Unified Economic Deadline Scheduling Algorithm for Computational Grid

Sugree Phatanapherom and Putchong Uthayopas

HPCNC, Faculty of Engineering, Kasetsart University,
Bangkok, 10900 Thailand

sugree@hpcnc.cpe.ku.ac.th and pu@ku.ac.th

Abstract

Efficient and dynamic resources scheduling algorithms are needed to harness the hidden power of a Grid System. In this work, a few economic scheduling algorithms have been proposed. First, Grid system is modeled using a vector space approach. Then, a cost function for the operation has been defined. Using the model and cost function defined, a few new economic scheduling algorithms have been proposed as an improved extension to many conventional Grid scheduling algorithms. In addition, the algorithms proposed are also address the problem of deadline scheduling as well. The concept proposed has then been evaluated using a simulation. The results show that the scheduling algorithms proposed perform better than the traditional algorithms. The results and approach presented here can be applied to improve Grid level scheduler system.

Keyword: Grid, Scheduling Algorithm, Heuristic, Deadline, Economic, Scheduler

I. Introduction and Related Work

Grid technology [1, 2] is a technology that allows the use of geographically distributed computing systems belonging to multiple-organizations as a single system. Grid system is usually built using Grid middleware such as Globus [3] to form a secure, distributed infrastructure. However, Globus provides only a basic mechanism such as security services, remote execution services, and data transfer services. Although these mechanisms can be used as a basis for the construction of the Grid, the task of discovering and allocating a set of optimal resources is left to higher level software. Therefore, it is important to have a smart resources scheduler that selects the optimal set of resources for the users.

There are a large number of literature that address the problem of mapping sets of tasks onto sets of processors in a view of minimizing overall execution time. Many of these works address the case where tasks are independent. For example, [4] proposed both on-line mode and batch mode scheduling heuristics: MinMin, MaxMin and Sufferage. [5] analyzed the same problem using stochastic model and proposed load balancing scheme using second order moments as well as first order moments. This scheme is verified to improve both in static and dynamic scheduling. In [6], XSufferage scheduling heuristic in batch mode has been proposed comparing to the original one. XSufferage is a modification of Sufferage to reuse already existing input file in some resources assuming that each computing system (cluster) has it own shared file system (NFS, IBP, GFS, AFS, PVFS or etc.). However, this heuristic does not take care of the economic model and only improve performance of this kind of application that input files could be reused by other tasks. A scheduling algorithm called “Placeholder scheduling” is proposed in [7]. The purpose is efficiently schedule general tasks to multiple remote queues relying on user-level meta-queue design. This methodology does not run on top of Grid but actually runs on a set of scripts for querying and submitting on PBS or SGE. Placeholder software is a script that fetches tasks from the pool and run it using existing tools (e.g. ssh and rcp). The problem of economic scheduling is left unaddressed as well in this work. For all these scheduling algorithms, simulation has been a major way to study the performance of each propose scheme. Effect of accuracy of the simulation is studied in [8] using simulation based on trace from NCSA mainly for parallel tasks. The result shows that increasing accuracy of submitted execution time helps improving performance of the system by fitting the task in backfill policy.

One of the major issues in the design of an efficient scheduler is the organization of Grid scheduling structure. Infrastructure classification is studied extensively in [9]. Subramani et al. [10] has classified the Grid scheduling structure into 3 categories [10] namely, Centralized, Hierarchical, and Distributed structure. First, the centralized scheduling structure consists of only one scheduler and one queue for all distributed resources. Most of the existing schedulers such as Nimrod/G [11], GrADS [12], and AppLeS [13]), belong to this category. Second, hierarchical scheduling structure leaves the final decision of selecting resources to local job manager of each organization. Finally, in a distributed scheduling structure, there are many interacted Grid-level scheduler that communicates with in order to make a scheduling decision. Subramani et al. [10] also proposed 2 scheduling heuristics that can improve the performance of a distributed infrastructure by limiting amount of knowledge of current status. However, the work is only focused on the problem to schedule parallel task on multiple computing resources. Economic scheduling for Grid system is also addressed in [14] for the same problem but the target of the work changes to the minimization of average wait time and average response time.

