2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation

Ant Colony Algorithm for Job Scheduling in Grid Computing

Ku Ruhana Ku-Mahamud
College of Arts and Sciences,
Universiti Utara Malaysia,
06010 Sintok, Kedah, Malaysia.
E-mail: ruhana@uum.edu.my

Husna Jamal Abdul Nasir
College of Arts and Sciences,
Universiti Utara Malaysia,
06010 Sintok, Kedah, Malaysia.
E-mail: oxalic2131@gmail.com

Abstract – **Scheduling jobs to resources in grid computing is complicated due to the distributed and heterogeneous nature of the resources. Stagnation in grid computing system may occur when all jobs require or are assigned to the same resources. This will lead to resources having high workload and stagnation may occur if computational times of the processed jobs are high. This paper proposed an enhanced ant colony optimization algorithm for jobs and resources scheduling in grid computing. The proposed ant colony algorithm for job scheduling in the grid environment combines the techniques from Ant Colony System and Max – Min Ant System. The algorithm focuses on local pheromone trail update and the trail limit values. A matrix is used to record the status of the available resources. The agent concept is also integrated in this algorithm for the purpose of updating the grid resource table. Experimental results obtained showed that this is a promising ant colony algorithm for job scheduling in grid environment.**

Keywords-Grid Computing, Job Scheduling, Stagnation, Ant Colony Algorithm, Grid Resource Table.

I. INTRODUCTION

Distributed systems consist of multiple computers that communicate through computer networks. Research by [6] defined that cluster and grid computing are the most suitable ways for establishing distributed systems. Cluster computing environment consists of several personal computers or workstations that combined through local networks in order to develop distributed applications. However, applications are difficult to be flexible in cluster computing because they are limited to a fixed area. Grid computing is proposed to overcome this problem where various resources from different geographic area are combined in order to develop a grid computing environment. The study by [7] defined that grid computing is based on large scale resources sharing in a widely connected network such as the Internet.

There are two types of scheduling namely static scheduling and dynamic scheduling in grid computing system. For the static scheduling, jobs are assigned to suitable resources before their execution begin. Once started, they keep running on the same resources without interruption. However, for the dynamic scheduling, reevaluation is allowed of already taken assignment decisions during job execution [9]. It can trigger job migration or interruption based on dynamic information about the status of the system and the workload.

In grid computing system, resources are not under the central control and can enter and leave the grid environment at any time. An effective grid resource management with good job and resource scheduling algorithm is needed to manage the grid computing system (refer Fig. 1). The algorithm must consider the dynamically changes conditions in grid environment because the computational performance changes from time to time, networks connections may become unreliable, resources may join or leave the system at any time and resources may become unavailable without any notifications.

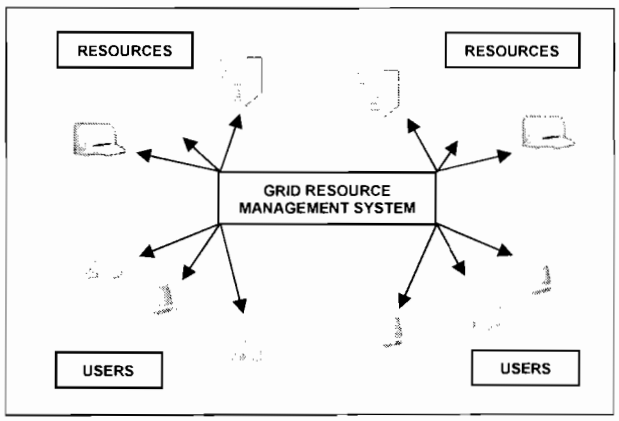

Figure 1: Grid Computing Environment

In grid computing environment, there exists more than one resource to process jobs. One of the main challenges is to find the best or optimal resources to process a particular job in term of minimizing the job computational time. Optimal resources refer to resources having high CPU speeds and large memory spaces. Computational time is a measure of how long that resource takes to complete the job. Stagnation in grid computing system may occur when all jobs required or are assigned to the same resources which will lead to the resources having high workload. An effective job scheduling algorithm is needed to avoid or reduce the stagnation problem.

This paper presents an ant colony algorithm, which is a bio inspired algorithm for job scheduling in grid computing

978-0-7695-4062-7/10 $26.00 © 2010 IEEE
DOI 10.1109/AMS.2010.21

system. Section 2 describes the use of ant colony optimization algorithms in grid computing while the proposed algorithm is discussed in Section 3. Experimental results are presented in Section 4. Lastly, concluding remarks are highlighted in Section 5.

