International Journal of Grid Distribution Computing
Vol. 8, No.6, (2015), pp.19-30
http://dx.doi.org/10.14257/ijgdc.2015.8.6.03

Load Balancing of Virtual Machines in Cloud Computing Environment Using Improved Ant Colony Algorithm

Yang Xianfeng[1] and Li HongTao[2]

[1]*School of Information Engineering,*
Henan Institute of Science and Technology, Henan Xinxiang, China
[2]*Hebi Polytechnic, Henan Hebi, China*
49377535@qq.com, hnhblht@126.com

Abstract

Load balancing of virtual machines is one of the most significant issues in cloud computing research. A common approach is to employ intelligent algorithms such as Ant Colony Optimization (ACO). However, there are two main issues with traditional ACO. First, ACO is very dependent on the initial conditions, which might affect the final optimal solution and the convergence speed. To solve this problem, we propose to employ Genetic Algorithm (GA) for ACO initialization. Second, ACO could arrive at local optimal point, and the convergence speed is typically low. Along this line, we introduce the idea of Simulated Annealing (SA) to avoid local optimal and accelerate the convergence. Lastly, our experiments show that our improved ACO achieves good performance in load balancing.

Keywords: *Load balancing, Cloud computing, Ant Colony Optimization (ACO)*

1. Introduction

Recent years have witnessed the development of cloud computing [1-2], and one of the most significant issues in cloud computing is load balancing [3-4]. In cloud computing environment, data centers are intensively clusters with high performance computers and network devices. Since the requirements of users are various and dynamically changing, and the resources and services are typically heterogeneous, it is common that the workload in the clusters is usually unbalanced. Specifically, some physical machines are over-loaded and the efficiency is affected, while others are often idle and thus the resources are wasted. Therefore, how to ensure the load balance is a significant problem is cloud environment.

Intelligent heuristic methods have attracted extensive attentions from many researchers. Among them, Ant Colony Optimization (ACO) [5] algorithm is a heuristic optimization technique with positive feedback mechanism, which is quite suitable for workload simulation. Indeed, there are some existing works using ACO in workload field. For example, Eugen *et al*., [6] modeled the workload consolidation problem as a multi-dimensional bin-packing (MDBP) problem and designed an ACO based solution. Chen *et al*., [7] Investigated the satellite data transmission resources workload balance scheduling with ACO. Ali *et al*., [8] studied on load balancing in distributed systems with ACO. Inspired by existing efforts, we propose to employ ACO for load balancing in cloud environment.

However, there still remain two challenges along this line. First, ACO is very dependent on the initial conditions, which might affect the final optimal solution and the convergence speed. Also, it is indicated that approximately 60% time cost is spent on pheromone accumulation [9]. Therefore, pheromone initialization is a significant issue. Second, ACO could arrive at local optimal point, and the convergence speed is typically low.

ISSN: 2005-4262 IJGDC

Therefore, in this paper, we propose some improvements on leveraging ACO for load balancing. First, to optimize the initialization conditions of ants and pheromones, we propose to employ Genetic Algorithm (GA) for ACO initialization, which is proved to be efficient [10-11]. Second, during the iterations of ACO, we introduce the idea of Simulated Annealing (SA) [12] to avoid local optimal and accelerate the convergence. Lastly, our experiments show that our improved ACO achieves good performance in load balancing.

2. Related Work

There are many efforts on load balancing. Godfrey *et al*. [13] developed a protocol based on Chord to solve heterogeneity and load balance issues in distributed hash tables. Pham *et al*. [14] investigated on multi-path routing mechanism with load balance to increases the network throughput. Godfrey *et al*. [15] proposed an algorithm for load balancing in such heterogeneous and dynamic P2P systems. Wang *et al*. [3] designed a two-phase scheduling algorithm under a three-level cloud computing network by combining OLB (Opportunistic Load Balancing) and LBMM (Load Balance Min-Min) scheduling algorithms. Hu *et al*. [16] solved the problem of load imbalance and high migration cost by traditional algorithms after scheduling. Xu *et al*. [17] proposed a load balancing model based on cloud partitioning. Nishant *et al*., [18] proposed an algorithm for load distribution of workloads among nodes of a cloud using ACO. Besides, there are many other efforts on solving load balancing in cloud environment based on ACO [19-21].

