International Journal of Computer Applications Technology and Research
Volume 5–Issue 1, 20-25, 2016, ISSN:2319–8656

Proposing a new Job Scheduling Algorithm in Grid Environment Using a Combination of Ant Colony Optimization Algorithm (ACO) and Suffrage

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, Iran

Ali Harounabadi
Department of Computer
Islamic Azad University
Central Tehran Branch,Iran

Abstract: Scheduling jobs to resources in grid computing is complicated due to the distributed and heterogeneous nature of the resources. The purpose of job scheduling in grid environment is to achieve high system throughput and minimize the execution time of applications. The complexity of scheduling problem increases with the size of the grid and becomes highly difficult to solve effectively.
To obtain a good and efficient method to solve scheduling problems in grid, a new area of research is implemented. In this paper, a job scheduling algorithm is proposed to assign jobs to available resources in grid environment. The proposed algorithm is based on Ant Colony Optimization (ACO) algorithm. This algorithm is combined with one of the best scheduling algorithm, Suffrage. This paper uses the result of Suffrage in proposed ACO algorithm. The main contribution of this work is to minimize the makespan of a given set of jobs. The experimental results show that the proposed algorithm can lead to significant performance in grid environment.

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

1. INTRODUCTION

Distributed systems consist of multiple computers that communicate through computer networks. Research by [1] 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 [2] defined that grid computing is based on large scale resources sharing in a widely connected network such as the Internet.
The past technologies such as cluster and parallel computing do not suit current scientific problems with a large amount of data files [3]. Especially, processing and storing massive volumes of data may take a very long time. Besides, we need to consider about the other 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.
How to schedule jobs efficiently in a grid environment is a main issue. The purpose of job scheduling is to balance the entire system load and minimize the completion time according to the environment status. A good scheduler would adjust its scheduling strategy according to the changing status of the entire environment and types of jobs.
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. Computational time is a measure of how long that resource takes to complete the job. An effective job scheduling algorithm is needed to reduce the computational time of each resource and also minimize the makespan of the system. Therefore, an algorithm in job scheduling such as Ant Colony Optimization (ACO) [4] is appropriate for the problems mentioned above.
ACO is a heuristic algorithm with efficient local search for combinatorial problems. ACO imitates the behavior of real ant colonies in nature to search food from nest and interact with each other by pheromone value which is laid on paths. Many researches use ACO to solve NP-hard problems such as traveling salesman problem [5], graph coloring problem [6], vehicle routing problem [7], and so on.
One of the best scheduling algorithms which used to schedule jobs in grid environment is Suffrage [8]. This paper combines the ACO and Suffrage to schedule a given set of jobs 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 combinatorial algorithm with Min-Min [9], Max-Min [9] and the Suffrage itself [8]. 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 and Suffrage. Section 3 introduces some related work. Section 4 details the proposed ACO algorithm for job scheduling in grid environment. Section 5 indicates the experimental results and finally, Section 6 concludes this paper.

2. BACKGROUND

2.1 Features of ACO

Dorigo introduced the ant algorithm [10], 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 [11]. 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.

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

Figure 2. Matrix ETC

Suppose that $E_{ij\ (i \in Jobs\ ,\ j \in Resources)}$ is the time of job i on resource j and $W_{j\ (j \in Resources)}$ 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.

$$\sum_{\forall\ Job\ is\ allocated\ to\ Resource\ j} (E_{ij} + W_j) \quad (1)$$

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

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

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

$$CT_{ij} = E_{ij} + W_j \quad (3)$$

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.

$$\text{makespan} = max_{\ (i,j) \in Scheduling\ List} (CT_{ij}) \quad (4)$$

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

2.3 Suffrage

Suffrage is one of the job scheduling algorithms that schedules the given set of jobs in grid environment. The idea behind *Suffrage* [9] is that a resource is assigned to a job that would "suffer" the most if that resource would not be assigned to it. The suffrage value of a job ($Difference_{ij}$) is defined by the difference between its second best completion time (Second Minimum) and its best completion time (First Minimum), equation 5:

$$Difference_{ij}\ (t_i) = [\text{First Minimum}_{ij} - \text{Second Minimum}_{ij}] \quad (5)$$

