JOB ONLINE SCHEDULING WITHIN DYNAMIC GRID ENVIRONMENT

Siriluck Lorpunmanee[1], Mohd Noor Md Sap[2], Abdul Hanan Abdullah[3]

[1]Faculty of Science and Technology, Suan Dusit Rajabhat University,
295 Rajasima Road, Dusit, Bangkok, Thailand, Tel: (662)-2445225, Fax: (662) 6687136
[1]siriluck_lor@dusit.ac.th

[2,3]Faculty of Computer Science and Information Systems, University Technology of Malaysia,
81310 Skudai, Johor, Malaysia, Tel: (607)-5532070, Fax: (607) 5565044
[2]mohdnoor@fsksm.utm.my, [3]hanan@fsksm.utm.my

Abstract: This paper proposes the idea of adaptive job scheduling algorithm by using hybrid Ant Colony Optimization (ACO) and Tabu algorithms. The idea behind the scheduling algorithm is evaluation of completion time of jobs in a service Grid. The algorithm comprises of two main techniques; first of all, Grid Information Service (GIS) collects information from each grid node, ACO evaluates complete time of jobs in possible grid nodes and then assigns job to appropriate grid node. ACO is used to minimize the average completion time of jobs through optimal job allocation on each node as well. While, Tabu algorithm is used to adjust performance of grid system because online jobs are submitted to grid system from time to time. This paper shows that 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.

Keywords: Ant Colony Optimization algorithm, Tabu algorithm, Grid Information Service, online job, Adaptive scheduling algorithm.

1. 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. In particular, the complex problems such as scientific, engineering and business need to utilize the huge resources and the performance potential of computation grid. Therefore, the performance of grid system is the key to success. Scheduling is the main point behind these successes.

Effective job scheduling in grid requires to model grid resources and computation requests of jobs, determine the current workload of each grid nodes, and evaluate the execution time of job before allocating to appropriate resource as well. The goals are to achieve the best performance and load balancing across the different domain system. However, when applying those methods to grid environment, the results often happen to be poor performance within heterogeneous grid resources. Whereas, grid environment is open and dynamic, the jobs often arrive at grid in various load and types. For instance, the scheduling needs time slot to schedule a job within various workloads. Hence, the adaptive grid scheduling is the desired algorithm to handle different jobs workload and also various load of grid resources. One of the most important methods of a Grid Resources Management System (GRMS) is currently capable to allocate jobs to grid resources.

The realistic grid system, it is usually be impossible for users and grid system administrators to manually find and specify all the needed grid resources during the arrival of jobs in fact. In the dynamic nature of grid, the various load of grid resources availability can constantly change at the moment. Therefore, the requirement of an automatic method to allocate jobs to grid resources is a critical. This method is generally provided by scheduler. Typically, it is difficult to achieve this point because the scheduling problem is defined as NP-hard problem [2] and it is not trivial.

Traditional and heuristic scheduling algorithms [1, 3-5] are often proposed in heterogeneous computing environment. These algorithms focus on explicit constraints and static information of system load to schedule jobs. The dynamic grid environment, the system workload of each grid nodes cannot be determined in advance. Therefore, grid scheduling desires an adaptive scheduling algorithm.

The motivation of this paper is to develop a grid scheduling algorithm that can perform efficiently and effectively in terms of grid efficiency, make-span time and job tardiness. 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 environment. Secondly, Tabu search algorithm is used to adjust performance of grid system because online jobs are often submitted to grid system. Third, the simulation of the experiment is presented. This simulation is an extension of GridSim [6] 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 size of each job, the criteria of a scheduler, etc. Finally, this paper discusses about this idea for dynamic grid environment within fully controlled conditions [7].

The rest of the paper is organized as follows. In section 2, the literature review is presented. Section 3 describes the Grid simulation, job scheduling design, and briefly illustrates how to implement an Ant Colony Optimization (ACO) with Tabu search into the discrete-event simulator. Section 4 illustrates the well performing of the adaptive job scheduling. Finally, Section 5 concludes the paper.

2. LITERATURE REVIEW

In the past few years, researchers have proposed scheduling algorithms for parallel system [8-12]. However, the problem of grid scheduling is still more complex than the proposed solutions. Therefore, this issue attracts the interest of the large number of researchers [3, 13-16].

Current system [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 components.

At the present time, job scheduling on grid computing aims not only 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. Those studies are the popular heuristic algorithms, which have been developed, are min-min [5], the fast greedy [5], tabu search [5], Genetic algorithm [18] and an Ant System [4].

The heuristic algorithms proposed for job scheduling in [5] and [4] rely on static environment and the expected value of execution times. Whereas, [18] proposed Genetic algorithm based on static jobs and static load of each node, while the realistic grid environment, in which the system load of grid nodes always change during the deployment of jobs in fact.

H. Casanova et al. [19] and R. Baraglia et al. [20] 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 [21] by using ACO.

Several studies have been trying to apply ACO for solving grid scheduling problem. Z. Xu et al. [22] proposed a simple ACO within grid simulation architecture environment and used evaluation index in response time and resource average utilization. E. Lu et al. [23] and H. Yan et al. [24] 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, grid efficiency, makespan time, average waiting time and total tardiness time that has been shown in [25].

3. GRID SIMULATOR AND JOB SCHEDULING DESIGN

3.1 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 simulator presented in this paper is based on GridSim [6] and [7]. It also extends internal entities to more complex requirements. Additionally, the architecture is comprised of distributed grid nodes, local schedulers with local queues, and grid Meta-scheduler with grid queue and job dispatcher. Moreover, the main idea behind adaptive scheduling algorithm with rescheduling algorithm is proposed within this simulator.

The layer of all simulation is depicted in Fig. 1. 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. Similarly, the registration of distributed system is required at Grid Information Service (GIS) in order to be grid resources.

