International Journal of Computer Applications Technology and Research
Volume 5–Issue 1, 15-19, 2016, ISSN:2319–8656

Presenting a New Ant Colony Optimization Algorithm (ACO) for Efficient Job Scheduling in Grid Environment

Firoozeh Ghazipour
Department of Computer
Science and Research Branch
Islamic Azad University, Kish, Iran

Seyyed Javad Mirabedini
Department of Computer
Islamic Azad University
Central Tehran Branch

Ali Harounabadi
Department of Computer
Islamic Azad University
Central Tehran Branch

Abstract: Grid computing utilizes the distributed heterogeneous resources in order to support complicated computing problems. Job scheduling in computing grid is a very important problem. To utilize grids efficiently, we need a good job scheduling algorithm to assign jobs to resources in grids.
In the natural environment, the ants have a tremendous ability to team up to find an optimal path to food resources. An ant algorithm simulates the behavior of ants. In this paper, a new Ant Colony Optimization (ACO) algorithm is proposed for job scheduling in the Grid environment. The main contribution of this paper is to minimize the makespan of a given set of jobs. Compared with the other job scheduling algorithms, the proposed algorithm can outperform them according to the experimental results.

Keywords: jobs, scheduling, Grid environment, Ant Colony Optimization (ACO), makespan.

1. INTRODUCTION

Current scientific problems are very complex and need huge computing power and storage space. The past technologies such as distributed or parallel computing are unsuitable for current scientific problems with large amounts of data. Processing and storing massive volumes of data may take a very long time.
Grid computing [1] is a new paradigm for solving those complex problems. In grids, we need to consider the conditions such as network status and resources status. If the network or resources are unstable, jobs would be failed or the total computation time would be very large. So we need an efficient job scheduling algorithm for these problems in the grid environment.
The purpose of job scheduling is to balance the entire system load while completing all the jobs at hand as soon as possible according to the environment status. Because the environment status may change frequently, traditional job scheduling algorithm may not be suitable for the dynamic environment in grids.
In grids, users may face hundreds of thousands of computers to utilize. It is impossible for anyone to manually assign jobs to computing resources in grids. Therefore, grid job scheduling is a very important issue in grid computing. Because of its importance, importance, many job scheduling algorithms for grids [2-5] have been proposed.
A good schedule would adjust its scheduling strategy according to the changing status of the entire environment and the types of jobs. Therefore, a dynamic algorithm in job scheduling such as Ant Colony Optimization (ACO) [6, 7] is appropriate for grids.
ACO is a heuristic algorithm with efficient local search for combinatorial problems. ACO imitates the behavior of real ant colonies in nature to search for food and to connect to each other by pheromone laid on paths traveled. Many researches use ACO to solve NP-hard problems such as traveling salesman problem [8], graph coloring problem [9], vehicle routing problem [10], and so on.
This paper applies the ACO algorithm to job scheduling problems in Grid computing. We assume each job is an ant and the algorithm sends the ants to search for resources. We also modify the global and local pheromone update functions in ACO algorithm in order to balance the load for each grid resource. Finally, we compare the proposed ACO algorithm with Min-Min [11] and Max-Min [12]. According to the experimental results, we can find out that our proposed ACO algorithm is capable of achieving system load balance and decreasing the makespan better than other job scheduling algorithms. The rest of the paper is organized as follows. Section 2 explains the background of ACO algorithm and scheduling problem. Section 3 introduces some related work. Section 4 details the proposed ACO algorithm for job scheduling. Section 5 indicates the experimental results and finally, Section 6 concludes the paper.