The suffrage value ($Difference_{ij}$) of all jobs in the given set of jobs is calculated and stored in a Difference List. Then, a job is possibly assigned to the resource that has the most suffrage value (has the most Priority), equation 6:

$$\text{Priority} = max_{(i,j) \in Difference\ list} (Difference_{ij}\ (t_j)) \quad (6)$$

If another job was previously assigned to the resource, the suffrage values of the job previously assigned and of the new job are compared. The job would be executed that has the greater suffrage value and the other one would be assigned

later. Every time a job finishes, all jobs that have not started yet are unscheduled and the algorithm is invoked again, using current values of suffrage. Consequently, the algorithm runs again and schedules the remaining jobs, but this time with the new load of the resources. This scheme is repeated until all jobs are assigned to the available resources and completed [12].

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.

The study by [14] 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.

Stutzle, T. proposes Max-Min Ant System (MMAS) [15] to limit the pheromone range to be greater than or equal to the low bound value (Min) and smaller than or equal to the upper bound value (Max) to avoid ants to converge too soon in some ranges.

M. Dorigo et al propose Fast Ant System (FANT) [16]. It uses one ant at each iteration and gets the solution of the ant to do a local search. FANT works without evaporation rule and it updates pheromone after each iteration. In order to avoid the sub-optimal solution, it applies the reset pheromone function.

Hui Yan et al [17] apply the basic idea of ACO, but change the pheromone update function by adding encouragement, punishment coefficient and local balancing factor. The initial pheromone value of each resource is based on its status. For example, the number of CPU, CPU speed and bandwidth will take into account on the initial pheromone value. They assign job to the resource with the maximum pheromone value and the pheromone of each resource will be update by update function. 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, it will be added more pheromone by the encouragement coefficient in order to be selected for next job execution. If a resource failed to complete a job, it will be punished by adding less pheromone value. They take the load of each resource into account and also apply the balancing factor to change the pheromone value of each resource.

Kwang Mong Sim et al. [18] use multiple kinds of ant to find multiple optimal paths for network routing. The idea can be applied to find multiple available resources to balance resources utilization in job scheduling. The key of the idea is each different kinds of ant can only sense their own kind of pheromone so that it can find much different paths including the shortest-path by different kinds of ant. Its main problem is that if all kinds of ant find the same path, it will do nothing like using one kind of ant. And how to compare their performance of each kind of ant creates another problem. Furthermore, one solution from this algorithm may work efficiently in an environment, but it may work inefficiently in another one.

J. Heinonen et al. [19] apply the hybrid ACO algorithm with different visibility to job-shop scheduling problem. The hybrid ACO algorithm consists of two ideas. One idea is the basic ACO algorithm, and the other idea uses the post-processing algorithm in the part of local search in ACO algorithm. When the ACO algorithm finished, all ants complete its own tour which can be decomposed into blocks. The block for swap must contain more than two operations. Then the post processing algorithm uses the swap operation on the blocks. If the swap reforms the makespan, the new path is accepted; otherwise the swap is invalid and the swapped block recovers to previous status.

Balanced job assignment based on ant algorithm for computing grids called BACO was proposed by [3]. 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 [20] 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.

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 randomly but an ant encountering a previously laid pheromone will detect it and decide to follow it with high probability and thereby reinforce the path 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 (7):

$$t_0 = \frac{1}{Jobs \times Resources} \quad (7)$$

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 (8) shows how it happens:

$$\text{Pheromone}_{ij}^{k} = \left((1-\varepsilon) \times \left(\text{Pheromone}_{ij}^{k}\right)\right) + (\varepsilon \times \theta) \quad (8)$$

In local pheromone update equation, θ is a coefficient which obtains its value from equation (9):

$$\theta = \frac{t_0}{CT_{ij}(Ant_k)} \quad (9)$$

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 (10):

$$\text{Heuristic}_{ij}^{k} = \frac{1}{(W_j) \times (ETC_{ij})} \quad (10)$$

With the heuristic information, we can update the Probability List, equation (11):

$$\text{Probability List}_{ij}^{k} = (\text{Pheromone}_{ij}^{k}) \times (\text{Heuristic}_{ij}^{k})^{\beta} \quad (11)$$