In this paper, three new scheduling heuristics are proposed to satisfy 2 goals; minimizing turnaround time, and minimizing rental cost of scheduling independent tasks. This is very important for the practical use of grid infrastructure since each organization on Grid may needs to charge for resource usage. A way to model this problem in order to make it easy and straight forward to solve are needed.

II. Modeling the Grid

Assuming that Grid consists of n nodes and each node has m resources. Let $\vec{N}$ be the node vector that represents the amount of each resource in a node (Eq. 1) and $\vec{G}$ be the Grid vector that represents global Grid system as vector of nodes (Eq.2). For each processor, an execution rate is used to represent its speed. An interconnection network is modeled using point-to-point bandwidth.

$$\vec{N}_i = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_m \end{bmatrix} \tag{1}$$

$$\vec{G} = \begin{bmatrix} \vec{N}_1 & \vec{N}_2 & \vec{N}_3 & \cdots & \vec{N}_n \end{bmatrix} \tag{2}$$

For each machine a set of charging rate corresponding to resource cost is represented by as a vector, says $\vec{R}_j$ for machine j. For the task, Let task vector $\vec{T}_i$ represents the *Resource Vector* of task i.

As a result, rental cost of running task i on machine j is computed by dot product of vector $\vec{T}_i$ and vector $\vec{R}_j$ (see Eq. 3 and Eq. 4).

An organization is responsible for specifying the charging rate of each resource under its administrative domain. For instance, Eq. 4 represents all possible costs when assigned the task to each compute node. Scheduler chooses the appropriate one based on turn-around times and these costs.

$$C_{ij} = \vec{T}_i \bullet \vec{R}_j \tag{3}$$

$$C_{ij} = \begin{bmatrix} R_{i1} & R_{i2} & R_{i3} & \cdots & R_{im} \end{bmatrix} \bullet \begin{bmatrix} R_{j1} \\ R_{j2} \\ R_{j3} \\ \vdots \\ R_{jm} \end{bmatrix} \tag{4}$$

An application is represented by a set of parameter to reflect its behavior. The essential one is workload in the unit matching to the execution rate defined above. It is not enough in some case that an application needs a large input files, produces big output files, or needs to be staged to the target machine before running. In this case, staging time is computed by file size and available bandwidth during transfer. However, if staging time is so small comparing to execution time, it is not necessary to include staging in the model. Unfortunately, most scientific applications need large data file and produces large output file.

For example, in case of a task is represented by amount of workload denoted by ω so a machine must define an execution rate, λ. Execution time, ϕ, of running this task on this machine is obtained by ω / λ. Network can also model in the same manner by defining amount of data to transfer through the network as σ and bandwidth between them as B. Transfer time η is obtained by σ / B. In this paper, staging time and execution time have been combined to simplify the model so final execution time in this paper is defined in Eq. 5.

$$e_{ij} = \phi_{ij} + \eta_{input} + \eta_{executable} + \eta_{output} \tag{5}$$

III. Grid Resource Scheduling

There are many well-known and well-perform heuristics for traditional computing platform like Network of Workstation or cluster, for example, MinMin, MaxMin, and Sufferage. Their original goal is to minimize makespan [15] of the whole heterogeneous computing system. However, it is not good enough for real-world application where resources must be charged. The new goal is to optimize 2 parameters; makespan and rental cost of running. To satisfy these goals, new heuristics are proposed based on models defined in previous section. Since these heuristics used different techniques and parameters, it is named “Unified Economic Deadline Scheduling Algorithm” to identify its class clearer.

