certain solution, if simulation module j is assigned to computing node i.

$T_{mod}[j]$ is the finishing time of simulation module $M_j$ $(j = 0, 1, \cdots, n-1)$.

$T_{nod}[i]$ is the current time of computing node $N_i$ $(i = 0, 1, \cdots, m-1)$.

After a certain simulation module $M_j$ is executed on computing node $N_i$, $T_{nod}[i]$ should be updated to $T^*_{nod}[i]$ $(i = 0, 1, \cdots, m-1;\ j = 0, 1, \cdots, n-1)$.

$$T_{mod}[j] = \max\{T_{nod}[i], T_{mod}[Prev(M_j)]\} + T[i][j] \quad (4)$$

$$T^*_{nod}[i] = \max\{T_{nod}[i], T_{mod}[Prev(M_j)]\} + T[i][j] \quad (5)$$

Thus, the run-time T of simulation task can be estimated as

$$T = \max\{T_{nod}[0], T_{nod}[1], \cdots, T_{nod}[m-1]\} \quad (6)$$

## 3. SIMULATION MODULE PARTITIONING STRATEGY

Genetic Algorithm was developed by Holland (Holland 1975) to study the adaptive process of natural systems and to develop artificial systems that mimic the adaptive mechanism of natural systems (Hou, Ansari, and Ren 1994). A standard Genetic Algorithm contains three basic operations: selection, crossover and mutation. In the following part, with the characteristics of the Cloud Simulation multi-task assignment problem considered, a partitioning mechanism based on Genetic Algorithm with improving strategy is presented.

### 3.1. Encoding and Initialization

Assume the population size of Genetic Algorithm is N. First of all, we initial the assignment array randomly use the following formula.

$$A_k[j] = i \quad (7)$$

In formula (7), $k = 0, 1, \cdots, N-1$; $j = 0, 1, \cdots, n-1$; $i \in \{0, 1, \cdots, m-1\}$.

### 3.2. Selection Operation

According to the evaluation function of a solution we discussed in part 3.2, the fitness function can be set as

$$f(A) = \max\{T_{nod}[0], T_{nod}[1], \cdots, T_{nod}[m-1]\} \quad (8)$$

According to the Roulette principle, the probability of a solution $A_k$ is selected to inherited to next generation is given by

$$P_s(A_k) = \frac{\max\{f(A_0), f(A_1), \cdots, f(A_{N-1})\} - f(A_k) + \varepsilon}{\sum_{i=0}^{N-1}(\max\{f(A_0), f(A_1), \cdots, f(A_{N-1})\} - f(A_i) + \varepsilon)} \quad (9)$$

In formula (9), $k = 0, 1, \cdots, N-1$ and $\varepsilon > 0$.

This selection mechanism ensures that the population size of Genetic Algorithm is fixed during the whole operation, while solutions with higher fitness value are more likely to survive to next generation.

### 3.3. Crossover Operation

For each iteration, the probability to execute crossover operation is $P_{c1} = ae^{-\frac{N_c}{b}}$, a and b are constant. With the increasing of iterations, the probability of perform crossover operation is decreasing.

The probability of a certain solution $A_k$ to be selected to execute the crossover operation is

$$P_{c2}(A_k) = \frac{(\max\{f(A_0), f(A_1), \cdots, f(A_{N-1})\} - f(A_k) + \varphi)^{\alpha}}{\sum_{i=0}^{N-1}(\max\{f(A_0), f(A_1), \cdots, f(A_{N-1})\} - f(A_i) + \varphi)^{\alpha}} \quad (10)$$

In formula (10), $\varphi \geq 1$ and $\alpha > 1$. Formula (10) can lead to a result that solutions with higher fitness value have a greater probability to perform crossover operation. Crossover operation between two solutions with high fitness value probably leads to a solution with higher fitness value.

Crossover operation utilize two-point crossover mechanism, select i and j randomly which satisfies $0 \leq i \leq j \leq n-1$, then exchange the segment between A[i] and A[j].

### 3.4. Mutation Operation

For each iteration, the probability to execute mutation operation is $P_{m1} = ae^{-\frac{N_c}{b}}$, a and b are constant. The same as crossover operation, the probability of perform mutation operation is decreasing as well with the increasing of iterations.