2. BACKGROUND

2.1 Characteristics of ACO

Dorigo introduced the ant algorithm [18], which is a new heuristic algorithm and based on the behavior of real ants. When the blind insects, such as ants look for food, the moving ant lays some pheromone on the ground, thus marking the path it followed by a trail of this substance. While an isolated ant moves essentially at random, an ant encountering a previously laid trail can detect it and decide with high probability to follow it, thus reinforcing the trail with its own pheromone. The collective behavior that emerges means where the more are the ants following a trail, the more that trail becomes attractive for being followed. The process is thus characterized by a positive feedback loop, where the probability with which an ant chooses an optimum path increases with the number of ants that chose the same path in the preceding steps. Above observations inspired a new type of algorithm called ant algorithms or ant systems, which is presented by Dorigo and Gambardella [19]. The ACO algorithm uses a colony of artificial ants that behave as co-operative agents in a mathematical space were they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. Solution that satisfies the constraints is feasible. All ACO algorithms adopt specific algorithmic scheme which is shown in Figure 1.

```
Procedure ACO
    Parameter Initialization
    While (stopping criterion not satisfied) do
        Construct solutions
        Local search
        Update statistics
        Update pheromone trails
    End while
```

Figure 1. All ACO Algorithm scheme.

We utilize the characteristics of ant algorithms above mentioned to schedule job. We can carry on new job scheduling by experience depend on the result in the past job scheduling. It is very helpful for being within the grid environment.

2.2 Scheduling Problem Formulation

The grid environment consists of a large number of resources and jobs which should be assigned to the resources. The jobs cannot divide into smaller parts and after assigning a job to a resource, its executing cannot be stopped.

The main challenge in scheduling problems is time. Finding a solution in these problems tries to decrease the time of executing all jobs. In this case, the most popular criterion is makespan and our purpose in this paper is reducing the makespan with the aid of ACO.

In grid, we have a set of resources (Resources = {m_1, m_2, …, m_m}) and a set of jobs (Jobs = {t_1, t_2, …, t_n}) which should be assigned to the resources and executed on them. There is a matrix ETC [Resources] [Jobs] (Figure 2) that represents the time of executing (EX) t_i on m_j.

$$\begin{array}{c|cccc|}
 & Job_1 & Job_2 & \cdots & Job_n \\
\hline
Resource_1 & EX_{11} & EX_{12} & \cdots & EX_{1n} \\
Resource_2 & EX_{12} & EX_{22} & \cdots & EX_{2n} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
Resource_m & EX_{m1} & EX_{m2} & \cdots & EX_{mn}
\end{array}$$

Figure 2. Matrix ETC

Suppose that $E_{ij\ (i\,\epsilon\,Jobs\ ,\ j\,\epsilon\,Resources)}$ is the time of job i on resource j and $W_{j\ (j\,\epsilon\,Resources)}$ is the time of executing jobs which are assigned to resource j before. Then equation (1) shows the time of executing all jobs which are allocated to m_j.

$$\sum\nolimits_{\forall\ Job\ is\ allocated\ to\ Resource\ j} \left(E_{ij}\ + W_j\right) \tag{1}$$

We can find the value of makespan according to equation (2):

$$\text{makespan} = \max \{\sum\nolimits_{\forall\ job\ is\ allocated\ to\ resource\ j} \left(E_{ij}\ + W_j\right)\}\ , i \in Jobs\ and\ j \in Resources \tag{2}$$

With another look on these definitions, we can calculate the completion time of resources with equation (3):

$$CT_{ij}\ = E_{ij}\ + W_j \tag{3}$$

There is a Scheduling List for all resources that shows the jobs which are assigned to each resource. Each resource has a completion time and according to equation (4), the value of makespan is equal to the maximum completion time.

$$\text{makespan} = max\ _{(i,j)\ \in\ Scheduling\ List}\ (CT_{ij}\ ) \tag{4}$$

The makespan is a criterion to evaluate the grid system and the main purpose of a scheduler is to minimize this criterion.

3. RELATED WORK

Jobs submitted to a grid computing system need to be processed by the available resources.

Best resources are categorized as optimal resources. In a research by [13], Ant Colony Optimization (ACO) has been used as an effective algorithm in solving the scheduling problem in grid computing.