Another category of related work is ACO modifications. Stutzle *et al*., [22] limited the updates of pheromones and proposed Max-Min Ant Systems (MMAS). Kaveh *et al*. [23] combined particle swarm optimization (PSO), passive congregation (PSOPC), ant colony optimization (ACO) and harmony search scheme (HS) together. Bell *et al*. [24] introduced ranking techniques to improve the ability of a single ant colony. Lee *et al*. [25] proposed an immunity-based ant colony algorithm to improve the performance of ACO. Besides, ACO is also combined with other intelligent algorithms, such as GA [10-11] and PSO [26-27].

3. Preliminary and Problem Formulation

3.1. ACO Basics

Ant Colony Optimization (ACO) algorithm was first proposed by Dorigo [28], which is inspired by the food searching behavior of ant colony. The basic idea is that ants leave pheromones when passing a route, and the more pheromones the route has, the more likely it would be chosen by other ants. In this way, the probability of choosing routes is related to the amount of pheromones, which contributes to positive feedback learning. That is, ant colony communicates with each other by releasing pheromones on routes when looking for food.

The ACO algorithm can be described as follows. Suppose the phenomenon of ant k on the path from node i to j at time t is notated as $\tau_{ij}^{k}(t)$, and the initial phenomenon value is $\tau_{ij}^{k}(0) = c$, where c is a constant. Notate the probability of ant k transferring from node i to j at time t as $p_{ij}^{k}(t)$, which is calculated as:

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}^{k}(t)^{\alpha}\left(1/d_{ij}^{k}\right)^{\beta}}{\sum_{s\in allowed_{k}}\tau_{ij}^{s}(t)^{\alpha}\left(1/d_{ij}^{s}\right)^{\beta}}, & j\in allowed_{k};\\ 0, & otherwise.\end{cases} \quad (1)$$

where $allowed_{k}$ is the set of nodes that haven't been walked through yet by ant k, α is the heuristic factor, meaning the importance of route with remaining pheromones, β is the heuristic factor for $1/d_{ij}^{k}$, denoting the effect of heuristic information, and d_{ij}^{k} is the distance of route (i,j) by ant k.

At the next iteration $t+1$, the phenomenon at route (i,j) is updated as follows:

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t) \quad , \quad (2)$$

$$\tau_{ij}(t)=\sum_{h=1}^{m}\Delta\tau_{ij}^{k}(t) \quad , \quad (3)$$

Where $1-\rho$ is the residual coefficient of pheromones, $\rho\subset[0,1)$, and m is the number of ants. $\Delta\tau_{ij}^{k}(t)$ is the amount of pheromones remaining on the route at current iteration for ant k, which is calculated as:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{L_{k}}, & \text{if ant } k \text{ passes } (i,j) \text{ at current iteration;}\\ 0, & \text{otherwise.}\end{cases} \quad (4)$$

Where Q is a constant, and L_{k} is the total length of ant k's tour.

3.2. Problem Formulation

The employment of virtualization in cloud computing causes heterogeneous infrastructures and different user demands, and therefore the allocation of cloud resources is typically unbalanced among physical nodes, which contributes to unbalance issues of cloud services. Indeed, load balancing of virtual machines in cloud environment is very important of cloud resource allocation and scheduling.

In this paper, we propose to apply ACO in load balancing problem. Specifically, the food searching process of ants is simulated as the best assignment of virtual machines on a set of physical nodes. The routing of each ant is an analogue of the assignment of each virtual machine. The amount of pheromones corresponds to the workload of nodes, and the more pheromone, the more suitable the node is for specific virtual machine.

Suppose there are M virtual machines $V=\{V_1,V_2,V_3...,V_M\}$ and N available physical nodes $P=\{P_1,P_2,P_3...,P_N\}$. The problem can be formulated as a combination optimization problem. Recall that our objective is to find the optimal assignment solution for load balancing, as well as minimizing the total makespan of all virtual machines.