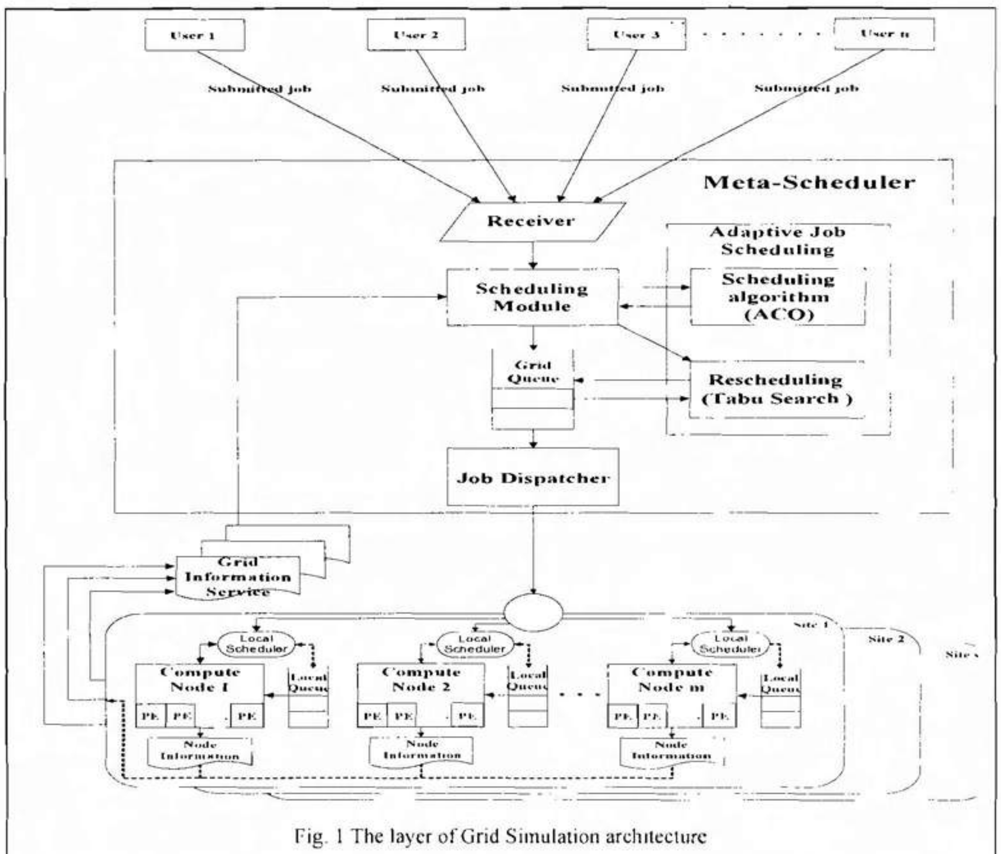


Fig. 1 The layer of Grid Simulation architecture

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 at the Grid Information Service (GIS). This process is similar to GRIS (Grid Resource Information Server) registering with GIIS (Grid Index Information Server) in Globus toolkit [26]. Their information is then used to calculate the optimal resources for processing the job. Therefore, the function of the Grid Information Service (GIS) is to maintain and update the status of the grid resources. The purpose of a Job Dispatcher is to deliver jobs to a selected resource which jobs are in grid queue.

The assumption is a close affinity between GIS and nodes information. Moreover, Scheduling module gathers the local schedule information via GIS for scheduling. In fact, the job stays in the local queue before beginning execution on the local system and the job is computed based on the local scheduling policy. The purpose of a job dispatcher is to deliver jobs to a selected resource. Finally, all activities mentioned above are the support evidence of this simulator.

This paper designs an Ant Colony Optimization (ACO) and Tabu search algorithm which are explained in the following section.

3.2 An event diagram of Grid Simulation

GridSim toolkit [6] 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 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. 2. When the simulator starts, Grid Resource Entities send 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 their details such as arrival time, releasing time, due date time, size 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 other propertiesThe 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. Whenever 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.

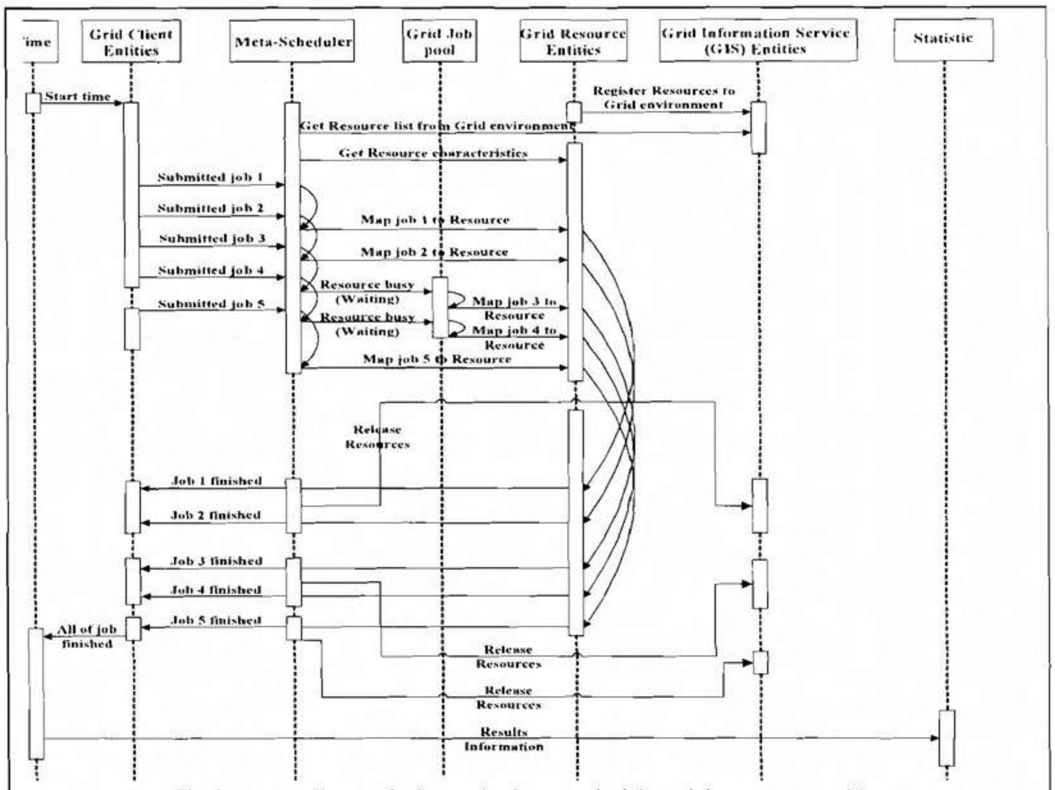


Fig. 2 an event diagram for interaction between the jobs and the resources entities