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 (12):

$$\text{Pheromone}_{ij}^{k} = (\text{Pheromone}_{ij}^{k}) \times (1-\rho) \quad (12)$$

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 (13):

$$\text{Pheromone}_{ij}^{Best\ Ant} = \text{Pheromone}_{ij}^{Best\ Ant} + \left((\rho) \times (\Delta)\right) + \frac{\varepsilon}{makespan\ (Best\ Ant)} \quad (13)$$

In global pheromone update, ρ is the elitism coefficient and Δ is calculated by equation (14):

$$\Delta = \frac{1}{makespan\ (Best\ Ant)} \quad (14)$$

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

The Proposed ACO Algorithm for Job Scheduling in Grid
Parameters Initialization (Table 1)
Create Ant Population
Choose one best ant randomly among ant population
Initialize the Pheromone and Probability Information (Formula 7)
Set the one best ant to Suffrage
While (stopping criterion not satisfied) **do**
  Construct Solution Phase
    For all Ants **do**
      Place all Ants in random chosen Resource EXCEPT the one best
      Apply Local Update for (Ant, Job, Resource) (Formula 8, 11)
    End For
    While (all Jobs not assigned) **do**
      Apply Decision Rules for (Ant, Job)
      Apply Local Update for (Ant, Job, Resource) (Formula 8, 11)
    End While
  End of Construct Solution Phase
  Global Pheromone Update Phase
    Evaporate (Formula 12)
    Find and Update best Ant of Iteration and best-so-far
    Deposit Pheromone (Formula 13)
  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.

5. EXPERIMENTAL RESULTS

The results of the evaluation of the proposed algorithm with the three algorithms of Min-Min and Max-Min [9] and Suffrage [8] 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.

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	3 (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. [21], 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 three scheduling algorithms, Suffrage, Min-Min and Max-Min. After executing these four algorithms, different amounts of makespan are obtained which are shown in Table 2.

Table 2. Comparison of four algorithms' makespans.

	The Proposed Algorithm	Suffrage	Min-Min	Max-Min
u_c_hihi	8106103.00	10249174.00	8460674.00	12385672.00
u_c_hilo	158464.09	168982.61	161805.42	204054.58
u_c_lohi	264181.47	337121.44	275837.34	392566.69
u_c_lolo	5328.91	5658.54	5441.43	6945.36
u_i_hihi	3073985.00	3306819.25	3513919.25	8018377.50
u_i_hilo	75600.41	77589.09	80755.68	151923.84
u_i_lohi	108290.52	114578.91	120517.71	251528.84
u_i_lolo	2617.01	2639.32	2785.65	5177.71
u_s_hihi	4776037.50	5121953.50	5160343.00	9208811.00
u_s_hilo	102499.90	102499.88	104375.17	172822.69
u_s_lohi	134873.02	150297.11	140284.50	282085.69
u_s_lolo	3587.83	3846.47	3806.83	6232.24

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

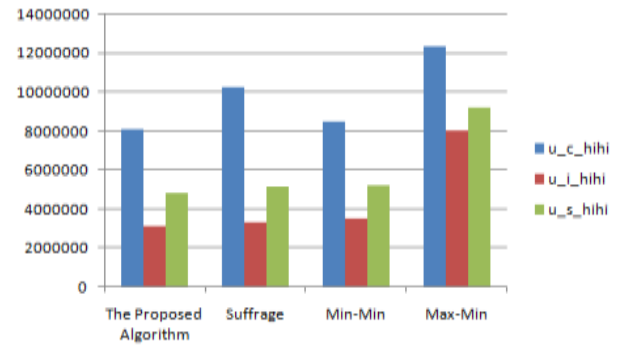

Figure 4. Algorithms' makespans based on u-*-hihi.

