An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment

Siriluck Lorpunmanee, Mohd Noor Sap, Abdul Hanan Abdullah, and Chai Chompoo-inwai

***Abstract*—**Grid computing is growing rapidly in the distributed heterogeneous systems for utilizing and sharing large-scale resources to solve complex scientific problems. Scheduling is the most recent topic used to achieve high performance in grid environments. It aims to find a suitable allocation of resources for each job. A typical problem which arises during this task is the decision of scheduling. It is about an effective utilization of processor to minimize tardiness time of a job, when it is being scheduled. This paper, therefore, addresses the problem by developing a general framework of grid scheduling using dynamic information and an ant colony optimization algorithm to improve the decision of scheduling. The performance of various dispatching rules such as First Come First Served (FCFS), Earliest Due Date (EDD), Earliest Release Date (ERD), and an Ant Colony Optimization (ACO) are compared. Moreover, the benefit of using an Ant Colony Optimization for performance improvement of the grid Scheduling is also discussed. It is found that the scheduling system using an Ant Colony Optimization algorithm can efficiently and effectively allocate jobs to proper resources.

***Keywords*—**Grid computing, Distributed heterogeneous system, Ant colony optimization algorithm, Grid scheduling, Dispatching rules.

I. Introduction

COMPUTATION Grid technologies [1] are a new trend and appear as a next generation of the distributed heterogeneous system. It combines physical dynamic resources and various applications. Currently, it is a great potential technology that leads to the effective utilization of the resources. Meanwhile, the complex problems such as scientific, engineering and business need to utilize the huge resources. Therefore, grid computing is considered as an ultimate solution to solve these problems. Scheduling is an important issue of current implementation of grid computing.

Siriluck Lorpunmanee is a PhD Candidate, Faculty of Information System and Computer Science, University Technology of Malaysia, Malaysia (e-mail: siriluck_lor@dusit.ac.th).

Mohd Noor Sap, PhD, is with Information System Department, Faculty of Information System and Computer Science, University Technology of Malaysia, Malaysia.

Abdul Hanan Abdullah, PhD, is with Network Security Department, Faculty of Information System and Computer Science, University Technology of Malaysia, Malaysia.

Chai Chompoo-inwai, PhD, is with Electrical Engineering Department, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Chalongkrung Rd., Ladkrabang, Bangkok 10250, Thailand (e-mail: kcchai@kmitl.ac.th).

The demand for scheduling is to achieve high performance computing. Typically, it is difficult to find an optimal resource allocation for specific job that minimizes the schedule length of jobs. The scheduling problem is defined NP-hard problem [15] and it is not trivial.

The motivation of this paper is to develop a grid scheduling algorithm that can perform efficiently and effectively in terms of minimizing total tardiness time. Not only does it improve the overall performance of the system but it also adapts to the dynamic grid system. First of all, this paper proposes an Ant Colony Optimization (ACO) algorithm to find the optimal resource allocation of each job within the dynamic grid system. Secondly, the simulation of the experiment is presented. This simulation is an extension of GridSim [14] toolkit version 4.0, which is a popular discrete-event simulator and grid scheduling algorithm. The simulator defines the different workload of resources, the arrival time of independent jobs, the length of each job, the criteria of a scheduler, etc. Finally, this paper compares the performance of various job schedulers and dispatching rules for grid environment within fully controlled conditions [16]. The experiment considers:

- First Come First Served (FCFS)
- Minimum Time Earliest Due Date (MTEDD)
- Minimum Time Earliest Release Date (MTERD)
- Ant Colony Optimization (ACO)

The rest of the paper is organized as follows. In section II, the literature review is presented. Section III discusses on the problem statement. Section IV describes the methodology and design, and briefly illustrates how to implement an Ant Colony Optimization (ACO) into the discrete-event simulator. Section V presents the experimental results and the discussions. Section VI concludes the paper. Finally, Section VII suggests the future works.

II. Literature Review

In the past few years, researchers have proposed scheduling algorithms for parallel system [2 - 6]. However, the problem of grid scheduling is still more complex than the proposed solutions. Therefore, this issue attracts the interests of the large number of researchers [7- 11].

Current systems [17] of grid resource management was surveyed and analyzed based on classification of scheduler organization, system status, scheduling and rescheduling policies. However, the characteristics and various techniques of the existing grid scheduling algorithms are still complex

particularly with extra added components.

At the present time, job scheduling on grid computing is not only aims to find an optimal resource to improve the overall system performance but also to utilize the existing resources more efficiently. Recently, many researchers have been studied several works on job scheduling on grid environment. Some of those are the popular heuristic algorithms, which have been developed, are min-min [12], the fast greedy [12], tabu search [12] and an Ant System [13].

The heuristic algorithms proposed for job scheduling in [12] and [13] rely on static environment and the expected value of execution times. H. Casanova et al. [18] and R. Baraglia et al. [19] proposed the heuristic algorithms to solve the scheduling problem based on the different static data, for example, the execution time and system load. Unfortunately, all of information such as execution time and workload cannot be determined in advance of dynamic grid environments.