3.3 Job Scheduling Model

The model including job workloads model and grid resources model, the main idea behind model is to associate metadata with the description of jobs and grid resources.

a) Job Workloads Model

The workloads model views jobs [27] in form of ‹JobID, Description, HostID› where the model is a set of arrival job to be executed in grid system and Description contains metadata about the job such as arrival time, release time, due date, size of job, requirement of PEs, host architecture, owner, etc. HostID is a host to execute the jobs.

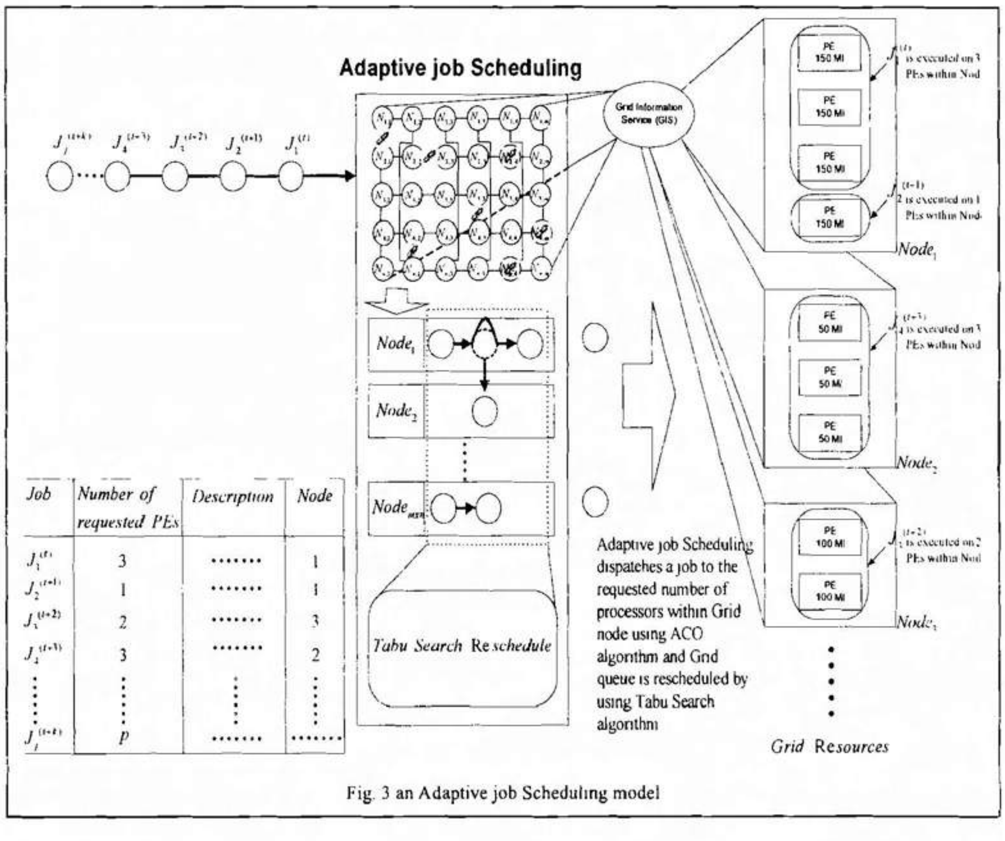
b) Grid Resources Model

Grid resources model contains characteristics and performances of grid resources. Therefore, the model ‹HostID, Description› is a set of grid nodes and hosts information e.g.

location of each hosts, operating system, PEs, etc.

3.4 Adaptive Job Scheduling Model and Algorithms

The model is shown in Fig. 3, in which the main idea behind the success is the adaptive job scheduling. Typically, not only most jobs to an open grid system are independent but also arrive to grid system within different time. They also consume the different resources and time to process. In the real grid system, job scheduling function is to assign the jobs to grid resources along with their time constraints based on the objective of optimizing a certain performance function [25]. The timing constraints are both based on jobs and include the dynamic grid resources situation. This model can solve the dynamic jobs arrival time and loads of grid resources. However, local queue of each grid node is simulated in a space-sharing fashion.


Adaptive job Scheduling
$J_l^{(t+k)}$
$J_4^{(t+3)}$
$J_3^{(t+2)}$
$J_2^{(t+1)}$
$J_1^{(t)}$
Grid Information Service (GIS)
$Node_1$
$Node_2$
$Node_{max}$
Tabu Search Reschedule
Adaptive job Scheduling dispatches a job to the requested number of processors within Grid node using ACO algorithm and Grid queue is rescheduled by using Tabu Search algorithm
PE 150 MI
PE 50 MI
PE 100 MI
$J_1^{(t)}$ is executed on 3 PEs within Nod
$J_2^{(t+1)}$ is executed on 1 PEs within Nod
$J_4^{(t+3)}$ is executed on 3 PEs within Nod
$J_3^{(t+2)}$ is executed on 2 PEs within Nod
$Node_1$
$Node_2$
$Node_3$
Grid Resources

Job	Number of requested PEs	Description	Node
$J_1^{(t)}$	3	....	1
$J_2^{(t+1)}$	1	....	1
$J_3^{(t+2)}$	2	....	3
$J_4^{(t+3)}$	3	....	2
⋮	⋮	⋮	⋮
$J_l^{(t+k)}$	p	....	....