Consider the load of virtual machines as CPU, memory and network bandwidth workloads. Notate the utilization rate of CPU, memory and network bandwidth as U^{Cpu}, U^{Mem}, and U^{Nb} respectively. Thus, U_j^{Cpu}, U_j^{Mem}, and U_j^{Nb} are the utilization rate of CPU, memory and network bandwidth for physical node $j(j=1,2,...,N)$ respectively, and U_{avg}^{Cpu}, U_{avg}^{Mem}, and U_{avg}^{Nb} are the average utilization rate of CPU, memory and network bandwidth for the cloud environment respectively. Therefore, we define the degree of balancing as:

$$DoB = \frac{1}{N}\sum_{j=1}^{N}\left|U_j^{Cpu} - U_{avg}^{Cpu}\right|^2 + \frac{1}{N}\sum_{j=1}^{N}\left|U_j^{Mem} - U_{avg}^{Mem}\right|^2 + \frac{1}{N}\sum_{j=1}^{N}\left|U_j^{Nb} - U_{avg}^{Nb}\right|^2 \quad (5)$$

And the smaller DoB is, the more balancing the cloud environment could achieve.

Therefore, the objective function is $\min\ DoB$. Besides, the requirement of cloud resources is to minimize the makespan of tasks execution on all nodes. Suppose t_{ej}, t_{sj} denote the execution time of the ending and starting task of virtual machines on node j respectively, and thus the makespan of node j is $Makespan = \max\{t_{ej}\} - \min\{t_{sj}\}$. Therefore, another objective function is $\min\ Makespan$.

4. Load Balancing with Improved ACO

There are some issues with the traditional ACO algorithm. First, the ACO is quite dependent on the initial conditions of ants, and if not set appropriately, it might take a long time for the algorithm to converge. Second, traditional ACO is easy to arrive at local optimal and usually has a low convergence speed. To this end, in this paper, we propose to introduce GA and SA to deal with above two issues respectively. Basically, we use GA to generate the initialization condition of ants, and embed the SA idea into route optimization and pheromone updates.

4.1. Initialization

First of all, we use GA for fast global search in order to initialize the pheromones of ACO. Recall that we aim to assign M virtual machines onto N physical nodes. Define the length of chromosome as the number of virtual machines M , and each gene within chromosome denotes which physical node specific virtual machine is assigned.

Solutions with great fitness values are selected and passed to the next generation, while those with small fitness values are eliminated. Therefore, with the consideration of objective functions, we define the fitness function as

$$F = \frac{1}{\varepsilon DoB + \lambda Makespan}, \quad (6)$$

Where ε, λ are the factors of load balancing and makespan, and $\varepsilon + \lambda = 1, 0 < \lambda < \varepsilon < 1$.

Figure 1, gives the flow chart of GA process, after which we get the initialized optimal solution. The key is to determine when to stop GA and then use the current solution to initialize ACO. We define the evolution rate of generation g as:

$$\eta_e(g) = \frac{|F(g) - F(g-1)|}{|F(g-1) - F(g-2)|}, \quad (7)$$

Where $F(g)$ is the best fitness of generation g .