In 1999, the Ant Colony Optimization (ACO) meta-heuristic was proposed by Dorigo, Di Caro and Gambardella, which has been successfully used to solve many NP-problem, such as TSP, job shop scheduling, etc. In the past few years, several researchers proposed solutions to solve grid scheduling problem by using ACO [20].

Several studies have been trying to apply ACO for solving grid scheduling problem. Z. Xu et al. [21] proposed a simple ACO within grid simulation architecture environment and used evaluation index in response time and resource average utilization. E. Lu et al. [22] and H. Yan et al. [23] also proposed an improved Ant Colony algorithm, which could improve the performance such as job finishing ratio. However, they have never used the various evaluation indices to evaluate their algorithm.

The ACO becomes very popular algorithm to apply for solving grid scheduling problem. However, most of the mentioned algorithms were not taken into consideration some evaluation indices, for example, total tardiness time that has been shown in [24].

III. Problem Statement

Grid computing is the heterogeneous environment, which is also the dynamic environment. The scheduled jobs rarely coincide between actual execution times and the expected ones in the real computing environment. Therefore, the main challenge of the job scheduling is in both the grid system since no one has the ability to fully control all jobs. The other challenge is in the available dynamic resources and the difference between the expected execution time with the actual time in algorithm.

To illustrate the impact of dynamic environment and uncertainty in the execution job time in grid scheduling system, heuristic algorithms will be used as a solving tool in the studied system. The assumptions regarding four jobs: J_1, J_2, J_3 and J_4. They are to be scheduled onto grid system of two machines, M_1, M_2. Table I shows the expected execution time and the actual execution time on the machines, where the actual execution time gives in parentheses, the min-min algorithm will allocate M_1 for J_1, M_2 for J_2 , M_2 for J_3 and M_1 for J_4 based on the expected execution time. The result is 40 time units of scheduling length, which is the optimal solution. However, the precise expected execution time of job cannot be determined in advance because of the load dynamic and many the other unknown. Hence, the previous solution is not the optimal because the scheduling will has length 50 time units. In the other words, if the assign job to the machine based on the actual execution time, it can achieve the shortest schedule length is in 30 time units.

TABLE I
THE EXPECTED AND ACTUAL EXECUTION TIME ON THE MACHINES

Job	Machine1: M1	Machine 2: M2
J_1	10(20)	20(10)
J_2	30(20)	25(30)
J_3	20(10)	15(20)
J_4	15(10)	20(5)

With the above concept there is an importance issue regarding the local search, which assigns the job to improper machine. Therefore, the scheduling considers and swaps the job from improper machine to more proper machine in the solution, as shown in Fig. 1.

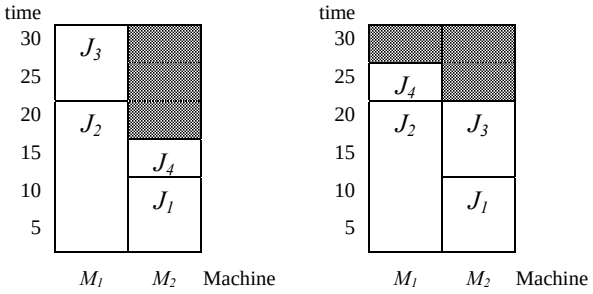

Fig. 1 Time table of problem processor (left hand side) and non-problem of processor (right hand side)

To illustrate the local search in Fig. 1, the scheduling length of the solution in the left hand side is 30 time units and the problem is in machine 2. The local search finds the optimal machine and directs the job to suitable assigned machine. Hence, this will reduce the overall of the scheduling length. The procedure can search a job on M_2 that can be swapped to the other machine. At the same time, the other jobs can be searched and swapped to another machines. In addition, a job J_3 is swapped to machine M_2 and J_4 is swapped to machine M_1 for that the scheduling length is reduced from 30 time units to 25 time unites.

Motivated by these facts, the goal of this paper is to propose an Ant Colony algorithm that can produce the optimal resources selection technique, which can find the optimal resources to process the jobs and overall performance system in term of minimizing tardiness time.

The approach is to develop scheduling algorithm within the objective to minimize total tardiness time of the jobs based on an Ant Colony Optimization (ACO). The problem is stated as follows [16]. A set of n jobs are available for processing on available set of m machines. Each job has a processing time p_j, a due date time d_j, an arrival time a_j and a release time r_j which is incurred when job j immediately follows job i. It is assumed that all the processing times, due dates, release time and arrival times are non-negative integers. Job preemptions are not allowed. Let C_{ij} be completion time of the operation of job j on machine i. Thus, the completion time of the job jth in machine ith is given as

$$C_{i,j} = a_j + r_j + p_{i,j}. \quad (1)$$

The tardiness of the jth job in machine i is given as

$$T_{i,j} = \max(C_{i,j} - d_j, 0). \quad (2)$$

The objective is to minimize the maximal total tardiness time of all the jobs within machine of grid environment.

$$\sum_{i=1}^{m}\left(\sum_{j=1}^{n} T_{i,j}\right). \quad (3)$$

IV. Methodology and Design

A. Grid Simulation Architecture

The grid environment is a complex heterogeneous system consisting of many organization domains in which no one has full control of all available resources and applications (jobs). For this reason, the simulation of these systems is complicated. The simulation presented in this paper is based on GridSim [14] and [16]. It also extends internal entities to more complex requirements. Additionally, the ACO algorithm for predicting the job is allocated to the optimal resources. The layer of simulation is shown in Fig. 2.