II. Related Works on ACO in Grid Computing Environment

Jobs submitted to a grid computing system need to be processed by the available resources. Best resources in term of processing speed, memory and availability status are more likely to be selected for the submitted jobs during the scheduling process [20]. Best resources are categorized as optimal resources. In a research by [17], Ant Colony Optimization (ACO) has been used as an effective algorithm in solving the scheduling problem in grid computing.

ACO is inspired by a colony of ants that work together to find the shortest path between their nest and food source. Every ant will deposit a chemical substance called pheromone on the ground after they move from the nest to food sources and vice versa. Therefore, they will choose the shortest or optimal path based on the pheromone value. The path with high pheromone value is shorter than the path with low pheromone value. This behavior is the basis for a cooperative communication. The presence of these and other unique characteristics have made ant societies an attractive and inspiring model for building new algorithms. Workers of ant colony specialize in particular tasks. For example, the soldiers aim for protection, the scouts specialize in searching for food sources, and the queen's task is producing new ants.

There are various types of ACO algorithm such as Ant Colony System (ACS), Max-Min Ant System (MMAS), Rank-Based Ant System (RAS) and Elitist Ant System (EAS) [11]. ACO has been applied in solving many problems in scheduling such as Job Shop Problem, Open Shop Problem, Permutation Flow Shop Problem, Single Machine Total Tardiness Problem, Single Machine Total Weighted Tardiness Problem, Resource Constraints Project Scheduling Problem, Group Shop Problem and Single Machine Total Tardiness Problem with Sequence Dependent Setup Times [10]. A recent approach of ACO researches in the use of ACO for scheduling job in grid computing [2]. ACO algorithm has been used in grid computing because it is easily adapted to solve both static and dynamic combinatorial optimization problems and job scheduling in grid computing is an example.

Balanced job assignment based on ant algorithm for computing grids called BACO was proposed by [15]. The research aims to minimize the computation time of job executing in Taiwan UniGrid environment which focused on load balancing factors of each resource. By considering the resource status and the size of the given job, BACO algorithm chooses optimal resources to process the submitted jobs by applying the local and global pheromone update technique to balance the system load. Local pheromone update function updates the status of the selected resource after job has been assigned and the job scheduler depends on the newest information of the selected resource for the next job submission. Global pheromone update function updates the status of each resource for all jobs after the completion of the jobs. By using these two update techniques, the job scheduler will get the newest information of all resources for the next job submission. From the experimental result, BACO is capable of balancing the entire system load regardless of the size of the jobs. However, BACO was only tested in Taiwan UniGrid environment.

An ant colony optimization for dynamic job scheduling in grid environment was proposed by [18] which aimed to minimize the total job tardiness time. The initial pheromone value of each resource is based on expected execution time and actual execution time of each job. The process to update the pheromone value on each resource is based on local update and global update rules as in ACS. In that study, ACO algorithm performed the best when compared to First Come First Serve, Minimal Tardiness Earliest Due Date and Minimal Tardiness Earliest Release Date techniques.

The study by [22] proposed a bio-inspired adaptive job scheduling mechanism in grid computing. The purpose of this research is to minimize the execution time of the computational jobs by effectively taking advantage of the large amount of distributed resource. Various software ant agents were designed with simple functionalities. The pheromone value of each resource depends on their execution time. Resource with high execution time will receive a large number of pheromone. In this research, the comparison was also performed between the bio inspired adaptive scheduling with the random mechanism and heuristic mechanism. Experimental results showed that a bio-inspired adaptive job scheduling has good adaptability and robustness in a dynamic computational grid.

The study to improved ant algorithm for job scheduling in grid computing which is based on the basic idea of ACO was proposed by [4]. The pheromone update function in this research is performed by adding encouragement, punishment coefficient and load balancing factor. The initial pheromone value of each resource is based on its status where job is assigned to the resource with the maximum pheromone value. The strength of pheromone of each resource will be updated after completion of the job. The encouragement and punishment and local balancing factor coefficient are defined by users and are used to update pheromone values of resources. If a resource completed a job successfully, more pheromone will be added by the encouragement coefficient in order to be selected for the next job execution. If a resource failed to complete a job, it will be punished by adding less pheromone value. The load of each resource is taken into account and the balancing factor is also applied to change the pheromone value of each resource.