A. *Unified Economic Deadline Scheduling Algorithm*

Proposed heuristics are named CMinMin, CMaxMin, and CSufferage. Assuming that all tasks and all available machines are listed in M_v and m, respectively. Instead of determining only completion time, these heuristics define another criteria, Priority Index, denoted by p_{ij}. p_{ij} (See Eq. 6) is a function of execution time e_{ij} (see Eq. 5), ready time r_j, cost C_{ij}, accumulative cost c_j, and time period between completion time and deadline d. All variables have their own factor, w_1, w_2, w_3, w_4, w_5, respectively. The general objectives in resource owner view are to minimize time, minimize cost, and minimize over-deadline tasks. To minimize over-deadline tasks, it is necessary to schedule tasks early on machine that might finish them in time. However, it is worth to wait for other faster machines that also finish them in time.

$$p_{ij} = w_1 e_{ij} + w_2 r_j - w_3 C_{ij} - w_4 c_j + w_5 d \tag{6}$$

At the beginning, p_{ij} will be pre-computed for later use in each algorithm and are recomputed again to match machines status for next task assignment.

```
recompute(r,c,d,p)
while count(task_list) > 0:
    t,m = choose(p,task_list)
    if t and m:
        assign(t,m)
        delete(task_list,t)
    recompute(r,c,d,p)
```

Unified algorithm is shown above. First of all, priority indexes are computed for use in main loop. The main loop will run until all tasks in task list are assigned to machines. In the loop, the priority indexes and task list are considered to choose a task and its correspondent machine to run on. The task then is deleted from task list and p is recomputed.

B. *Applying the Model to Traditional Algorithm*

Generally, UED is a generic scheduling framework that is backward compatible with previous algorithms by adjustment of factors. For example, if $w_1 = w_2 = 1$, $w_3 = w_4 = w_5 = 0$ then priority index is representing completion time as usual. Other adjustments are shown in Table 1.

Table 1. Several factors of Unified Economic Deadline scheduling algorithm when applying to existing algorithms and proposed algorithms.

	w_1	w_2	w_3	w_4	w_5
MCT	1	1	0	0	0
MET	0	1	0	0	0
MinMin	1	1	0	0	0
MaxMin	1	1	0	0	0
Sufferage	1	1	0	0	0
CMinMin	x	x	x	x	x
CMaxMin	x	x	x	x	x
CSufferage	x	x	x	x	x

With UED, CMinMin, CMaxMin and Csufferage are able to handle circumstances that original heuristics cannot handle. For example, assuming there are 4 tasks in a batch and 2 machines, says A and B, with attributes in Table **2**. In this situation, MinMin assigns task T_1 to A, T_2 to A, T_3 to B, and T_4 to A. As a result, machine A took time 15 units and cost 18 units while machine B took time 18 units and cost 9 units. In contrast, MinMin gave makespan 18 units and rental cost 27 units. However, CMinMin with UED assigns task T_1 to

A, T_2 to B, T_3 to B, and T_4 to A. It turned out that CMinMin gave makespan 27 and rental cost 21.

Table 2. A sample batch consists of 4 tasks and 2 machines, A and B. Each task represented by corresponding executime time, cost and priority index obtained from Eq. 6.

	Executime time		**Cost**		**P**	
T	**A**	**B**	**A**	**B**	**A**	**B**
1	2	4	4	2	6	6
2	5	8	10	4	15	12
3	10	18	20	9	30	27
4	8	12	4	6	12	18

IV. Experimental Results

There are many kinds of simulation process and simulation model. These experiments in this section are based on HyperSim [16] which run on Event Graph Modeling. The defining feature of DES models is that they have state variables whose transits in simulated time are piecewise constant. State transitions only occur at discrete time epochs, which are designated as events. The Event List is responsible for determining which events occur and that the appropriate state transitions are executed. The occurrence of an event may trigger the occurrence of other events at later times. These future occurrences of events are implemented in a DES model by placing the appropriate scheduled events on the Event List. The Event List algorithm sorts the events in ascending temporal order and executed the simulation by always.