The users can register any time in the grid environment, after that they are able to submit jobs to the grid environment. The submitted jobs are received at the Receiver side. The receiver side is to automatically submit the selected jobs to the Scheduling Module. Where, the receiver has the function to maintain the job work load, user name, etc.

At the Scheduling Module, all of the information in the grid system such as CPU speed, CPU load, number of CPUs in the resources, etc. has been collected from the Grid Information Service (GIS) and they are used to calculate the optimal resources for processing the job. This paper designs an Ant Colony Optimization (ACO), which is explained in the following section.

The purpose for a Job Dispatcher is to deliver jobs to a selected resource and maintain these jobs in the grid queue. The function of the Grid Information Service (GIS) is to maintain and update the status of the grid resources.

B. An Event Diagram of Grid Simulation

GridSim toolkit [14] provides the basic functionality to create common entities such as computation resources, users, simple jobs, etc. and the simple implementation of grid environment. Different amount Processing Elements (PEs) are

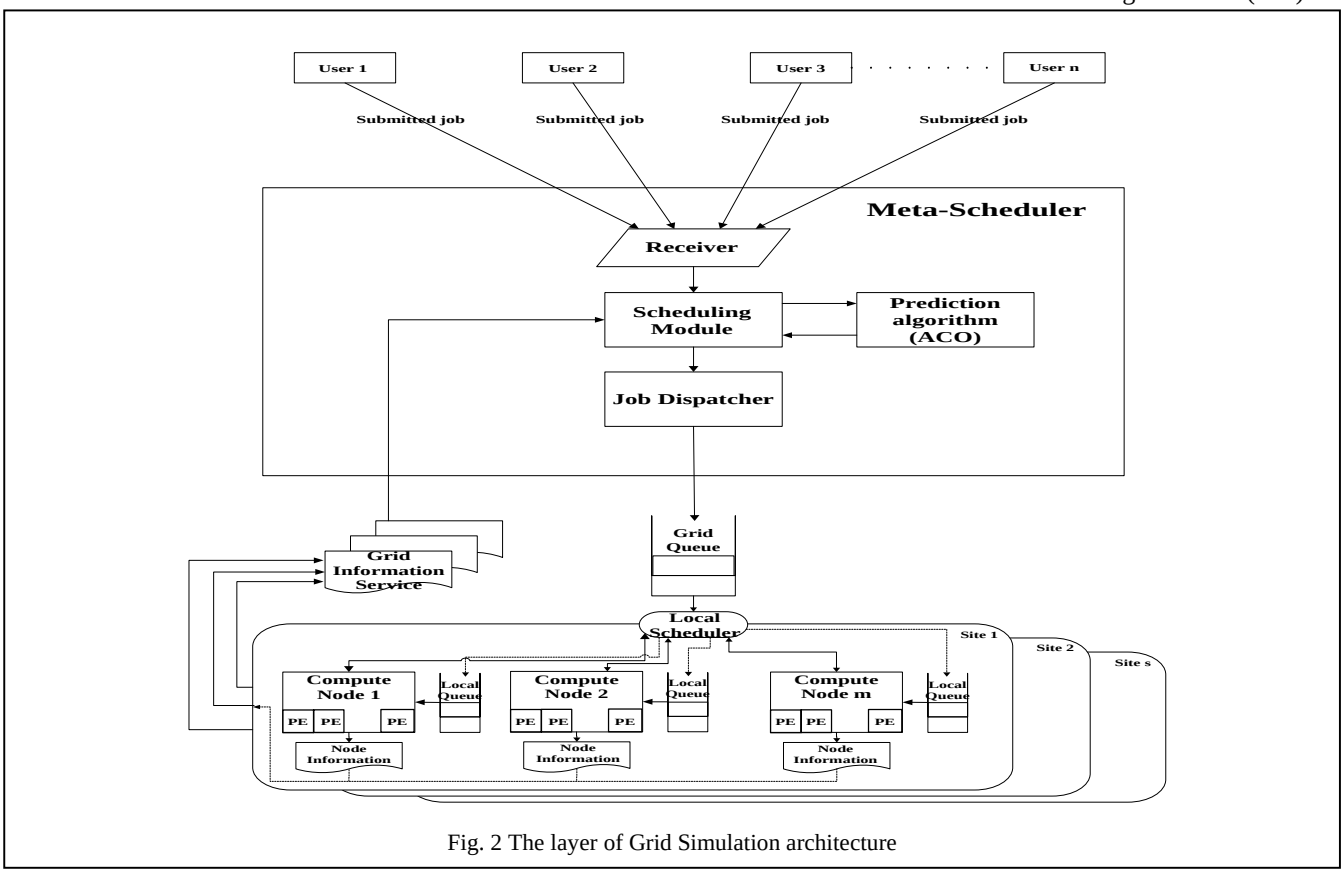

Fig. 2 The layer of Grid Simulation architecture

created during the simulation. These Processing Elements (PEs) are also different in the speed of processing. Likewise, one or more machines can be put together to create grid resources.