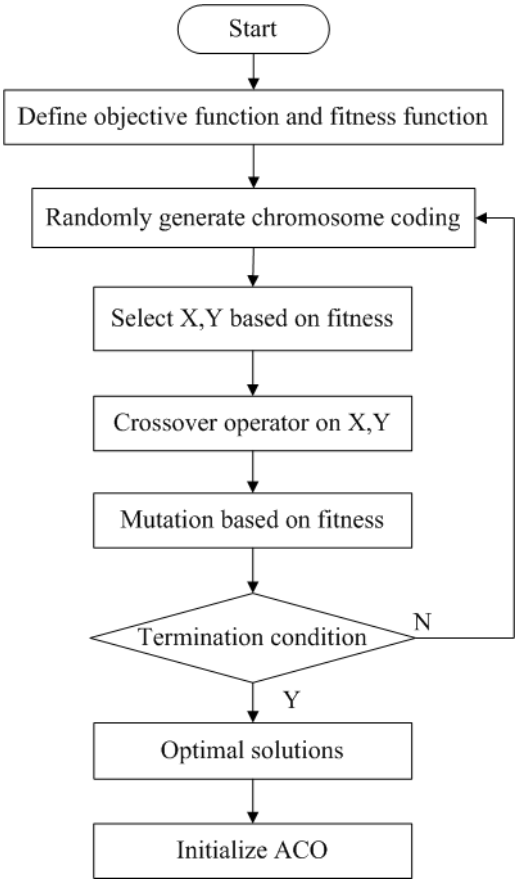

Figure1. Flow Chart of GA

Set the termination condition of GA as $\eta_e < \varepsilon$, where threshold ε is a very small constant. That is, when the improvement between GA generations is not significant, GA algorithm stops. Suppose the initial pheromone of physical node j is $\tau_j(0)$, which is set by the optimal solution of GA. Therefore, the initial pheromone of virtual machine i on node j is set as $\tau_{ij}(0) = \tau_j(0)$.

4.2. Node Selection

Ants decide next movement based on the amount of pheromones on routes. Add nodes that are passed through by ant k into a tabu list, notated as $tabu_k$. For ant k, the probability of virtual machine i to be assigned on node j at time t is calculated as:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}^{k}(t)]^{\alpha}[\eta_{ij}^{k}(t)]^{\beta}}{\sum_{s \in allowed_k} [\tau_{ij}^{s}(t)]^{\alpha}[\eta_{ij}^{s}(t)]^{\beta}}, & j \in allowed_k; \\ 0, & otherwise. \end{cases} \quad (8)$$

where $allowed_k = N - tabu_k$ denotes the set of nodes ant k has not yet passed, $\tau_{ij}^{k}(t)$ is the pheromone of virtual machine i on node j for ant k, and $\eta_{ij}^{k}(t)$ is the expectation of assigning virtual machine i on node j. Define $\eta_{ij}^{k}(t)$ based on the distance between virtual machines and physical nodes:

$$\eta_{ij}^{k}(t) = \frac{1}{\sqrt{(S_j^{Cpu} - R_i^{Cpu})^2 + (S_j^{Mem} - R_i^{Mem})^2 + (S_j^{Nb} - R_i^{Nb})^2}}, \quad (9)$$

Where S_j^{Cpu}, S_j^{Mem}, S_j^{Nb} are the available supply of CPU, memory and network bandwidth resources of node j respectively, and R_i^{Cpu}, R_i^{Mem}, R_i^{Nb} are the requirement of CPU, memory and network bandwidth resources of virtual machine i respectively.