The probability of a certain solution $A_k$ to be selected to execute the mutation operation is

$$P_{m2}(A_k) = \frac{(f(A_k) - \min\{f(A_0), f(A_1), \cdots, f(A_{N-1})\} + \delta)^{\beta}}{\sum_{i=0}^{N-1}(f(A_i) - \min\{f(A_0), f(A_1), \cdots, f(A_{N-1})\} + \delta)^{\beta}} \quad (11)$$

In formula (11), $\delta \geq 1$ and $\beta > 1$. Mutation probability is greater to those solutions with low fitness value. Mutation operation of solutions with low fitness value is more likely to lead to better solutions.

For a certain solution $A_k$. Select i randomly and substitute computing node $A_k[i]$ with another available computing node.

Based on these rules, the basic steps of the algorithm can be summarized as the pseudo code shown in figure 1.

**Begin**
*Generate initial population of N solutions* $A_k$*(k=0, 1, ⋯, N-1).*
**While** *(t<MaxGeneration) or (stop criterion)*
*For each* $A_k$ *in the population, evaluate its fitness value.*
*Execute selection operation and update the population.*
*Select solutions according to* $P_{c2}$ *,execute crossover operation and update the population.*
*Select solutions according to* $P_{m2}$ *, execute mutation operation and update the population.*
*Update the best solution.*
**End while**
*Postprocess results.*
**End**

Figure 1: Pseudo code for the algorithm

## 4. SIMULATION EXPERIMENTS AND RESULTS

Historical simulation cases on Cloud Simulation Platform can be used to improve the performance of new simulation task. With large amount of simulation cases, the run-time of a simulation module on a certain computing node of the Cloud Simulation Platform is able to be estimated. Thus, the simulation module assignment approach works.

In our experiment, the time matrix $T_{m\times n}$ is generated randomly. Run-time of different modules of a same computing node is generated randomly. While run-time of a certain module on different computing nodes satisfy a linear relationship. That is, computing node with multi-core and larger memory will lead to a short run-time to a certain simulation module. A Gaussian noise is introduced to the run-time at last.

We conducted the simulation experiment with different parameters for several times. With the following parameters: $m = 30$, $n = 100$, $\varepsilon = 0.5$, $\varphi = 1$, $\delta = 1$, $a = 0.2$, $b = 100$, $\alpha = 2$, $\beta = 2$, the relationship between simulation task run-time and the end criterion (max iterations) with certain population size is presented in figure 2. We conducted each simulation experiment for ten times and then calculated the mean value of run-time.

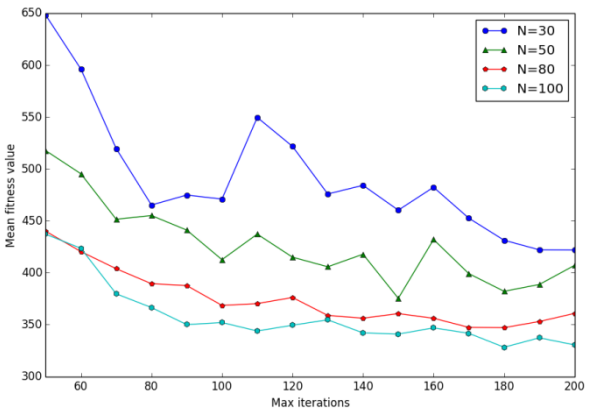


Figure 2: The relationship between simulation task run-time and different end criterion (different max iterations) with certain population size

From figure 2, we could conclude that the simulation run-time has a close relationship with the algorithm population size and end criterion. The simulation task run-time reduced with the increasing of population size of Genetic Algorithm. What's more, the run-time shows downward trend with the max iterations of end criterion increased.

Compared with Greedy Algorithm, the typical experimental results of our partitioning approach are shown in figure 3 (Population size N=50).