The event diagram of grid simulation is shown in Fig. 3. When the simulator starts, Grid Resource Entities sends an event to GIS entities for registration. Hence, GIS entities return a list of registered resources and their details to Meta-scheduler. Therefore, Grid Client Entities submit the jobs and its details such as arrival time, releasing time, due date time, length of job and the request of resources configuration, properties, etc, to Meta-scheduler. The Meta-scheduler responds with dynamic information such as resources workload, available resources, capability, and the other properties.

The Meta-Scheduler is an important part of the job scheduler. Jobs are selected or scheduled to grid resources by an Ant Colony Optimization (ACO). If grid resources are not available, the job is kept at the Grid Job Pool. When the job processing is finished, Grid Resource Entity is released and Meta-Scheduler sends the event to GIS for updating the grid resources status.

Once the processing of all the jobs is finished the statistical results are generated automatically.

C. The Proposed ACO Algorithm

Generally, when the jobs are submitted to the grid system at different times, they have different time steps, therefore they consumes different resources. Therefore, the processing is performed within the different execution times. In order to deal with those jobs, they are submitted to the dynamic load of the system, which the system takes effect to the job execution time. This problem is more difficult than static load of the system. This paper proposes ACO algorithm to solve this problem by considering the requirements of each job, which is independent from other jobs and where only one of job processes in any unit time slot (space sharing).

Ant Colony Optimization (ACO) is one of the meta-heuristics. It can be applied not only to solve discrete optimization problems but also to solve both static and dynamic combinational optimization problems.

ACO is inspired by a colony of artificial ants cooperate in foraging behavior, which is described in more details in [25]. This behavior enables ants to find the shortest paths between food sources and their nest. In fact, they deposit a chemical pheromone tail on the ground after they walk from the nest to food sources and vice versa. Hence, they choose the way with higher probability paths, which are marked by stronger pheromone concentrations. This behavior is the basis for a cooperative interaction. It gives the distinction described in Fig. 4.