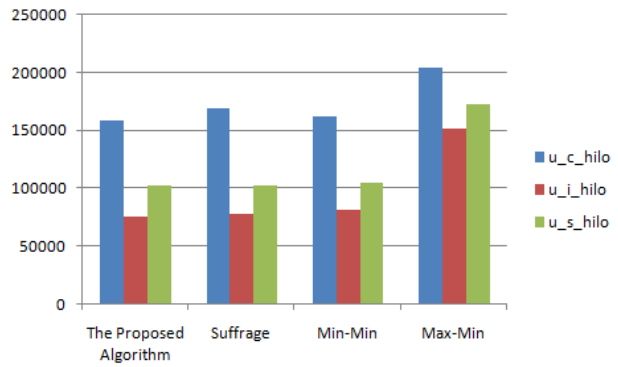

Figure 5. Algorithms' makespans based on u-*-hilo.

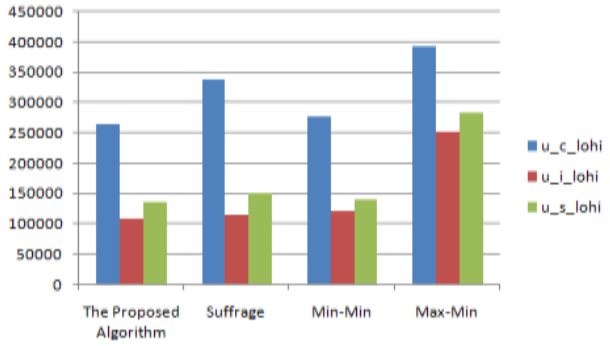

Figure 6. Algorithms' makespans based on u-*-lohi.

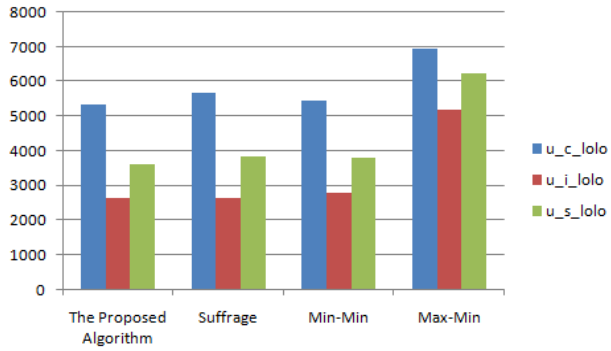

Figure 7. Algorithms' makespans based on u-*-lolo.

The three scheduling algorithms, Suffrage, Min-Min and Max-Min, are three 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 three 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 proposed algorithm is a combination of ACO and Suffrage; which means that the result of Suffrage is used in proposed ACO. One ant as an elite one is set to Suffrage' solution. 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 states that the proposed combinatorial algorithm is capable of minimizing the makespan better than other three scheduling algorithms.

7. REFERENCES

[1] Yan, K. Q., Wang, S. S., Wang, S. C., Chang, C. P., "Towards a hybrid load balancing policy in grid computing system", Expert Systems with Applications, Vol. 36, pp. 12054-12064, 2009.

[2] Yang, K., Guo, X., Galis, A., Yang, B., Liu, D., "Towards efficient resource on-demand in Grid Computing", ACM SIGOPS Operating Systems Review, Vol. 37, Issue 2, pp. 37-43, 2003.

[3] 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.

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

[5] 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.

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

[7] 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.

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

[9] 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.

[10] 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.

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

[12] 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.

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

[14] Li, Y., "A bio-inspired adaptive job scheduling mechanism on a computational grid", International Journal of Science and Network Security (IJCSNS), Vol. 6, Issue 3, pp. 1-7, 2006.

[15] Stutzle, T., "Max-Min Ant System for Quadratic Assignment Problems", Citeseer, 1997.

[16] Taillard, E. D., Gambardella, L. M., "Adaptive Memories for the Quadratic Assignment Problem", Citeseer, pp. 1-18, 1997.

[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] Sim, K. M., Sun, W. H., "Multiple ant-colony optimization for network routing", Proceedings of the First International Symposium on Cyber Worlds, pp. 277-281, 2002.

[19] Heinonen, J., Pettersson, F., "Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem", Applied Mathematics and Computation, Vol. 187, Issue 2, pp. 989-998, 2007.

[20] 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.

[21] 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.