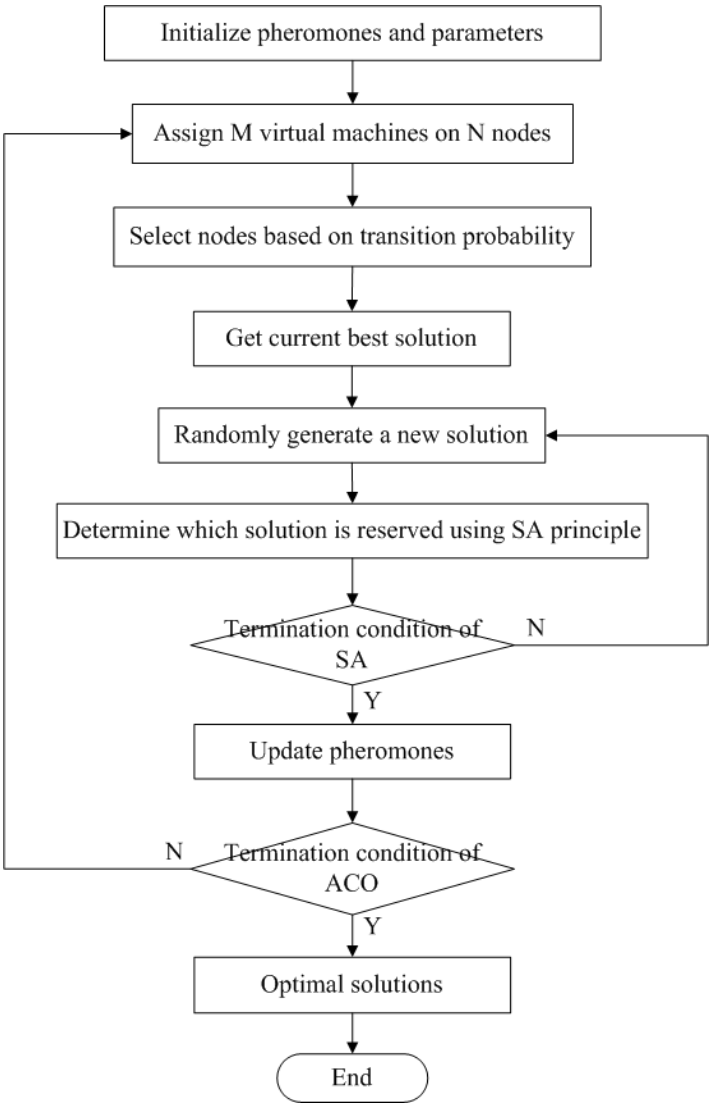

Figure 2. Flow Chart of Improved ACO with SA

4.3. Pheromone Updates

Now we have the best solution of node selection sequence X for ants. Traditional ACO use this sequence for pheromone updates directly until convergence or the maximum iteration number is satisfied. In order to improve the search performance, we introduce the SA idea for pheromone updates. Specifically, as shown in Figure 2, the process of pheromone update is as follows.

Step 1: randomly exchange two assignments to generate a new solution X'.

Step 2: calculate fitness values for X, X'. If $F(X) - F(X') < 0$, replace X with X'; otherwise, if $\exp(-(F(X) - F(X'))/T) > rand(0,1)$, replace X with X'; otherwise, preserve X.

Step 3: $T(t+1) = L \cdot T(t)$, where T is the annealing temperature, and L is the annealing coefficient.

Step 4: if $T < T_{end}$, where T_{end} is the stopping temperature, go to Step 5; otherwise, go to Step 1.

Step 5: update pheromone as Equations (2) and (3).

To sum up, combining Figures 1 and 2, the steps of improved ACO for load balancing are described as follows.

Step 1: decode solutions as chromosomes; initialize fitness values, and GA populations.

Step 2: perform selection, crossover and mutation operations, and get the optimal solution, notated as X^G.

Step 3: initialize the amount of pheromones on physical nodes based on X^G.

Step 4: each ant selects node by Equation (8) to assign virtual machines on nodes, as discussed in Section 4.2.

Step 5: update pheromones as Section 4.3.

Step 6: if the maximum number of iteration is satisfied, algorithm stops; otherwise, go to Step 4.

5. Experiment

We extend CloudSim [29] to simulate cloud environment. We have 4 physical nodes totally, and the physical parameters of our experimental environment are listed in Table 1. The weights of CPU, memory and network bandwidth are set as 0.4, 0.3 and 0.3. Parameters for GA are as follows: the population size is 100, crossover rate is 0.6, and mutation rate is 0.05. Parameters for ACO are: the number of ants is 100, $\alpha = 0.5, \beta = 0.25$. And the annealing coefficient $L = 0.95$. The baseline methods are GA only, ACO only, GA for initialization and then ACO (GA+ACO), and our proposed method.