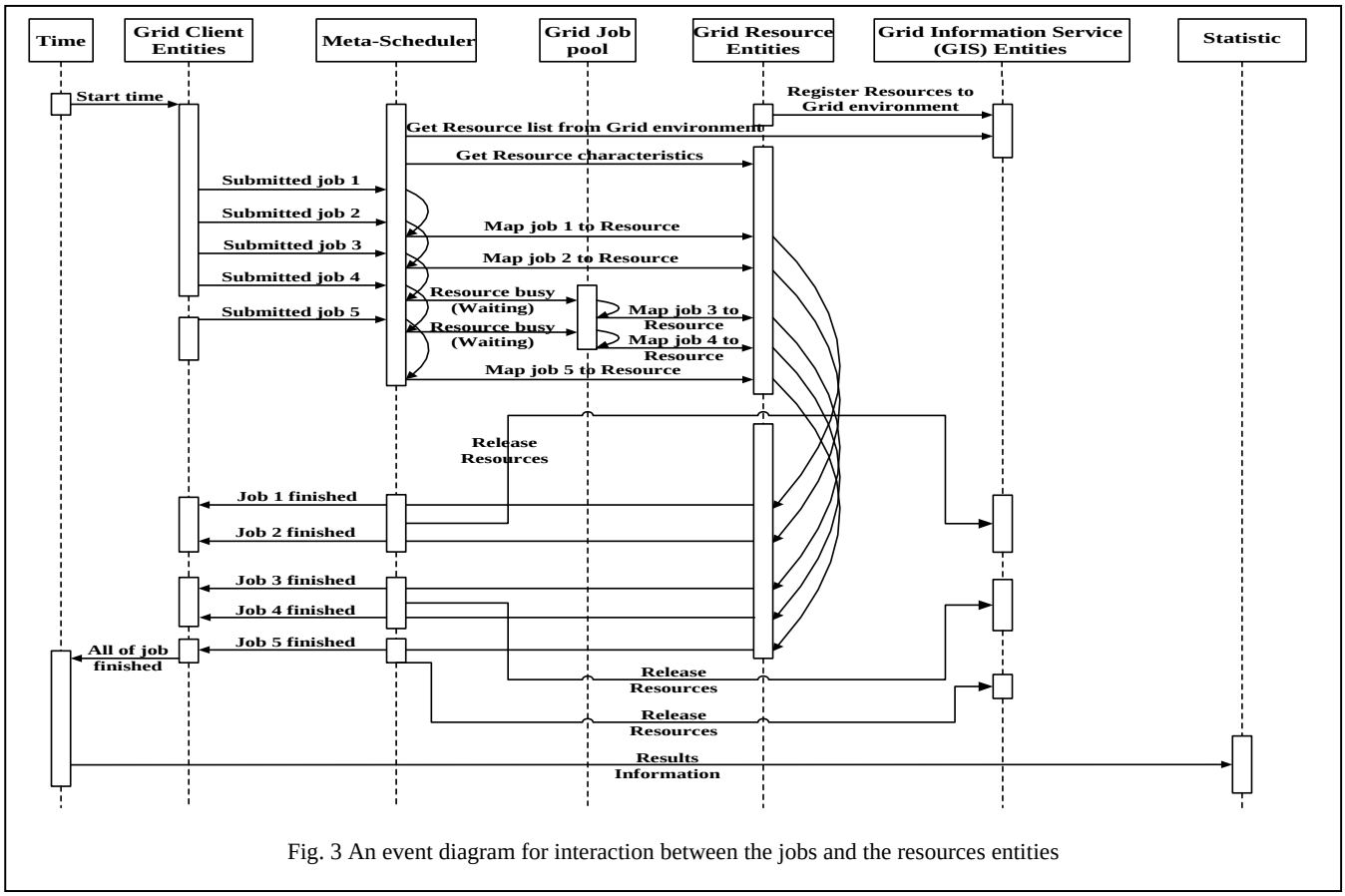

Fig. 3 An event diagram for interaction between the jobs and the resources entities

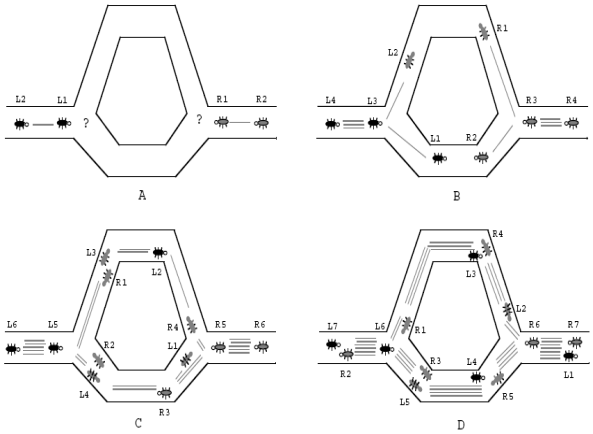

Fig. 4 T0 illustrate of how real ants can lead to identify the shortest path around an obstacle. Figure from [26]

In this paper, the general adaptive ACO algorithm is basically the ACS [26], is applied to solve the scheduling problem in grid environment. Four main key terms of ACO algorithm are listed and defined below:

The expected time complete (ETC) is defined as the amount of time that the job is processed at a machine, and it has no load to the assigned job. Where the complete time (CT) is defined as the wall clock time, which machine completes for each job. Then, $CT_{ij} = a_j + r_j + ETC_{ij}$, where a_j is arrival time of a job jth, r_j is a release time of a job jth and i is a machine ith.

The main important of ACO algorithm is to utilize the graph representation. Therefore, the design of the graph is to identify the problem to connect the arcs correspondent for each job on each machine.

Let M be a set of machines $\{m_1, m_2, m_3,.., m_m\}$ and let J be a set of jobs $\{j_1, j_2, j_3, ..., j_n\}$, and $n > m$. Therefore, the graph $G = (M, \{CT_{ij}\}_{mxn})$. The problem is to find the optimal resources for the jobs. It can minimize the total tardiness time.

Step 1: Pheromone initialization. Let the initial pheromone trail

$$\tau_0 = \frac{1}{m.\left(\sum_{i=1}^{m} T_i \Big|_{Actual} + \sum_{i=1}^{m}\left(\sum_{j=1}^{n} T_{i,j}\right)_{Expected}\right)} \quad (4)$$

where $\sum_{i=1}^{m} T_i \Big|_{Actual} + \sum_{i=1}^{m}\left(\sum_{j=1}^{n} T_{i,j}\right)_{Expected}$ is the total current tardiness time by applying the dispatching rule of Minimum Completion time (MCT).

$\sum_{i=1}^{m} T_i \Big|_{Actual}$ is the actual tardiness time of jobs, which are already completed on machine i, $\sum_{i=1}^{m}\left(\sum_{j=1}^{n} T_{i,j}\right)_{Expected}$ is the expected tardiness time of jobs, which are scheduled on machine i.

Step 2: The State Transition Rule. A set of artificial ants is initially created. Each ant starts with unscheduled job in machine and then they build a job tour in machines until a feasible solution is constructed. For the next machine m to be scheduled from current machine i, the ant applies the rule as shown in Eq. (5)

$$m = \begin{cases} \arg\max_{u \in U}[\tau(i,u)].[\eta(i,u)]^{\beta} & if\ \ q \le q_0, \\ S & otherwise, \end{cases} \quad (5)$$

where q is a random number uniformly distributed in [0, 1], and q_0 is a parameter $(0 \le q_0 \le 1)$, $\tau\ (i, u)$ is the associated pheromone trail with the assignment of job j to be scheduled at machine i to machine u, U is the machines with no schedule jobs yet. The parameter q_0 determines the relative importance of exploitation versus exploration. In the other hand, with probability q_0 the ant makes the best possible move as marked by pheromone trails and the heuristic information. Hence, if $q \le q_0$ the unscheduled job j with maximum value is scheduled at machine m, that is, a job j move from machine i to schedule at machine m. S is a random variable selected according to the probability distribution is implemented by Eq. (6)

$$p(i,m) = \begin{cases} \dfrac{[\tau(i,m)][\eta(i,m)]^{\beta}}{\sum_{u \in U}[\tau(i,u)][\eta(i,u)]^{\beta}} & if\ \ m \in U \\ 0 & otherwise \end{cases} \quad (6)$$

where $\eta(i,m)$ is the heuristic desirability of assignment job j on machine i directly moves to m, is inversely proportional to the completion time of job that has been assigned on machine. Then, the formula follows as, $\eta_j(i,m) = \dfrac{1}{a_j + r_j + p_{m,j}}$, a is the arrival time that a job arrives to the system, r is the release time that a job use to start processing and p is the processing time of the job on the machine.