Fig. 3 an Adaptive job Scheduling model

On account of the fact that the loads of each grid node are dynamic and many other unknowns within the dynamic nature of grids, the execution time of a job can not be exactly determined in advance. Therefore, job scheduling; is very difficult to deal with the dynamic load of the grid node that affects the execution time of the submitted job.

Motivated by these facts, this model is designed to solve job scheduling problems. The model comprises two main algorithms; first of all, ACO is proposed to perform the scheduling that assigns the jobs to appropriate grid resources, secondly, reschedule technique is performed by using Tabu search algorithm.

To simplify grid scheduling problem, the independent jobs are submitted and processed at grid node in any time, in which there no communication between itself. In this case, an Expected Time Complete (ETC) matrix is performed, which cell *(i, j)* is the ETC of job *i* is operated onto machine *j*. A set of *n* jobs are submitted for processing on available set of *m* machines. Each job is characterized with an arrival time a_i representing the arrival job on grid system. A release time r_i representing the earliest time when the job can start its execution. A due date time d_i representing the desire job completion time, and a size of job l_i, requirement of *PEs*, etc. Therefore, the processing time $p_{i,j}$ of job *i* onto machine *j* is l_i/PEs of machine *j* which depends on the CPU speed of machine *j*. Moreover, they are assumed that all arrival times, due dates, release time and processing time are non-negative integers. Job preemptions are not allowed as well. Let C_{ij} be completion time of the operation of job *i* on machine *j* and the job requirement of PEs is less than or equal to PEs of machine $(i_{PEs} <= j_{PEs})$. Thus, the completion time of the job *i*th in machine *j*th is given as:

$$C_{i,j} = a_i + r_i + p_{i,j}\,. \tag{1}$$

The makespan time is defined as $max(C_1, C_2, \ldots, C_n)$ that means the last complete time of job to leave the grid system.

$$Makespan\ time = (EndTime_{last\_job} - SubmitTime_{first\_job}) \tag{2}$$

Whereas, average waiting time and grid efficiency formulas are given as

$$Average\ wait\ time = \frac{1}{N}\sum_{i\in Jobs}(StartTime_i - SubmitTime_i) \tag{3}$$

$$Grid\ Efficiency = \frac{\sum_{j \in Jobs} (EndTime_j - StartTime_j) \times Mflops_j}{(EndTime_{last\_job} - SubmitTime_{first\_job}) \times \sum_{m \in grid\_sites} Mflops_m} \tag{4}$$

In this model, therefore, all of these can be handled and satisfy stakeholder.

a) ACO Algorithm

At the first of the model, ACO algorithm is proposed to schedule jobs onto different machines and different speeds as well. 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 $\{m_1, m_2, m_3, \ldots, m_m\}$ and let J be a set of jobs $\{j_1, j_2, j_3, \ldots, j_n\}$, 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 makespan time, average waiting time and maximize grid efficiency as well.

Step 1: Pheromone initialization. Let the initial pheromone trail follows as

$$\tau_0 = \frac{1}{m.\left(Min\_MakeSpan\_Time_{i,j}\right)^{(Current)}_{Expected}} \tag{5}$$

where $\left(Min\_MakeSpan\_Time_{i,j}\right)^{(Current)}_{Expected}$ is the current expected makespan time of online job by applying the dispatching rule of Minimum Completion time (MCT) that calculate online job into each queue of grid nodes if there is a free PE available. Therefore, the minimization of makespan time is applied to be minimal makespan time at that moment. It is shown as Fig. 4

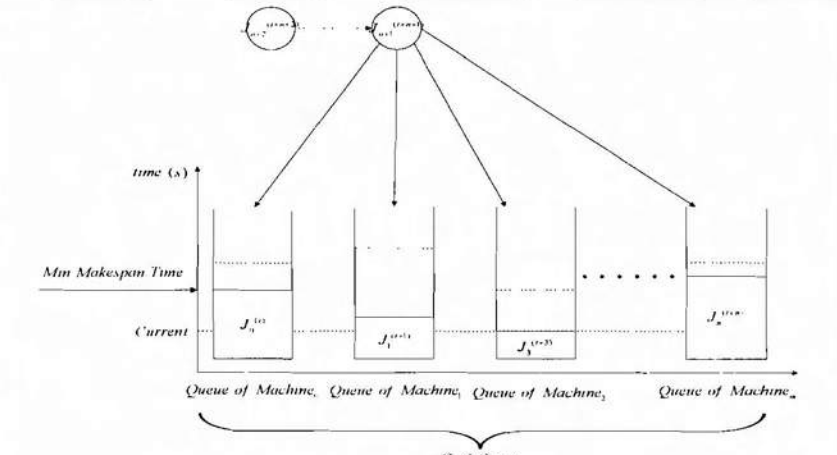


Fig. 4 Expected makespan time of online job into each queue grid nodes

***Step** 2:* The State Transition Rule. A set of parallel artificial ants is initially created. Each ant starts with unscheduled job in machine and then it builds 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. (6).

$$m = \begin{cases} \arg\max\limits_{u \in U} [\tau(i,u)].[\eta(i,u)]^{\beta} & \textit{if } q \leq q_0, \\ S & \textit{otherwise}, \end{cases} \tag{6}$$

where q is a random number uniformly distributed in [0, 1], and q_0 is a parameter $(0 \leq q_0 \leq 1)$, $\tau(i, u)$ 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 words, with probability q_0 the ant makes the best possible move as marked by pheromone trails and the heuristic information. Hence, if $q \leq q_0$ 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. (7)

$$p(i,m) = \begin{cases} \dfrac{[\tau(i,m)][\eta(i,m)]^{\beta}}{\sum\limits_{u \in U} [\tau(i,u)][\eta(i,u)]^{\beta}} & \textit{if } m \in U \\ 0 & \textit{otherwise} \end{cases} \tag{7}$$

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. However, arrival time and release time are perhaps the same time because when a job arrives, grid node starts immediately its execution.