A simple grid simulation architecture for resource management and task scheduling was proposed in [23]. This study also validated the scalability of ant algorithm. The ant algorithm for grid task scheduling is integrated into the simulation architecture and good results were obtained in terms of resource average utilization, response time and task fulfill proportion.

From the above research, ACS is the most popular variant of ACO that has been successfully used in grid computing environment to solve the scheduling problems which eventually reduce the stagnation problem. This is a fertile area of research for the improvement of grid resource management with new or enhanced ACS algorithm for job scheduling. This study proposed a new pheromone initializing process which is different from [22], where the consideration was only on the condition of the resource. The scheduling process in [18] has proposed resource with the lightest load to be assigned to new submitted job regardless of the job size. This study will consider assigning new submitted jobs to resources that are suitable based on the resource processing ability as well as the characteristics of the jobs.

III. Proposed ACO for Grid Load Balancing

This proposed algorithm aims to minimize the computational time of each job that must be processed by available resources in grid computing system. The algorithm will select the resources based on the pheromone value on each resource. A matrix that contains the pheromone value on each resource has been used to facilitate the selection of suitable resources to process submitted jobs.

The proposed algorithm has been implemented in the grid system architecture which consists of four main components namely the grid information server, grid resource broker, jobs and resources (refer Fig. 2). The algorithm works as follow:

1) User will send request to process a job. Details about the job such as the total number of jobs, size of each job, and CPU time needed by jobs will be included in the request.
2) Grid resource broker starts to calculate the relevant parameter to schedule the job after receiving the message from the user. The information server also provides the resource information to grid resource broker.
3) The largest entry in the pheromone value (PV) matrix will be selected by proposed technique as the resource to process the submitted job. A local pheromone update is performed after a job is assigned to a resource.
4) A global pheromone update is performed after a resource completed processing a job.
5) The execution results will be sent to the user.

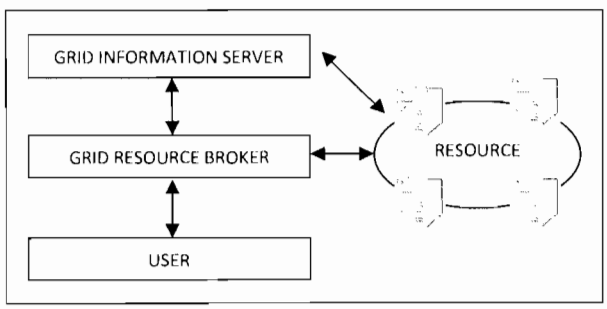

Figure 2. System Architecture

In this proposed algorithm, an ant represents a job in the grid system. The grid resource broker will find available resources from grid information server. Ant will move randomly in grid system and check the status of each resource. Pheromone value on a resource indicates the capacity of each resource in grid system. Pheromone value will be determined by two types of pheromone update technique which are local pheromone update in ACS [10] and global pheromone update in MMAS [19].

The initial pheromone value of each resource for each job is calculated based on the estimated transmission time and execution time of a given job when assigned to this resource. The estimated transmission time can be determined by $\frac{S_j}{bandwidth_r}$ where S_j is the size of a given job j and $bandwidth_r$ is the bandwidth available between the grid resource broker and the resource. The initial pheromone value is defined by:

$$PV_{ij} = \left[\frac{S_j}{bandwidth_r} + \frac{C_j}{MIPS_r * (1 - load_r)} \right]^{-1} \quad (1)$$

where PV_{ij} is the pheromone value for job j assigned to resource r, C_j is the CPU time needed of job j, $MIPS_r$ is the processer speed of resource r and $1 - load$ is the current load of resource r. The load, processor speed and bandwidth can be obtained from grid information server.