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 [6]. A recent approach of ACO researches in the use of ACO for scheduling job in grid computing [14]. 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 [16] 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 to improved ant algorithm for job scheduling in grid computing which is based on the basic idea of ACO was proposed by [17]. 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.

4. THE PROPOSED ALGORITHM

Real ants foraging for food lay down quantities of pheromone (chemical substance) marking the path that they follow. An isolated ant moves essentially at random but an ant encountering a previously laid pheromone will detect it and decide to follow it with high probability and thereby reinforce it with a further quantity of pheromone. The repetition of the above mechanism represents the auto catalytic behavior of real ant colony where the more the ants follow a trail, the more attractive that trail becomes [13].

The ACO algorithm uses a colony of artificial ants that behave as co-operative agents in a mathematical space were they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. Solution that satisfies the constraints is feasible. After initialization of the pheromone trails, ants construct feasible solutions, starting from random nodes, and then the pheromone trails are updated. At each step ants compute a set of feasible moves and select the best one (according to some probabilistic rules) to carry out the rest of the tour. The transition probability is based on the heuristic information and pheromone trail level of the move. The higher value of the pheromone and the heuristic information, the more profitable it is to select this move and resume the search.

4.1 Initialize Pheromone and Probability List

At the beginning, a population of ants is generated and they start their own search from one of the resource (the ants assigned to resources randomly). The initial Pheromone and Probability List is set to a small positive value t_0 and then ants update this value after completing the construction stage. In the nature there is not any pheromone on the ground at the beginning, or the initial pheromone in the nature is $t_0 = 0$. If in ACO algorithm the initial Pheromone is zero, then the probability to choose the next state will be zero and the search process will stop from the beginning. Thus it is important to set the initial Pheromone and Probability List to a positive value. The value of t0 is calculated with equation (5):

$$t_0 = \frac{1}{Jobs \times Resources} \qquad (5)$$

4.2 Local Pheromone Update

Moving from a state to another means that a job is assigned to a resource. After choosing next node by ants, the pheromone trail should be updated. This update is local pheromone update and the equation (6) shows how it happens:

$$\text{Pheromone}_{ij}^{k} = \left((1-\varepsilon) \times \left(\text{Pheromone}_{ij}^{k}\right)\right) + (\varepsilon \times \theta) \qquad (6)$$

In local pheromone update equation, θ is a coefficient which obtains its value from equation (7):

$$\theta = \frac{t_0}{CT_{ij}(Ant_k)} \qquad (7)$$

The less value of CT_{ij}, the more value of θ. In fact, if the value of θ is larger, the more pheromone value will be deposited. Therefore, the chance of choosing resource j in next assigning is more than other resources.

4.3 Probability List Update

In addition to update Pheromone, the Probability List should be updated, too. The ants choose the next states based on heuristic information, equation (8):

$$\text{Heuristic}_{ij}^{k} = \frac{1}{(W_j) \times (ETC_{ij})} \qquad (8)$$

With the heuristic information, we can update the Probability List, equation (9):

$$\text{Probability List}_{ij}^{k} = (\text{Pheromone}_{ij}^{k}) \times (\text{Heuristic}_{ij}^{k})^{\beta} \qquad (9)$$

4.4 Global Pheromone Update

In the nature, some pheromone value on the trails evaporates. At the end of each iteration in the proposed algorithm, when all ants finish the search process, the all ants' value of pheromone will be reduced by evaporation rule, equation (10):

$$\text{Pheromone}_{ij}^{k} = (\text{Pheromone}_{ij}^{k}) \times (1-\rho) \qquad (10)$$

When all ants construct a solution, it means that the ants moves from the nest to the food resource and finish the search process (all the jobs are assigned to the resources in grid). In the proposed algorithm, the best solution and the best ant which construct that solution will be found. The global pheromone update is just for the ant that finds the best solution. This ant is the best ant of iteration. At this stage, the value of Pheromone should be updated, equation (11):

$$\text{Pheromone}_{ij}^{Best\ Ant} = \text{Pheromone}_{ij}^{Best\ Ant} + \left((\rho) \times (\Delta)\right) + \frac{\varepsilon}{makespan\ (Best\ Ant)} \qquad (11)$$

In global pheromone update, ρ is the elitism coefficient and Δ is calculated by equation (12):

$$\Delta = \frac{1}{makespan\ (Best\ Ant)} \qquad (12)$$

The pseudo-code of the proposed algorithm is presented in Figure 3.