***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. (8)

$$\tau(i,j) = (1-\rho).\tau(i,j) + \rho.\tau_0 \tag{8}$$

where ρ *(0<ρ≤ 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. (9)

$$\tau(i,m) = (1-\alpha).\tau(i,m) + \alpha.\Delta\tau(i,m) \tag{9}$$

Where

$$\Delta\tau(i,m) = \begin{cases} (\min_{All\ M}(Makespan\_Time_{gb}))^{-} & \text{if}(i,m) \in \text{global-best-solution} \\ 0 & \text{otherwise} \end{cases}$$

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

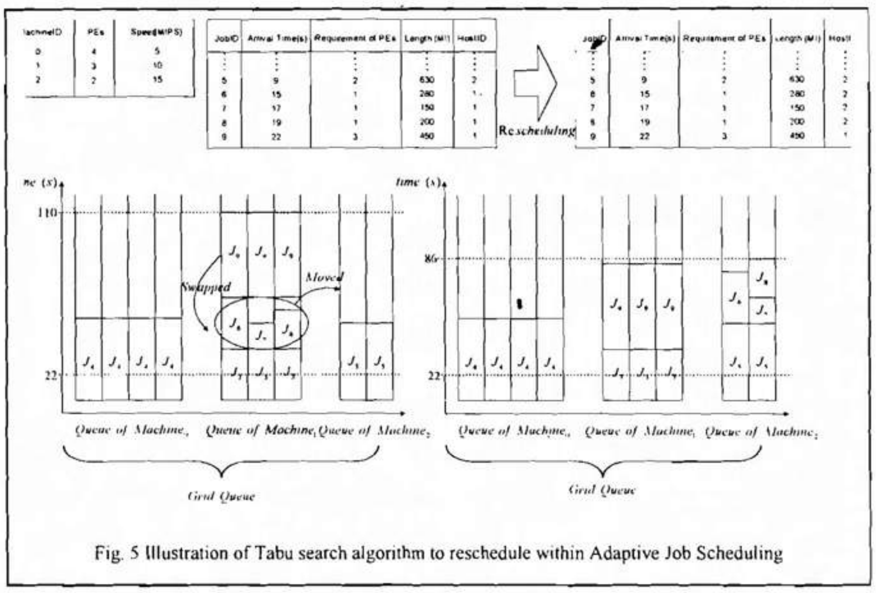
Algorithm 1 The implementation of ACO algorithm

```
Procedure ACO
begin
   Initialize parameters, generate initial graph and the pheromone trails Eq. (5)
   while (termination criterion not satisfied or a maximum number of algorithm
iterations has been
            reached) do
      Each ant position starts at starting machine
      while (stopping when every ant has built a solution) do
         for each ant do
            Chose next machine by applying the state transition rule Eq. (6)
            Apply step-by-step pheromone update Eq. (8)
         end for
      end while
      Update best solution, if the best makespan time this teration is better than the
globally
         makespan time then set this is the globally best makespan time Update the
pheromone Eq.(9)
   end while
end
```

Unfortunately, the realistic grid system, it may usually be possible grid resources are unavailable free of PEs. Therefore, address, the job is kept at the Grid Job Pool (Grid queue). According to this issue, Tabu search technique is proposed and its detail is depicted in the following next section.

b) Tabu Search Algorithm

In this model, addresses Tabu search algorithm to solve the problem. The idea of this model is depicted in Fig. 5.



lachneID	PEs	Speed(MIPS)
0	4	5
1	3	10
2	2	15

JobID	Arrival Time(s)	Requirement of PEs	Length (MI)	HostID
⋮	⋮	⋮	⋮	⋮
5	9	2	630	2
6	15	1	280	1
7	17	1	150	1
8	19	1	200	1
9	22	3	450	1

JobID	Arrival Time(s)	Requirement of PEs	Length (MI)	HostI
⋮	⋮	⋮	⋮	⋮
5	9	2	630	2
6	15	1	280	2
7	17	1	150	2
8	19	1	200	2
9	22	3	450	1

Fig. 5 Illustration of Tabu search algorithm to reschedule within Adaptive Job Scheduling

For example, an arrived job is submitted at 22th second and it is then allocated to appropriate grid node using ACO. Unfortunately, at that time, grid resources are unavailable free of PEs and some arrived jobs are still in grid queue. Therefore, Tabu search algorithm starts immediately to reschedule within grid queue.

However, the due date time of each job is considered. In particular, during that time, before the swap operator exchanges three jobs to other grid nodes. It calculates a due date time of each job in the proper grid node. The swap operator relocates job based on due date time of each job in sequence. Finally, the makespan time of grid system is better.

Tabu search algorithm [28] is a metaheuristic and it is one of the most popular techniques to solve scheduling problems and the other types of combinatorial problems. The main idea behind a Tabu search is to have a large number of iterations. Every iteration of the local heuristic search procedure to explore the solution space beyond local optimality. At each iteration of the local search procedure, it only allows an operation called move or swap to define the neighborhood of any solution when the objective function is improved and the operations are not repeating a previous solution that already improved in the previous iterations.

Consequently, to prevent repeating the indefinite execution of the same sequence of the operations and direct the search to regions are not explored. Tabu search uses tabu list for checking the operations. This technique is very useful for forbidding or tabu as well as escaping local optimal during the performing of a searching global optimal. Accordingly, tabu search algorithm is applied and integrated within this model to solve dynamic grid scheduling problem. The following is an algorithm description.