Assume there are n jobs and m resources in the PV matrix:

$$PV = \begin{array}{c} \\ r_1 \\ r_2 \\ .. \\ r_m \end{array} \begin{array}{c} \begin{array}{cccc} j_1 & j_2 & .. & j_n \end{array} \\ \begin{bmatrix} PV_{11} & PV_{12} & .. & PV_{1n} \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ PV_{m1} & PV_{m2} & .. & PV_{mn} \end{bmatrix} \end{array}$$

The largest entry from PV matrix which reflects the best resource, will be selected in each iteration. Assuming PV_{ij} is selected then job j will be processed by resource r. The local pheromone update is performed after job j has been assigned to resource r. This formula can only be applied to unassigned jobs in the PV matrix. The local pheromone update is formulated as follow:

$$\tau_{jr} = (1 - \xi).\tau_{jr} + \xi.\tau_0 \quad (2)$$

where ξ, $0 < \xi < 1$ and τ_0 are two parameters. The value of τ_0 is set to be the same as the initial value for the pheromone trails. A good value for ξ was found to be 0.1, while a good value for τ_0 was found to be $1/nC^{nn}$, where n is the number of resources and C^{nn} is the resource with high pheromone value. The effect of the local pheromone update is to ensure an already chosen resource less desirable

for the following ant [10] which will increase the exploration of not yet visited resource.

When a job is completely processed, global pheromone update is performed to recalculate the entire PV matrix. After all ants have constructed a solution, the pheromone trails are updated according to the following formula:

$$\tau_{jr}(t+1) = (1-\rho).\tau_{jr} + \rho\Delta\tau_{jr}^{bs} \qquad (3)$$

where $\Delta\tau_{jr}^{best} = 1/L^{best}$. This global pheromone update is limited to a specific upper and lower trail limit. The ant which is allowed to add pheromone may be the *iteration-best solution* or *global best solution*. If a specific resource is often used in the best solution, it will receive a larger amount of pheromone and stagnation will occur. So, lower and upper limits on the possible pheromone strengths on any resource are imposed to avoid stagnation. The imposed trails limits have the effects of limiting the probability ρ_{iu} of selecting resource u when ants is in resource i to an interval $[p_{min}, p_{max}]$, with $0<p_{min}\leq p_{ij}\leq p_{max}\leq 1$. With this minimum trail limit, the resource is less desire to be selected by the jobs since it will select the resource that has the upper trail limit.

For example, there are three jobs (j_1, j_2, and j_3) that need to be processed by three resources (r_1, r_2, and r_3) in the grid system. The initial status of each resource is shown in Table I and size of each job is 5MB, 3MB and 1MB. The CPU cycles needed for each job are 5M, 3M and 1M respectively.

TABLE I. STATUS OF EACH RESOURCE

Status	r_1	r_2	r_3
Processor Speed (MIPS)	217	464	195
Load	15%	10%	20%
Bandwidth (Megabits/s)	10.62	24.50	12.62

The initial pheromone values of each entry obtained from equation (1) are shown in the following PV matrix:

$$PV = \begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \end{array} \begin{array}{c} \begin{array}{ccc} j_1 & j_2 & j_3 \end{array} \\ \begin{bmatrix} PV_{11}=2.01 & PV_{12}=3.35 & PV_{13}=10.04 \\ PV_{21}=4.63 & PV_{22}=7.71 & PV_{23}=23.14 \\ PV_{31}=2.34 & PV_{32}=3.89 & PV_{33}=11.68 \end{bmatrix} \end{array}$$

The resource with high pheromone value will be selected by grid resource broker. So j_3 will be processed by r_2. After assigning j_3 to r_2, the local pheromone update is performed to the second row of r_2. Column 3 is no longer needed because j_3 has been assigned. The new PV matrix is as follows:

Local update

$$PV = \begin{bmatrix} PV_{11}=2.01 & PV_{12}=3.35 \\ PV_{21}=4.17 & PV_{22}=6.94 \\ PV_{31}=2.34 & PV_{32}=3.89 \end{bmatrix}$$

After r_2 finished processing j_3, the global pheromone update is performed to get the newest pheromone value for the next job submission. The newest status of each resource after the execution of j_3 is as shown in Table II. The load status of each resource will be changed according to the size of the current load. On the other hand, the ρ value is used in evaporation process.

TABLE II. UPDATE STATUS OF EACH RESOURCE

Status	r_1	r_2	r_3
Processor Speed (MIPS)	217	464	195
Load	15%	25%	20%
Bandwidth (Megabits/s)	8.67	15.87	10.26
ρ	0.00	0.05	0.00

The ρ value of r_2 is 0.05 and ρ values for r_1 and r_3 are zero since they have not been assigned any job for execution. The new PV matrix is as follows:

$$PV = \begin{bmatrix} PV_{11}=1.64 & PV_{12}=2.73 \\ PV_{21}=2.88 & PV_{22}=4.81 \\ PV_{31}=1.93 & PV_{32}=3.22 \end{bmatrix}$$

The remaining job will be assigned in the same way. The local pheromone update will be performed after a grid resource broker assigned a job to a resource. After a resource finished processing a job, all entries of the PV matrix will be updated by the global pheromone update rules.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

The performance of the proposed algorithm has been compared to other algorithms such as particle swarm, space shared and time shared. The average completion time has been used to compare the performance of the algorithms.