5. EXPERIMENTAL RESULTS

The results of the evaluation of the proposed algorithm with the two algorithms of Min-Min and Max-Min [11] for scheduling independent jobs in grid environment are presented in this section.

All experiments have been done on a system running Windows 7 Professional operating system with configuration of 2 GHz CPU and 2GB of RAM.

```
The Proposed ACO Algorithm for Job Scheduling in Grid
Parameters Initialization (Table 1)
Create Ant Population
Initialize the Pheromone and Probability Information (Formula 5)
While (stopping criterion not satisfied) do
  Construct Solution Phase
    For all Ants do
      Place all Ants in random chosen Resource
      Apply Local Update for (Ant, Job, Resource) (Formula 6, 9)
    End For
    While (all Jobs not assigned) do
      Apply Decision Rules for (Ant, Job)
      Apply Local Update for (Ant, Job, Resource) (Formula 6, 9)
    End While
  End of Construct Solution Phase
  Global Pheromone Update Phase
    Evaporate (Formula 10)
    Find and Update best Ant of Iteration and best-so-far
    Deposit Pheromone (Formula 11)
  End of Global Pheromone Update Phase
  Find best solution "best makespan"
  Reset the knowledge of all Ants
End While
```

Figure 3. Pseudo-code of the proposed algorithm.

Table 1 indicates the amounts of parameters which are used in executing the proposed algorithm.

Table 1. Parameters of the proposed algorithm.

Number of resources	16
Number of jobs	512
Number of ants	100
Number of iterations	1000
ε	0.1
β	2 (between 2 and 5)
ρ	0.2

A real heterogeneous computational system such as grid is a combination of hardware and software elements and a comparison of the scheduling techniques is often complicated in this environment. To solve this problem, Braun et al. [20], proposed a simulation model. They defined a grid environment which consists of a set of resources and a set of independent jobs. The scheduling algorithms aim to minimize the makespan. All scheduling algorithms need to know the completion time of each job on each resource. The model consists of 12 different kinds of examples: u_c_hihi, u_c_hilo, u_c_lohi, u_c_lolo, u_i_hihi, u_i_hilo, u_i_lohi, u_i_lolo, u_s_hihi, u_s_hilo, u_s_lohi, u_s_lolo; that any of them can be shown in a matrix. This model uses a matrix ETC which illustrates the estimated times of completion (Figure 2).
In this paper, the same model is used to evaluate the proposed algorithm and the two scheduling algorithms, Min-Min and Max-Min. After executing these three algorithms, different amounts of makespan are obtained which are shown in Table 2.

Table 2. Comparison of three algorithms' makespans.

	The Proposed Algorithm	Min-Min	Max-Min
u_c_hihi	8291837.50	8460674.00	12385672.00
u_c_hilo	158629.16	161805.42	204054.58
u_c_lohi	265301.53	275837.34	392566.69
u_c_lolo	5388.70	5441.43	6945.36
u_i_hihi	3085218.25	3513919.25	8018377.50
u_i_hilo	77190.69	80755.68	151923.84
u_i_lohi	108951.52	120517.71	251528.84
u_i_lolo	2617.01	2785.65	5177.71
u_s_hihi	4776037.50	5160343.00	9208811.00
u_s_hilo	102499.90	104375.17	172822.69
u_s_lohi	134873.02	140284.50	282085.69
u_s_lolo	3590.33	3806.83	6232.24

The Results of experiment are shown as a chart in Figure 4, 5, 6 and 7.