Step 3: The Local Update Rule. While ants build a solution of the scheduling, they visit edges and change their pheromone level which is used immediately to locally update the rule. The local update rule reduces the convergence because ants choose new machine based on high pheromone level therefore, this machine becomes less desirable for the following ants, if the pheromone trail is reduced. Hence, this is achieved as shown by Eq. (7)

$$\tau(i,j) = (1-\rho).\tau(i,j) + \rho.\tau_0 \quad (7)$$

where $\rho\ (0 < \rho \le 1)$, is the parameter.

Step 4: The Global Update Rule. The global update rule is performed after each ant has completed their tour (a feasible solution) and only one ant that is the best solution found so

far, is allowed to deposit pheromone to the path after each iteration. Therefore, job *j* is assigned from machine *i* to machine *m* in the global best solution after each iteration, the formula as shown by Eq. (8)

$$\tau(i,m) = (1-\alpha).\tau(i,m) + \alpha.\Delta\tau(i,m) \qquad (8)$$

where

$$\Delta\tau(i,m) = \begin{cases} (\min(\sum_{all\ M} T_{gb}))^{-1} & \text{, if } (i,m) \in \text{global-best-solution} \\ 0 & \text{, otherwise} \end{cases}$$

$\alpha(0< \alpha \leq 1)$ is the parameter and represents the pheromone evaporation.

```
Procedure ACO
begin
    Initialize parameters, the pheromone trails
    while (stopping criterion not satisfied) do
        Each ant position starts at starting machine
        while (stopping when every ant has build a
              solution) do
            for each ant do
                Chose next machine by applying
                the state transition rate
            end for
        end while
        Update the pheromone
    end while
end
```

Fig. 5 The ACO implemenation

V. Experiment and Results

The experimental simulation is designed and implemented based on grid simulation architecture described in section III. The experiment is performed on Intel Pentium 4 2.6 GHz machine with 640 MB RAM. Different computational workloads are tested on the experiment as is shown in Table II, and the different resource is shown in Table III.

TABLE II
Jobs Workload Attributes

Parameters	Values	Notations
Total number of job	3,000	
Length of a job	1,000 – 5,000	Million Instructions (MI)
Arrival time distribution	1 – 20,000	Uniform distribution
Number of CPUs requirement	1-4 CPUs	Processing Elements (PEs)

TABLE III
The Grid Resources Attributed

Parameters	Values	Notations
Total number of resource	10 - 20	machines
Speed of CPU	50 - 200	Million instructions per second (MIPS)
bandwidth	10	Mbit
Number of CPUs each machine	1-4 CPUs	Processing Elements (PEs)

The different data sets are generated into 20 types and the result is averaged from it. The ACO algorithm parameters values as shown in the following Table IV:

TABLE IV
The Ant Parameters Values

α	β	P	*ants*	q_0
0.1	2.0	0.1	30	0.8

In Fig. 6, shows the results of comparison between FCFS, MTEDD, MTERD and ACO algorithms. The results show that the tardiness time based on the number of available machines in the grid system. Although the MTEDD algorithm is good when is compared with the basic algorithms but it still has not good enough when compares with an Ant Colony Optimization (ACO) algorithm. As the results stated ACO able to perform the optimal scheduling job. Besides, the ACO accounts for less than 17%of the total tardiness time in the average when it is compared with the other scheduler algorithms.

On the other hand as shown in Fig. 7, an ACO algorithm performs much slower than the other scheduler algorithms. The reason for that ACO calculates several times for searching the optimal resource that is assigned to process the job. Hence, the calculation time of scheduling consumes more resources. Therefore, the results show FCFS algorithm is much faster than the others.

VI. Conclusion

This paper investigates the job scheduling algorithm in grid environments as an optimization problem. The proposal is a cost model for modeling schedule lengths in grid computing in term of minimizing tardiness time. The algorithm is developed based on an Ant Colony Optimization to find a proper resource allocation on each processing job. The algorithm can find an optimal processor for each machine to allocate to a job that minimizes the tardiness time of a job when the job is scheduled in the system.

In the study, the algorithm is designed and compared to different grid environments. Moreover, comparison of the experiments of minimizing the maximal lateness is evaluated and that means the demand of the grid users tries to have the latency of each job as small as possible. The results show that an Ant Colony Optimization algorithm is the best average-case of the tardiness time. It is also observed that the different performance and number of machines do have significant impact on schedule lengths.

VII. Future Work

The future work will focus on extension of the grid environment addition to the job migrations, the different characteristic of grid domain, network topology and also the job workflow. Moreover, the plan to implement and evaluate the different cost measures such as make-span time, Grid Efficiency and job error ratio.