Two experiments have been conducted to evaluate the performance of the proposed algorithm in terms of their processing time. First experiment processed 10 jobs with 100 resources while the second experiment processed 100 jobs with 1000 resources. Details of the scheduling parameters are shown in the Table 3.

TABLE III. SCHEDULING PARAMETERS FOR THE EXPERIMENTS

Experiment	1	2
No. of machine per resource	1	1
Number of PEs per machine	1 – 5	1 – 5
PE ratings	10 or 50 MIPS	10 or 50 MIPS
Bandwidth	1000 or 5000	1000 or 5000

	B/S	B/S
Number of Gridlet	100	1000
Number of Resource	10	100

Fig. 3 depicts the comparison between the average completion times of algorithms for the first experiment. It can be seen that the proposed algorithm labeled as Enhanced ACO has the lowest average completion time as compared to Particle Swarm, Space Shared and Time Shared. This is expected because Enhanced ACO keeps track of the state of all the resources at all time. Enhanced ACO performs better than Particle Swarm (the second best algorithm) by a factor of 1.78.

A comparison between the average completion times for the second experiment is shown in Fig. 4. Enhanced ACO still performs better than the other algorithms. Enhanced ACO performs better than Particle Swarm by a factor of 2.41. From this comparison, Enhanced ACO performs better in the second scenario compared to the first scenario. This is because the communication that occur when each ant taking a step for searching the best resource is less when processing a large amount of jobs. Each ant will carry the history of visited node and will refer to it to find the best resources to process jobs.

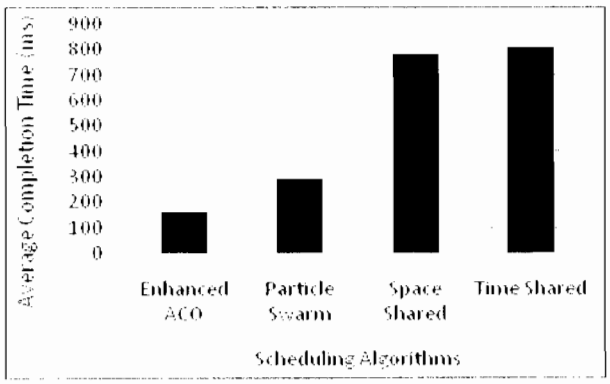

Figure 3. Comparison among the scheduling algorithms for the first scenario.

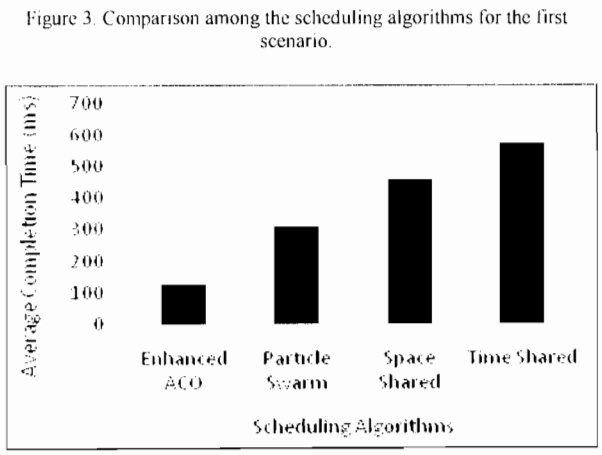

Figure 4. Comparison among the scheduling algorithms for the second scenario.

V. CONCLUSION

The scheduling process in the proposed algorithm is based on the combination of local pheromone update and trail limits. This proposed technique is different from the previous algorithm based on its initial pheromone value and the used of the PV matrix. The initial pheromone value for this algorithm considered the estimated transmission time and characteristics of each jobs and resources. The local pheromone trail update will reduce the amount of pheromone in assigned resource, to ensure the resource is less desirable for other ants while the trail limit, which is the allowed range of the pheromone strength, is limited to maximum and minimum trail strength. This is a technique to control the value of pheromone updated on each resource to ensure that already assigned optimal resources will not be chosen for newly submitted jobs. The proposed algorithm is simple to be implemented due to the existing of information of each resources and jobs. This algorithm was able to minimize the completion time of each job.