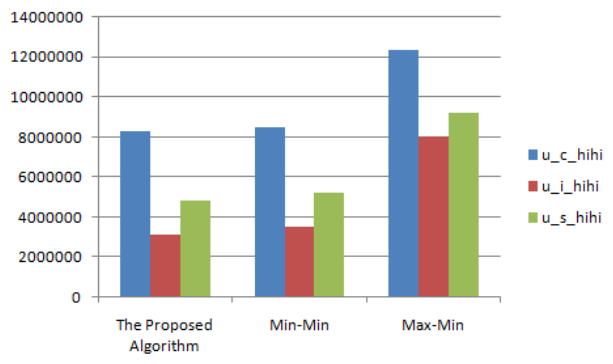

Figure 4. Algorithms' makespans based on u-*-hihi.

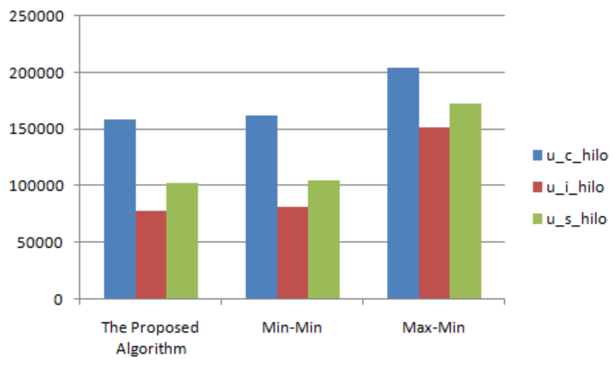

Figure 5. Algorithms' makespans based on u-*-hilo.

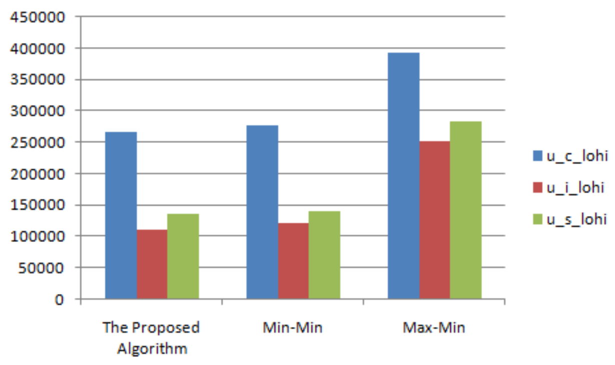

Figure 6. Algorithms' makespans based on u-*-lohi.

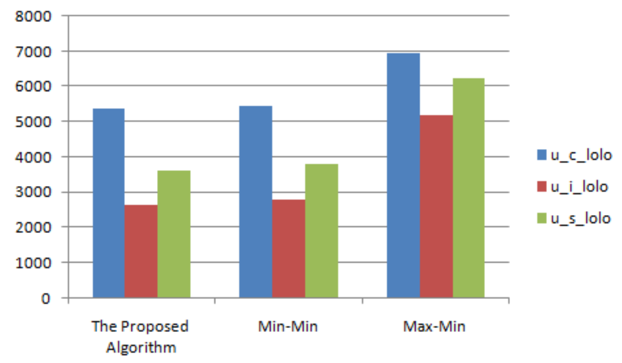

Figure 7. Algorithms' makespans based on u-*-lolo.

The two scheduling algorithms, Min-Min and Max-Min, are two best algorithms among other scheduling algorithms but the results of the experiment (Table 2) indicate that the proposed algorithm has higher performance and lower amount of makespan than the other two scheduling algorithms.

6. CONCLUSIONS

In this paper, a new ACO algorithm is proposed to choose suitable resources to execute jobs according to the completion times of resources and the size of given job in the grid environment. The local and global pheromone update functions are changed to do balance the system load. Local pheromone update function updates the status of the selected resource after jobs assignment. Global pheromone update function updates the status of scheduling list of best solution. The purpose of this paper is to minimize the makespan and the experimental results show that the proposed algorithm is capable of minimizing the makespan better than other two scheduling algorithms.