The adaptive job scheduling model in particular, the schedule is consist of several resource schedules. Each resource schedule is a queue of jobs that they planed to be executed within grid node. For this model, the algorithm is designed to minimize the makespan time of the grid system. It is noted that the makespan time of each resource schedules follow as

$$\overset{|schedule[1]|}{\max}\ (CT_{1,1}, CT_{2,1} \ldots, CT_{n,1})$$

$$\overset{|schedule[2]|}{\max}\ (CT_{1,2}, CT_{2,2} \ldots, CT_{m,2})$$

$$\vdots$$

$$\overset{|schedule[j]|}{\max}\ (CT_{1,j}, CT_{2,j} \ldots, CT_{i,j})$$

where $CT_{i,j}$ approximates the current expected complete time of job *i*th that plan to be executed on machine *j*th. It is defined as the wall-clock time at which machine *j*th completes job *i*th.

The value of expected $CT_{i,j}$ can be approximated because the processing time $p_{i,j}$ is known as well as other important parameters such as arrival time, release time, due date time, requirement of PEs, machine, speed of PEs, etc. Consequently, this information and the current schedule are possible to calculate expected makespan time for each job present in the grid system at this moment. The formula as shown by Eq. (10)

$$Makespan\ time = \max(\overset{|schedule[1]|}{makespan}, \overset{|schedule[2]|}{makespan}, \ldots, \overset{|schedule[j]|}{makespan}) \qquad (10)$$

Algorithm 2 The implementation of Tabu search algorithm

```
Procedure Tabu Search Algorithm
begin
  Initialize Tabu list with maximum size of iterations
  $schedule = [res\_schedule_1,\ldots, res\_schedule_l]$;
  $tabu\_resource\_schedule = \emptyset$; $tabu\_jobs\_list = \emptyset$;
  $res\_schedule_{\max}^{(Current)} = null$;
  while ($l <$ iterations) do
   $l = l + 1$;
   $previous\_makespan$ = Eq. (10)
   Calculation maximum of the current expected makespan time of each resource
schedule
    if $(\exists schedule_{[i]}[i \in res\_schedule])$ and $(schedule \notin tabu\_resource\_schedule)$ and
       ($schedule$ is maximal makespan time) then
       $res\_schedule_{\max}^{(Current)} = schedule[i] \quad ; i \in \{1,\ldots,j\}$
       Calculation minimize of the processing time of a job within maximal
makespan time resource
       schedule
       $(res\_schedule_{\max}^{(Current)})$
       if $(\exists job[job \in res\_schedule_{\max}^{(Current)}])$ and
         $(job \notin tabu\_jobs\_list)$ and (the processing time of a job is minimal) then
         Moving job to the other appropriate resource schedule that the current
makespan time to be
         minimized.
         MoveJob($job$, $tabu\_jobs\_list$, $res\_schedule_{\max}^{(Current)}$,
         $tabu\_resource\_schedule$, $previous\_makespan$);
       else
          $tabu\_resource\_schedule = tabu\_resource\_schedule \cup res\_schedule_{\max}^{(Current)}$;
       end if
    end if
  end while
end
```

Algorithm 3 The implementation of MoveJob of Tabu search algorithm

```
Procedure MoveJob(job, tabu_jobs_list, $res\_schedule_{\max}^{(Current)}$, tabu_resource_schedule,
                  previous_makespan)
begin
   $res\_schedule_{\min}^{(Current)} = null$;
   Calculation minimum of the current expected makespan time of each resource
schedule.
   if $(\exists schedule_{[i,j]}[i \in res\_schedule])$ and $(schedule \notin res\_schedule_{\max}^{(Current)})$ and
     (schedule is minimal makespan time) and (requirement of PEs of job <= PEs
of resource) then
      $res\_schedule_{\min}^{(Current)} = schedule[i] \quad ; i \in \{1, \ldots, j\}$
      if $res\_schedule_{\min}^{(Current)} = null$ then
         tabu_jobs_list = tabu_jobs_list U job;
     else
         remove job from $res\_schedule_{\max}^{(Current)}$ to $res\_schedule_{\min}^{(Current)}$ by using EDF, in which
         remove job on position $l$ such that $(due\ date\ -1 \leq l \leq due\ date\ +1)$
         if (Eq. (10) >= previous_makespan) then
            remove job from $res\_schedule_{\min}^{(Current)}$ back to
            $res\_schedule_{\max}^{(Current)}$ at the same previous position
         end if
      end if
   end if
end
```

4. ILLUSTRATION OF ADAPTIVE JOB SCHEDULING IN GRID SCHEDULING PROBLEM

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 the grid system since no one has the ability to fully control all jobs. Another challenge is availability of 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, adaptive job scheduling algorithm is proposed. Fig. 6 illustrates the operation of this model, in well performing conditions.

Consider the example in Fig. 6, the assumptions regarding ten jobs: J_0, J_1, J_2,..,J_9 within the different job characteristics such as job arrival time, requirement of PEs, size of each job.

They are scheduled onto grid resource of three machines: M_0, M_1 and M_2 within different PEs speed and amount of PEs. This assumption is shown in Fig. 6 as well.

First of all, the simulation begins, when J_0 arrives on grid system at time 0, optimal FCFS is performed in scheduling module, and then it is mapped to machine2 at PE0, PE1 and it is delivered at time 3.33. At time 1, J_1 arrives and it is mapped to the PE0 of machine1 because machine2 is processing J_0. The delivery time of J_1 is 11. At time 2, J_2 arrives and it is mapped to PE0, PE1, PE2 of machine0. It is delivered at 33. At time 3, J_3 arrives, but there is now no free PE available on the grid node.

However, optimal FCFS still finds the optimal grid node and assigns to machine1 but it is put into the queue of grid job pool. The completion time of J_3 is predicted as 31. At time 4, J_4 arrives and it is then put into the queue of grid job pool with suitable grid node assignment because grid nodes are not free. In this case, the completion time of J_4 is predicted as 82. At time 9, J_5 arrives and it is then put into queue. However, it is already assigned to perform at machine2. the completion time of J5 is predicted as 51. Unfortunately, at time 82, it signifies the makespan time at the moment, in which depicted in Fig. 6 (a).