Table 1. Parameters for Physical Nodes

# of CPUs	MIPS(10^3)	Memory (GB)	Network bandwidth (Mb/s)
4	4	8	1500
4	3	6	1500
2	3	4	1000
2	2	4	1000

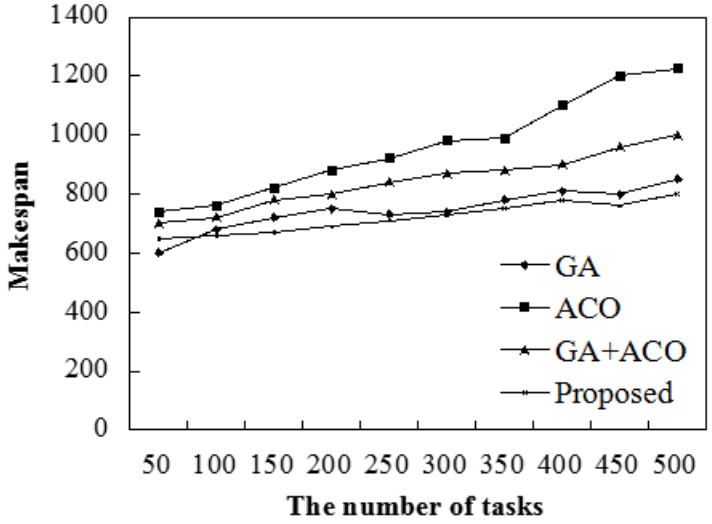

Figure 3. Makespan Comparison

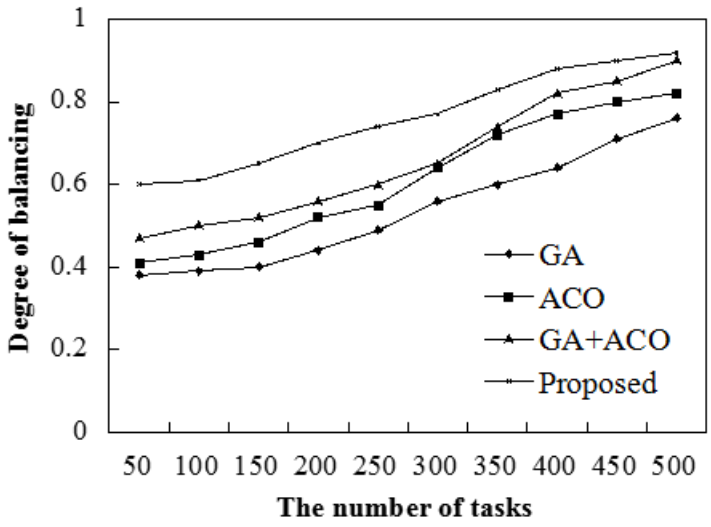

Figure 4. Balancing Performance Comparison

First, we evaluate the performance with different numbers of tasks in terms of total makespan of the whole cluster as well as the degree of load balancing. Figure 3 gives the makespan comparison, which is calculated as the average of 20 executions. Basically, with the number of tasks increasing, total makespan increases as well for all methods. Fortunately, the makespan increment on the curves is not dramatic, because all the algorithms are essentially parallel and are suitable for multi-task execution. From Figure 3, we can see that the makespan of our proposed method is best. On the other hand, Figure 4 shows the degree of balancing DoB for four methods. We can observe that our proposed method outperforms others. Besides, GA performs worst, ACO is better than GA, and combining GA and ACO can achieve even better load balancing performance.

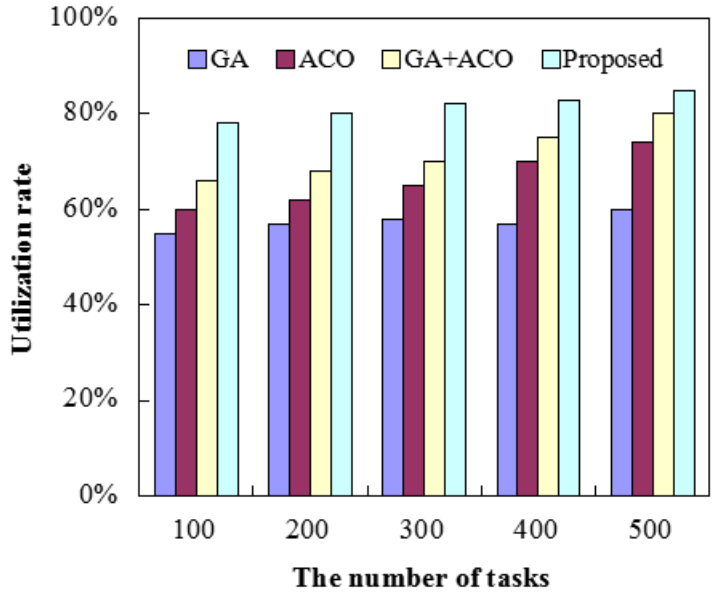

Figure 5. Utilization Rate Comparison

Second, Figure 5 compares the utilization rate of resources with different numbers of tasks. Generally, the more tasks are executed in the cloud, the more balanced the cluster is. Also, the overall utilization rate of resources are more than 50%, which means intelligence algorithms enlightened by biological systems are suitable for load balancing in cloud resources scheduling. Moreover, our proposed algorithm is obviously better than other three methods, and the utilization rate is relatively stable as the number of tasks increases.