7. REFERENCES

[1] Reed, D. A., "Grids, the TeraGrid and beyond", IEEE, Vol. 36, Issue 1, 2003.

[2] Chang, R. S., Chang, J. S., Lin, P. S., "An ant algorithm for balanced job scheduling in grids", Future Generation Computer Systems, Vol. 25, pp. 20-27, 2009.

[3] Ku-Mahamud, K. R., Nasir, H. J. A., "Ant Colony Algorithm for Job Scheduling in Grid Computing", Fourth Asia International Conference on Mathematical/Analytical Modeling and Computer Simulation, pp. 40-45, 2010.

[4] Chang, R. S., Chang, J. S., Lin, S. Y., "Job scheduling and data replication on data grids", Future Generation Computer Systems, Vol. 23, Issue 7, pp. 846-860, 2007.

[5] Gao, Y., Rong, H., Huang, J. Z., "Adaptive grid job scheduling with genetic algorithms", Future Generation Computer Systems, Vol. 21, Issue 1, pp. 151-161, 2005.

[6] Dorigo, M., Stutzle, T., "Ant Colony Optimization", A Bradford Book, 2004.

[7] Dorigo, M., Blum, C., "Ant colony optimization theory: A survey", Theoretical Computer Science, Vol. 344, Issue 2, pp. 243-278, 2005.

[8] Dorigo, M., Gambardella, L. M., "Ant colony system: a cooperative learning approach to the traveling salesman problem", IEEE Transaction on Evolutionary Computation, Vol. 1, Issue 1, pp. 53-66, 1997.

[9] Salari, E., Eshghi, K., "An ACO algorithm for graph coloring problem", ICSC Congress on Computational Intelligence Methods and Applications, 2005.

[10] Zhang, X., Tang, L., "CT-ACO-hybridizing ant colony optimization with cyclic transfer search for the vehicle routing problem", ICSC Congress on Computational Intelligence Methods and Applications, 2005.

[11] Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., Freund, R. F., "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems", Journal of Parallel and Distributed Computing, pp. 107-131, 1999.

[12] Casanova, H., Legrand, A., Zagorodnov, D., Berman, F., "Heuristics for scheduling parameter sweep applications in grid environments", Heterogeneous Computing Workshop, pp. 349-363, 2000.

[13] Fidanova, S., Durchova, M., "Ant Algorithm for Grid Scheduling Problem", Springer, pp. 405-412, 2006.

[14] Pavani, G. S., Waldman, H., "Grid Resource Management by means of Ant Colony Optimization", 3rd International Conference on Broadband Communications, Networks and Systems, pp. 1-9, 2006.

[15] Chang, R. S., Chang, J. S., Lin, P. S., "Balanced Job Assignment Based on Ant Algorithm for Computing Grids", The 2nd IEEE Asia-Pacific Service Computing Conference, pp. 291-295, 2007.

[16] Lorpunmanee, S., Sap, M. N., Abdullah, A. H., Chompoo-inwai, C., "An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment", Proceedings of World Academy of Science, Engineering and Technology, Vol. 23, pp. 314-321, 2007.

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

[18] Dorigo, M., Maniezzo, V., Colorni, A., "Ant system: optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man and Cybernetics - Part B, Vol. 26, No. 1, pp. 1-13, 1996.

[19] Dorigo, M., Gambardella, L. M., "Ant colonies for the traveling salesman problem", Biosystems, Vol. 43, Issue 2, pp. 73-81, 1997.

[20] Braun, T. D., Siegel, H. J., Beck, N., "A Comparison of Eleven static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing systems", Journal of Parallel and Distributed Computing, Vol. 61, pp. 810-837, 2001.