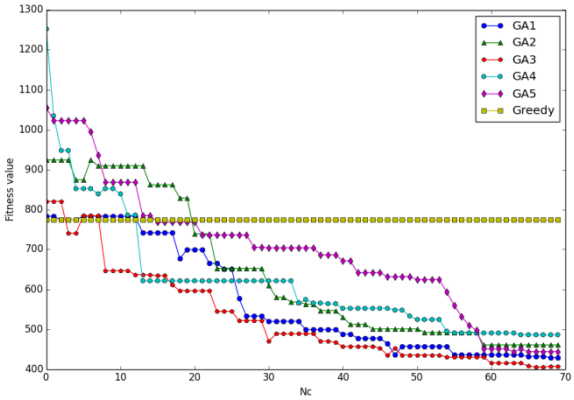


Figure 3: Comparison between Greedy Algorithm and our strategy

We have tested the strategy for five times with an iteration of 70. The mean run-time of the simulation task is 445.09. Meanwhile, the result of Greedy Algorithm is 774.87. Compared with the result of Greedy Algorithm using priority function, our strategy has a better performance with acceptable convergence. Though Greedy Algorithm has an advantage of short execute time, it can be neglected compared with the significant reduction of simulation task run-time by our strategy.

## 5. CONCLUSION AND FUTURE WORKS

In this paper, we revisited the resource scheduling problem in simulation system. With characteristics of the Cloud Simulation Platform taken into account, a multi-task assignment strategy based on Genetic Algorithm is proposed. The probability of crossover operation and mutation operation are modified in order to get better partitioning results. Simulation experiential results demonstrate that the method can efficiently solve the Cloud Simulation multi-task assignment problem. Therefore, our simulation module partitioning mechanism could improve the performance of the Cloud Simulation Platform.

As future research, we plan to focus on improving the simulation performance of Cloud Simulation Platform via utilizing the historical simulation cases. Dynamic partitioning approach is also an important aspect to study in Cloud Simulation Platform.

## ACKNOWLEDGMENTS

The research is supported by the NSFC (National Science Foundation of China) Projects (No.61103096) in China, the National High-Tech Research and Development Plan of China under Grant No. 2013AA041302, and the Fundamental Research Funds for the Central Universities in China.

## REFERENCES

Abraham, A., Buyya, R., Nath, B., 2000. Nature's heuristics for scheduling jobs on computational grids. *Proceedings of the 8th IEEE International Conference on Advanced Computing and Communications*, pp. 45-52. December 16-19, Cochin (India).

Deelman, E., Szymanski, B.K., 1998. Dynamic load balancing in parallel discrete event simulation for spatially explicit problems. *Proceedings of the Parallel and Distributed Simulation*, pp. 46-53. December 14-16, TaiWan.

Su, F., Chen, Y., Shen, L.C., 2008. UAV cooperative multi-task assignment based on ant colony algorithm. *Acta Aeronautica et Astronautica Sinica*, 29(S1): S184-S191.

Holland, J.H., 1975. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Michigan: University of Michigan Press.

Hou, E.S.H., Ansari, N., Ren, H., 1994. A genetic algorithm for multiprocessor scheduling. *IEEE Transactions on Parallel and Distributed Systems*, pp. 113-120. December 19-21, TaiWan.

Ren, L., Zhang, L., 2011. Key issues in cloud simulation platform based on cloud computing. *Proceedings of the 23th European Modeling & Simulation Symposium*, pp. 50-507. September 12-14, Rome( Italy).

Li, B.H., Chai, X.D., Hou, B.C., 2009. Networked modeling & simulation platform based on concept of cloud computing—cloud simulation platform. *Journal of System Simulation*, 21(17), 5292-5299.

Li, B.H., Chai, X.D., Zhang, L., 2012. New advances of the research on cloud simulation. *Advanced Methods, Techniques, and Applications in Modeling and Simulation*. Japan: Springer.

Low, M.Y.H, 2002. Dynamic load-balancing for bsp time warp. *Proceedings of the 35th Annual Simulation Symposium*, pp. 267-274. April 14-18, San Diego (California, USA).

Lui, J.C.S., Chan, M.F., 2002. An efficient partitioning algorithm for distributed virtual environment systems. *IEEE Transactions on Parallel and Distributed Systems*, pp. 193-211. December 17-20, TaiWan.

Pandey, S., Wu, L., Guru, S.M., 2010. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications* , pp. 400-407. April 20-23, Perth (Australia).

Peschlow, P., Honecker, T., Martini, P., 2007. A flexible dynamic partitioning algorithm for optimistic distributed simulation. *Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation*, pp. 219-228. June 12-15, San Diego (California, USA).