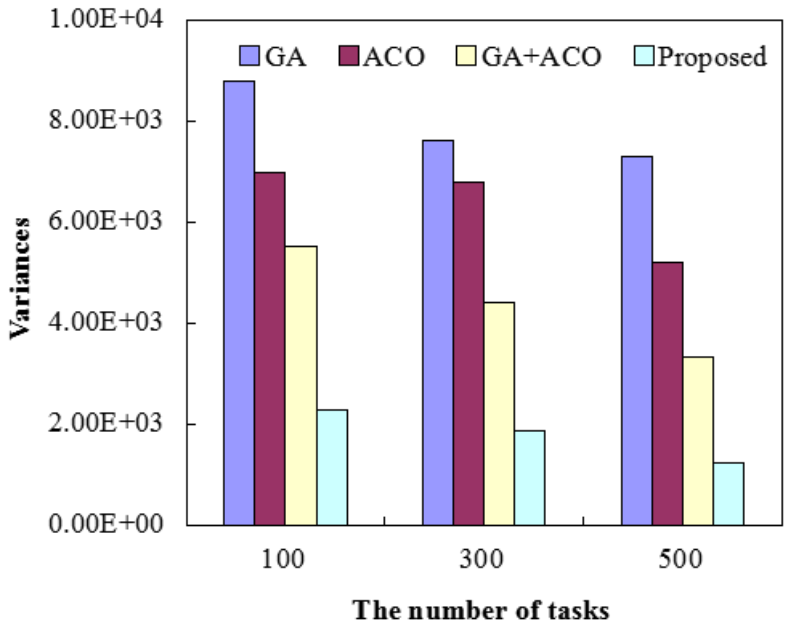

Figure 6. Variances of Makespan

Third, we evaluate the variances of makespan with specific number of tasks to see how balanced the cluster is. As shown in Figure 6, the more variances of makespan, the less balanced the cluster is. Therefore, we can see that our proposed algorithm has the smallest variances no matter how many tasks are executed, and thus our method can efficiently achieve load balancing in cloud environment.

6. Conclusion

In this paper, we propose a modified ACO algorithm for load balancing of virtual machines in cloud environment. Specifically, we introduce GA and SA ideas for performance improvement. In future works, we would like to explore more intelligent algorithms and their applications in cloud computing and big data fields.