On the other hand, in this case, adaptive job scheduling is proposed and ACO is then applied. Fig. 6 (b), shows the solution, let start at J_0 when it arrives grid system and then ACO immediately performs, parallel artificial ants start at each grid node to find appropriate one. At machine2, it has high performance when compare with the other. J_0 is then executed there and it is delivered at 3.33. J_1 arrives grid system at time 1 and it is then allocated to machine2 by ACO. Since the processing time of J_1 in machine2 is predicted as 6.67 that it is faster than processing in the others and it is predicted the delivery at 10. Nevertheless, it is kept in the queue of grid job pool. At time 2, J_2 arrives and allocated to machine1 by ACO because the amount of PEs of machine1 is 3 and faster than machine0 but an amount of PEs of machine2 is not sufficient. J_3, J_4, J_5 arrive grid system at any time and allocated by ACO. Finally, the makespan time is predicted as 53 which is shown in Fig. 6 (b).

In fact, the arrived jobs are submitted to the grid system at any time and the loads of grid node are unknown in advance. Therefore, during scheduling perform, some jobs are kept in the queue of grid job pool. Whenever the jobs are kept in there, Tabu search algorithm is immediately performed. Fig. 5 and Fig. 6 (c-d) is shown the detail.

5. CONCLUSION

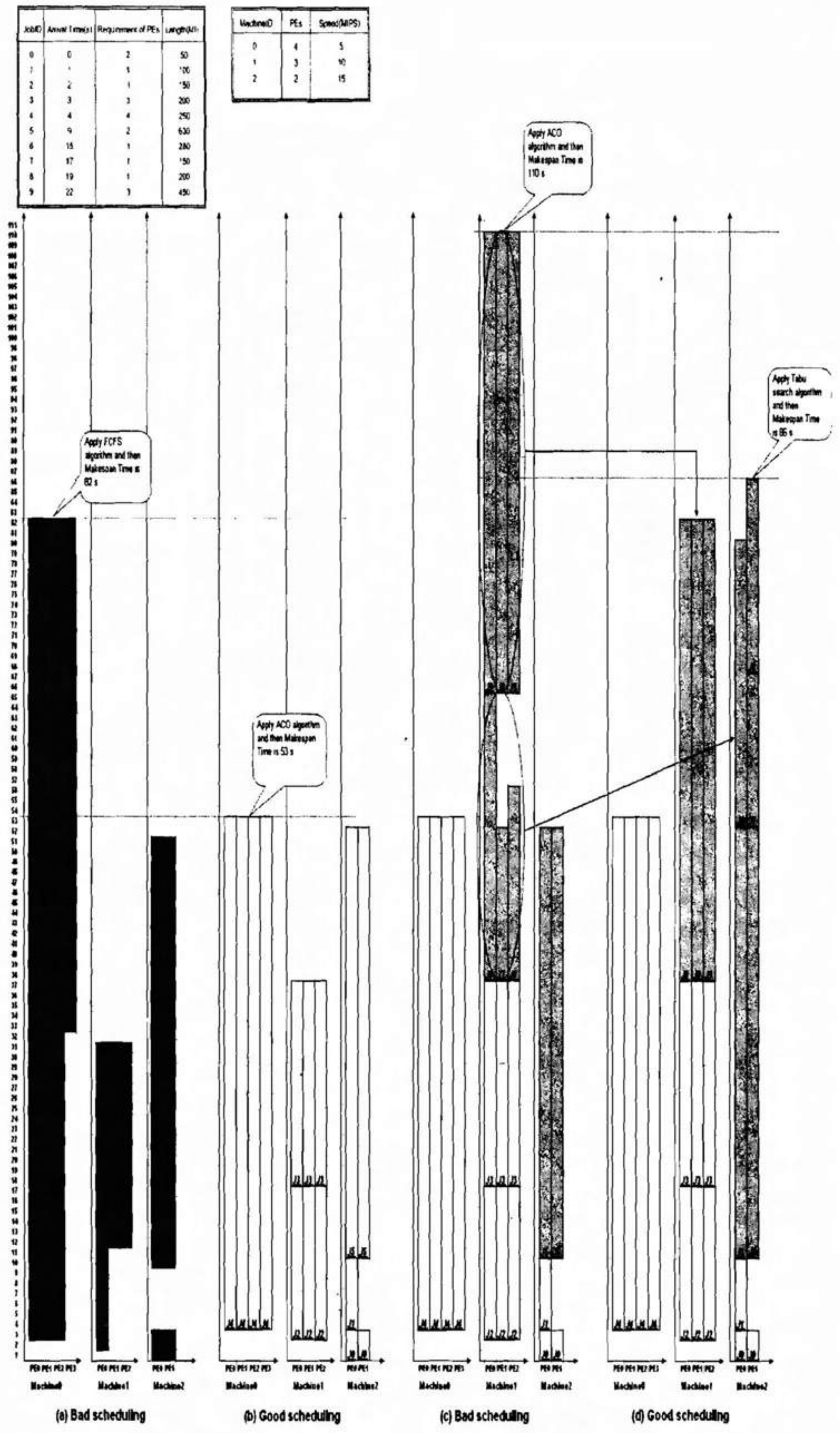
This paper investigates the job scheduling algorithm in grid environments as an optimization problem. The proposal is a cost model for modeling schedule lengths in grid computing in term of minimizing tardiness time. The algorithm is developed based on an Ant Colony Optimization to find a proper resource 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 within dynamic CPUs load, in which the natural grid usually occur different load of CPUs from time to time. Moreover, this paper makes effort to design probably adaptive algorithm based on multiple objective of different stakeholder, for example makespan time, average waiting time and grid efficiency.

The future work will show the comparison with tradition algorithm FCFS for example and another heuristics such as MCT, Climbing Search, Simulated Annealing and Tabu search. Moreover, to show the evaluation of scheduling algorithms, in which this algorithm will probably get good enough results.