REFERENCES

[1] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," presented at Proceedings of the First European Conference on Artificial Life, Paris, France, Amsterdam: Elsevier Publishing, pp. 134-142, 1991.

[2] G. Pavani and H. Waldman, "Grid resource management by means of ant colony optimization," 3rd International Conference on Broadband Communications, Networks and Systems (BROADNETS), 2006.

[3] H. Singh and A. Youssef, "Mapping and scheduling heterogeneous task graphs using genetic algorithms," presented at 5th IEEE Heterogeneous Computing Workshop (HCW'96), 1996.

[4] H. Yan, X. Shen, X. Li, and M. Wu, "An improved ant algorithm for job scheduling in grid computing," presented at Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, vol. 5, pp. 2957-2961, 2005.

[5] Foster and C. Kesselman, *The grid: blueprint for a new computing infrastructure*. San Francisco: Morgan Kaufmann, 2004.

[6] K. Yan, S. Wang, S. Wang, and C. Chang, "Towards a hybrid load balancing policy in grid computing system," *Expert Systems with Applications*, vol. 36, pp. 12054-12064, 2009.

[7] K. Yang, X. Guo, A. Galis, B. Yang, and D. Liu, "Towards efficient resource on-demand in grid computing," *ACM SIGOPS Operating Systems Review*, vol. 37(2), pp. 37-43, 2003.

[8] L. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," presented at Proceedings of the IEEE Conference on Evolutionary Computation, Nagoya, Japan (ICEC96), pp. 622-627, 1996.

[9] M. Chtepen, "Dynamic scheduling in grids system," Sixth Firw PhD Symposium, Faculty of Engineering, Ghent University, pp 110, 2005.

[10] M. Dorigo and T. Stützle, *Ant colony optimization, Cambridge, Massachusetts, London, England*: MIT Press, 2004.

[11] M. Dorigo, "Optimization, learning and natural algorithms," *Ph. D. dissertation, Politecnico di Milano, Milan, Italy, 1992*.

[12] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: An autocatalytic optimizing process," *Thechnical Report 91-016 Revised, Dipartimento di Electronica, Politecno di Milano, 1991*.

[13] O. Ibarra and C. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," *Journal of the ACM (JACM)*, vol. 24(2), pp. 280-289, 1977.
[14] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," presented at 7th IEEE Heterogeneous Computing Workshop (HCW'98), 1998.
[15] R. Chang, J. Chang, and P. Lin, "Balanced Job Assignment Based on Ant Algorithm for Grid Computing," presented at Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference, pp. 291-295, 2007.
[16] R. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, and J. Lima, "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet," presented at 7th IEEE Heterogeneous Computing Workshop (HCW'98), 1998.
[17] S. Fidanova and M. Durchova, "Ant algorithm for grid scheduling problem," *Lecture Notes in Computer Science*, vol. 3743, pp. 405-412, 2006.
[18] S. Lorpunmanee, M. Sap, A. Abdullah, and C. Chompoo-inwai, "An ant colony optimization for dynamic job scheduling in grid environment," *International Journal of Computer and Information Science and Engineering*, vol. 1(4), pp. 207-214, 2007.
[19] T. Stutzle and H. Hoos, "MAX-MIN ant system," *Future Generation Computer Systems*, vol. 16, pp. 889-914, 2000.
[20] V. Naik, P. Garbacki, K. Kummamuru, and Y. Zhao, "On-line evolutionary resource matching for job scheduling in heterogeneous grid environments," *Proceedings of the 12th International Conference on Parallel and Distributed Systems(ICPADS'06), 2006*.
[21] X. Bai, H. Yu, Y. Ji, and D. Marinescu, "Resource matching and a matchmaking service for an intelligent grid," *International Journal of Computational Intelligence*, vol. 1(3), pp. 197-205, 2004.
[22] Y. Li, "A Bio-inspired Adaptive Job Scheduling Mechanism on a Computational Grid," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 6(3), pp. 1-7, 2006.
[23] Z. Xu, X. Hou, and J. Sun, "Ant algorithm-based task scheduling in grid computing," presented at Electrical and Computer Engineering IEEE CCECE, Canadian Conference, pp. 1107-1110, 2003.