References

[1] A. Michael, "A view of cloud computing", Communications of the ACM, vol. 53, no. 4, **(2010)**, pp. 50-58.
[2] M. Peter and T. Grance, "The NIST definition of cloud computing (draft)", NIST special publication, vol. 800, no. 145, **(2011)** July.
[3] S. C. Wang, K. Q. Yan and W. P. Liao, "Towards a load balancing in a three-level cloud computing network", Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on IEEE, vol. 1. **(2010)**, pp. 108-113.
[4] J. Hu, J. Gu and G. Sun, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment", Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on. IEEE, **(2010)**, pp. 89-96.
[5] M. Dorigo and M. Birattari, "Ant colony optimization", Encyclopedia of Machine Learning. Springer US, **(2010)**, pp. 36-39.
[6] E. Feller, L. Rilling and C. Morin, "Energy-Aware Ant Colony Based Workload Placement in Clouds", In Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing (GRID '11). IEEE Computer Society, Washington, DC, USA, **(2011)**, pp. 26-33.
[7] C. Xiang-guo, W. Xiao-yue, "Model of Satellite Data Transmission Resource Workload Balance Scheduling and Ant Colony Optimization Algorithm", Systems Engineering, (in Chinese), **(2008)** December.
[8] A.-D. Ali, M. A. Belal and M. B. Al-Zoubi, "Load Balancing of Distributed Systems Based on Multiple Ant Colonies Optimization", American Journal of Applied Sciences, vol. 7, no. 3, **(2010)**, pp. 428-433.
[9] Z. Qiaohong, C. Mengmeng and G. Fang, "Slotting optimization of warehouse based on hybrid genetic algorithm", Proceedings-2011 6th International Conference on Pervasive Computing and Applications 2011. United States: IEEE Computer Society, **(2011)**, pp. 19-2.
[10] Z.-J. Lee, S.-F. Su, C.-C. Chuang and K.-H. Liu, "Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment", Applied Soft Computing, ISSN 1568-4946, vol. 8, no. 1, **(2008)** January, pp. 55-78.
[11] S. Fidanova, M. Paprzycki and O. Roeva, "Hybrid GA-ACO Algorithm for a model parameters identification problem", Computer Science and Information Systems (FedCSIS), 2014 Federated Conference, **(2014)** September, pp. 413,420, 7-10.
[12] E. Aarts and J. Korst, "Simulated annealing and Boltzmann machines", **(1988)**.
[13] P B Godfrey and I. Stoica, "Heterogeneity and load balance in distributed hash tables", INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, **(2005)** January, pp. 596-606.
[14] P. P. Pham and S. Perreau, "Increasing the network performance using multi-path routing mechanism with load balance", Ad Hoc Networks, no. 4, **(2004)**, pp. 433-459.
[15] B. Godfrey, K. Lakshminarayanan and S. Surana, "Load balancing in dynamic structured P2P systems", INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. IEEE, **(2004)** April, pp. 2253-2262.
[16] J. Hu, J. Gu and G. Sun, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment", Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on IEEE, **(2010)**, pp. 89-96.
[17] G. Xu, J. Pang and X. Fu, "A load balancing model based on cloud partitioning for the public cloud", Tsinghua Science and Technology, vol. 18, no.1, **(2013)**, pp. 34-39.
[18] K. Nishant, P. Sharma and V. Krishna, "Load balancing of nodes in cloud using ant colony optimization", Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on. IEEE, **(2012)**, pp. 3-8.
[19] R. Mishra and A. Jaiswal, "Ant colony optimization: A solution of load balancing in cloud", International Journal of Web & Semantic Technology (IJWesT), vol. 3, no. 2, **(2012)**, pp. 33-50.

[20] K. Li, G. Xu and G. Zhao, "Cloud task scheduling based on load balancing ant colony optimization", Chinagrid Conference (ChinaGrid), 2011 Sixth Annual, IEEE, **(2011)**, pp. 3-9.
[21] J. S. Pan, H. Wang and H. Zhao, "Interaction Artificial Bee Colony Based Load Balance Method in Cloud Computing", Genetic and Evolutionary Computing. Springer International Publishing, **(2015)**, pp. 49-57.
[22] T. Stutzle and H. Hoos, "MAX-MIN ant system and local search for the traveling salesman problem", Evolutionary Computation, IEEE International Conference on. IEEE, **(1997)**, pp. 309-314.
[23] A. Kaveh and S. Talatahari, "A particle swarm ant colony optimization for truss structures with discrete variables", Journal of Constructional Steel Research, vol. 65, no. 8, **(2009)**, pp. 1558-1568.
[24] J. E. Bell and P. R. McMullen. "Ant colony optimization techniques for the vehicle routing problem", Advanced Engineering Informatics, vol. 18, no.1, **(2004)**, pp. 41-48.
[25] Z. J. Lee, C. Y. Lee and S. F. Su, "An immunity-based ant colony optimization algorithm for solving weapon–target assignment problem", Applied Soft Computing, vol. 2, no. 1, **(2002)**, pp. 39-47.
[26] P. S. Shelokar, P. Siarry and V. K. Jayaraman, "Particle swarm and ant colony algorithms hybridized for improved continuous optimization", Journal of Applied mathematics and computation, vol. 188, no. 1, **(2007)**, pp. 129-142.
[27] Y. T. Kao and E. Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions", Applied Soft Computing, vol. 8, no. 2, **(2008)**, pp. 849-857.
[28] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem", Evolutionary Computation, IEEE Transactions on, vol. 1, no. 1, **(1997)**, pp. 53-66.
[29] R. N. Calheiros, R. Ranjan and A. Beloglazov, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Journal of Software: Practice and Experience, vol. 41, no. 1, **(2011)**, pp. 23-50.