Acknowledgment

We would like to thank Suan Dusit Rajabhat University for supporting us.

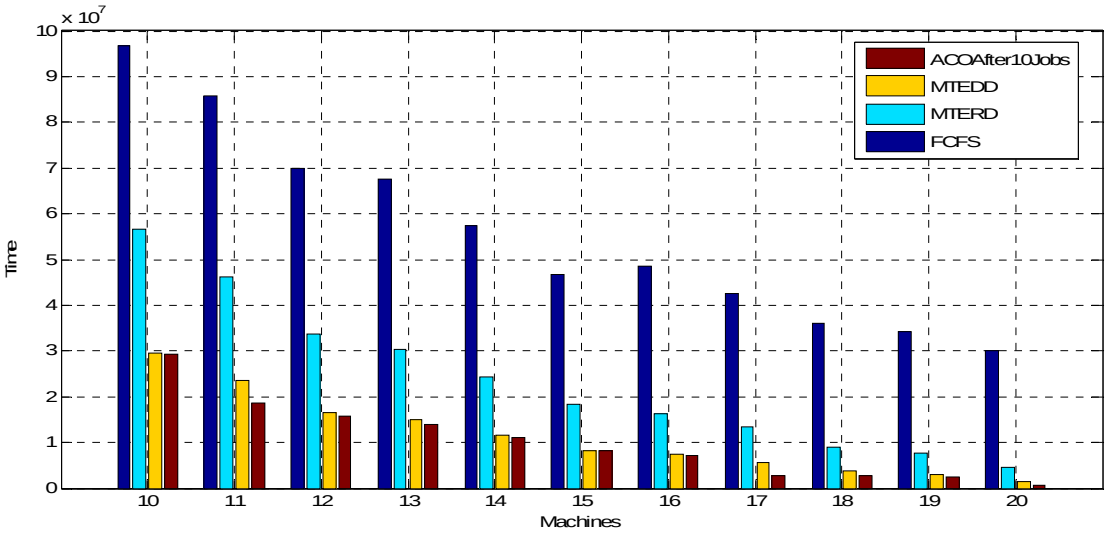

Fig. 6 Total tardiness time of dynamic job scheduling in grid computing

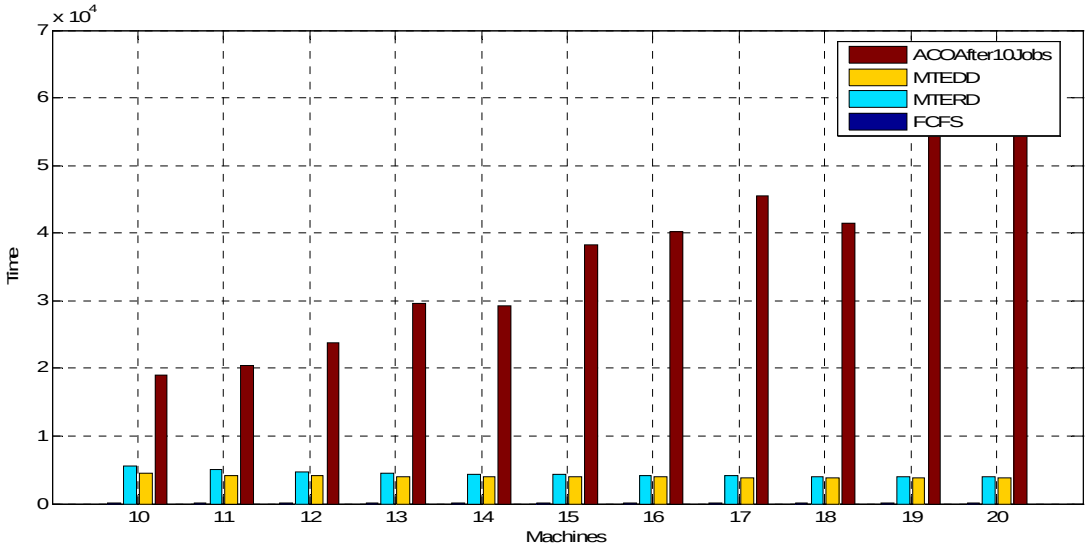

Fig. 7 Total scheduling time of dynamic job scheduling in grid computing

REFERENCES

[1] I. Foster and C. Kesselman, editors, "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann Publishers, 1999.

[2] D.G. Feitelson. "Packing schemes for gang scheduling", In *2nd Workshop on Job Scheduling Strategies for Parallel Processing*, volume LNCS 1162, pages 89–100, 1996.

[3] D.G. Feitelson, L. Rudolph, U. Schwiegelshohn, K.C. Sevcik, and P. Wong. "Theory and practice in parallel job scheduling", In *3rd Workshop on Job Scheduling Strategies for Parallel Processing*, volume LNCS 1291, pages 1–34, 1997.

[4] J. Krallmann, U. Schwiegelshohn, and R. Yahyapour. "On the design and evaluation of job scheduling algorithms", In *5th Workshop on Job Scheduling Strategies for Parallel Processing*, volume LNCS 1659, pages 17–42, 1999.