ACKNOWLEDGMENT

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



JobID	Arrival Time(s)	Requirement of PEs	Length(MI)
0	0	2	50
1	1	1	100
2	2	1	150
3	3	3	200
4	4	4	250
5	9	2	630
6	15	1	280
7	17	1	150
8	19	1	200
9	22	3	450

MachineID	PEs	Speed(MIPS)
0	4	5
1	3	10
2	2	15

Fig. 6 Example of best scheduling of Adaptive scheduling algorithm by using ACO and Tabu search

REFERENCES

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure* 1st ed.: Morgan Kaufmann, 1999.

[2] D. Fernandez-Baca, "Allocating Modules to Processors in a Distributed System," *IEEE Transactions on Software Engineering,* vol. 15, p. 1427, 1989.

[3] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour, "Evaluation of Job-Scheduling Strategies for Grid Computing," in *Grid Computing — GRID 2000*, 2000, pp. 191-202.

[4] G. Ritchie and J. Levine, "A Fast, Effective Local Search for Scheduling Independent Jobs in Heterogeneous Computing Environments," in *PLANSIG 2003: Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group*, Glasgow, 2003.

[5] D. B. Tracy, S. Howard Jay, B. Noah, L. B. Lasislau, l, ni, M. Muthucumara, I. R. Albert, P. R. James, D. T. Mitchell, Y. Bin, H. Debra, and F. F. Richard, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. Parallel Distrib. Comput.,* vol. 61, pp. 810-837, 2001.

[6] R. Buyya and M. Manzur, "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),* pp. 1175-1220, 2002.

[7] D. Klus'a˘cek, L. Matyska, and H. Rudov'a, "Alea - Grid Scheduling Simulation Environment," *Lecture Notes in Computer Science,* 2007.

[8] G. F. Dror, R. Larry, S. Uwe, C. S. Kenneth, and W. Parkson, "Theory and Practice in Parallel Job Scheduling," in *Proceedings of the Job Scheduling Strategies for Parallel Processing*: Springer-Verlag, 1997.

[9] D. Feitelson, "Packing schemes for gang scheduling," *Job Scheduling Strategies for Parallel Processing,* vol. Volume 1162/1996, pp. 89-110, 1996.

[10] K. Jochen, S. Uwe, and Y. Ramin, "On the Design and Evaluation of Job Scheduling Algorithms," in *Proceedings of the Job Scheduling Strategies for Parallel Processing*: Springer-Verlag, 1999.

[11] N. Randolph, T. Don, and N. T. Asser, "Performance Analysis of Parallel Processing Systems," *IEEE Trans. Softw. Eng.,* vol. 14, pp. 532-540, 1988.

[12] J. van den Akker, J. Hoogeveen, and J. van Kempen, "Parallel Machine Scheduling Through Column Generation: Minimax Objective Functions," in *Algorithms – ESA 2006*, 2006, pp. 648-659.

[13] E. Carsten, H. Volker, and Y. Ramin, "Benefits of Global Grid Computıng for Job Scheduling," in *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*: IEEE Computer Society, 2004.

[14] S. Hongzhang, O. Leonid, and B. Rupak, "Job Superscheduler Architecture and Performance in Computational Grid Environments," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*: IEEE Computer Society, 2003.

[15] L. Keqin, "Job scheduling and processor allocation for grid computing on metacomputers," *J. Parallel Distrib. Comput.*, vol. 65, pp. 1406-1418, 2005.

[16] S. Vijay, K. Rajkumar, S. Srividya, and P. Sadayappan, "Distributed Job Scheduling on Computational Grids Using Multiple Simultaneous Requests," in *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*: IEEE Computer Society, 2002.

[17] "A taxonomy and survey of grid resource management systems for distributed computing," *Softw. Pract. Exper.*, vol. 32, pp. 135-164, 2002.

[18] S. Lorpunmanee, M. N. M. Sap, A. H. Abdullah, and S. Surat, "Meta-scheduler in Grid environment with multiple objectives by using genetic algorithm," *WSEAS TRANSACTIONS on COMPUTERS* vol. 5, pp. 484 - 491, 2006.

[19] C. Henri, Z. Dmitrii, B. Francine, and L. Arnaud, "Heuristics for Scheduling Parameter Sweep Applications in Grid Environments," in *Proceedings of the 9th Heterogeneous Computing Workshop*: IEEE Computer Society, 2000.

[20] B. Ranieri, F. Renato, and R. Pierluigi, "A static mapping heuristics to map parallel applications to heterogeneous computing systems: Research Articles," *Concurr. Comput. : Pract. Exper.*, vol. 17, pp. 1579-1605, 2005.

[21] P. Fibich, L. Matyska, and H. Rudov'a, "Model of Grid Scheduling Problem," in *In Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing* 2005.

[22] X. Zhihong, H. Xiangdan, and S. Jizhou, "Ant algorithm-based task scheduling in grid computing," in *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, 2003, pp. 1107-1110 vol.2.

[23] E. Lu, Z. Xu, and J. Sun, "An Extendable Grid Simulation Environment Based on GridSim," in *Grid and Cooperative Computing*, 2004, pp. 205-208

[24] H. Yan,.X. Shen, X. Li, and M. Wu, "AN IMPROVED ANT ALGORITHM FOR JOB SCHEDULING IN GRID COMPUTING," in *In Proceedings of the Fourth International Conference. on Machine Learning and Cybernetics*, Guangzhou, 2005

[25] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, second ed.: Prentice Hall, 1995.

[26] "http://www.globus.org/toolkit/."

[27] A. Pugliese, D. Talia, and R. Yahyapour, "Modeling and Supporting Grid Scheduling," *Journal of Grid Computing*.

[28] F. W. Glover and M. Laguna, *Tabu Search* 1ed.: Kluwer Academic, 1997.