[5] J. M. van den Akker, J. A. Hoogeveen, and J. W. van Kempen. "Parallel machine scheduling through column generation: Minimax objective functions", In Y. Azar and T. Erlebach, editors, *European Symposium on Algorithms*, volume 2996 of *Lecture Notes in Computer Science*, pages 648-659. Springer, 2004.

[6] R.D. Nelson, D.F. Towsley, and A.N. Tantawi. "Performance analysis of parallel processing systems", *IEEE Transactions on Software Engineering*, 14(4):532–540, 1988.

[7] V. Hamscher, U. Schwiegelshohn, A. Streit, R.Yahyapour, "Evaluation of job-scheduling strategies for grid computing", *Proceedings of First IEEE/ACM International Workshop on Grid Computing, Lecture Notes in Computer Science*, vol. 1971, Springer, Berlin, 2000, pp. 191–202.

[8] V. Subramani, R. Kettimuthu, S. Srinivasan, P. Sadayappan, "Distributed job scheduling on computational grids using multiple simultaneous requests", *Proceedings of the 11th International Symposium on High Performance Distributed Computing*, 2002, pp. 359–366.

[9] H. Shan, L, Oliker and R. Biswas, "Job Superscheduler Architecture and Performance in Computational Grid Environments", *Proceedings of the ACM/IEEE SC2003 Conference (SC03),* 2003.

[10] C. Ernemann , V. Hamscher and R. Yahyapour, "Benefits of Global Grid Computing for Job Scheduling", *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04),* 2004.

[11] K. Li, "Job scheduling and processor allocation for grid computing on Metacomputers ", *Journal of Parallel and Distributed Computing, Elsevier,* 2005

[12] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen and R. F. Freund (2001), "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(6): Pages 810-837.

[13] G. Ritchie and J. Levine, "A fast, effective local search for scheduling independent jobs in heterogeneous computing environments".

[14] R. Buyya and M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing", *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, 14:1175-1220, 2002.

[15] D. Fernandez-Baca (1989), "Allocating Modules to Processors in a Distributed System", *IEEE Transactions on Software Engineering*. Vol. 15(11): Pages 1427-1436.

[16] D. Klus´a˘cek, L. Matyska, and H. Rudov´a, "Grid scheduling simulation environment", Submitted to MISTA, *3rd Multidisciplinary International Scheduling Conference: Theory and Applications*, France, 2007.

[17] K. Krauter, R. Buyya and M. Maheswaran, "A taxonomy and survey of Grid resource management systems for distributed computing", Software Pract. Exp. 2 (2002) 135–164.

[18] H. Casanova, A. Legrand, D. Zagorodnov and F. Berman, "Heuristics for scheduling parameter sweep applications in Grid environments", *in: Heterogeneous ComputingWorkshop 2000*, IEEE Computer Society Press, 2000, pp. 349–363.

[19] R. Baraglia, R. Ferrini, and P. Ritrovato, "A static mapping heuristics to map parallel applications to heterogeneous computing systems", Research articles. *Concurrency and Computation: Practice and Experience*, 17(13):1579–1605, 2005.

[20] P. Fibich, L. Matyska, and H. Rudov´a, "Model of Grid Scheduling Problem", *In Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing*, Papers from the AAAI-05 workshop. Technical Report WS-05-03, AAAI Press, 2005.

[21] Z. Xu, X. Hou and J. Sun, "Ant Algorithm-Based Task Scheduling in Grid Computing", *Electrical and Computer Engineering, IEEE CCECE 2003, Canadian Conference*, 2003.

[22] E. Lu, Z. Xu and J. Sun, "An Extendable Grid Simulation Environment Based on GridSim", *Second International Workshop, GCC 2003*, volume LNCS 3032, pages 205–208, 2004.

[23] H. Yan, X. Shen, X. Li and M. Wu, "An Improved Ant Algorithm for Job Scheduling in Grid Computing", *In Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, 18-21 August 2005.

[24] M. Pinedo (1995). "Scheduling –Theory, Algorithms, and Systems", Prentice Hall.

[25] M. Dorigo and T. Stutzle, "Ant Colony Optimization", The MIT Press.

[26] M. Dorigo and LM. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *In the IEEE Transactions on Evolutionary Computation*, Vol.1, No.1, pages: 53-66, 1997.

Siriluck Lorpunmanee was born at Chiang-rai (Thailand) on August 30, 1971. Current, he is Ph.D. candidate at faculty of Computer Science and Information Systems (FSKSM), Universiti Teknologi Malaysia, Johor, Malaysia.

His research areas are on Grid Computing system, Swarm Intelligence algorithms and simulation model.

Chai Chompoo-inwai (S'01-M'07) received his B.S. and M.S. degrees in Electrical Engineering from King Mongkut's Institute Technology of Ladkrabang (KMITL), Bangkok, Thailand in 1995 and 1997, respectively. He received Ph.D. in Electrical Engineering from University of Texas at Arlington in 2005.

He joined KMITL as a faculty member since 1995. His research areas are on power system analysis, transient and dynamic stability, transmission congestion management in electricity deregulation, renewable power planning and Illumination engineering.