Table 3**.** Parameter of KU Grid and Thai Grid

Name	Processors	Execution Rate	Bandwidth (Mbps)	Grid
AMATA	15	1000	10	KU/Thai
GASS	12	1800	18	KU/Thai
MAEKA	16	1400	80	KU/Thai
MAGI	4	2200	100	KU/Thai
Optima	8	1000	0.7	Thai
CAMETA	16	1800	1	Thai
Palm	8	1000	1	Thai
Algorithm	12	1000	1	Thai

First experimental is to study characteristics of each heuristic in terms of relation between makespan and cost. Grid system is derived from KU Grid and Thai Grid [17] as shown in Table 1. All machines are initially no loads. These Grids are assumed that (1) faster machines might be more expensive; (2) each cluster has their own scheduler with MaxMin heuristic because MaxMin gives better results for common task pattern (exponential distribution).
The experimental is to randomly submit 10,000 tasks and schedule them to the specific cluster based on each heuristic. Inter-arrival time of each task is determined randomly in exponential distribution with mean of 50 seconds. Workloads are also in exponential distribution with mean of 30,000 million instructions. Each heuristic are reputably ran 1,000 times with different seeds. These experimental configurations are to bring up performance of the heuristics as many as possible. Interesting variables are recorded and plot average results in Fig. 1, Fig. 2, and Fig. 3 for makespan, cost, and cost-time ratio, respectively. Note that all tasks are finished within their deadline.

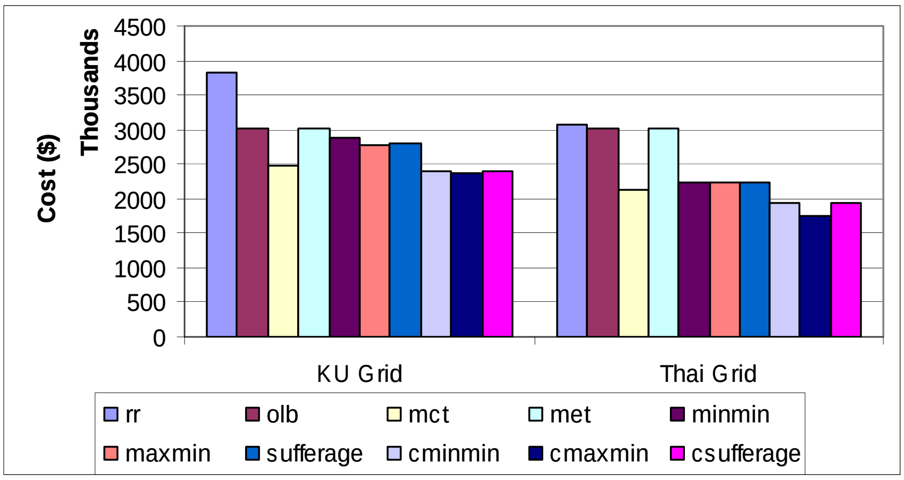

Fig. 1. Cost of KU Grid and Thai Grid

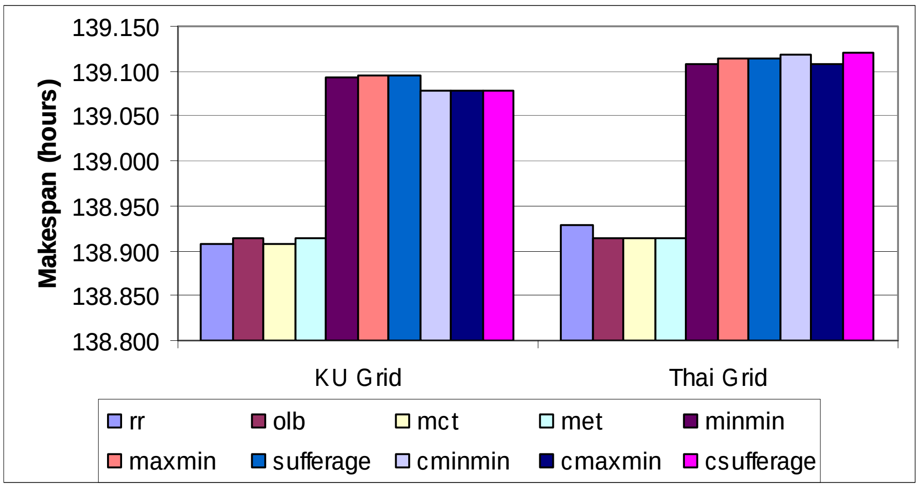

Fig. 2. Makespan of KU Grid and Thai Grid

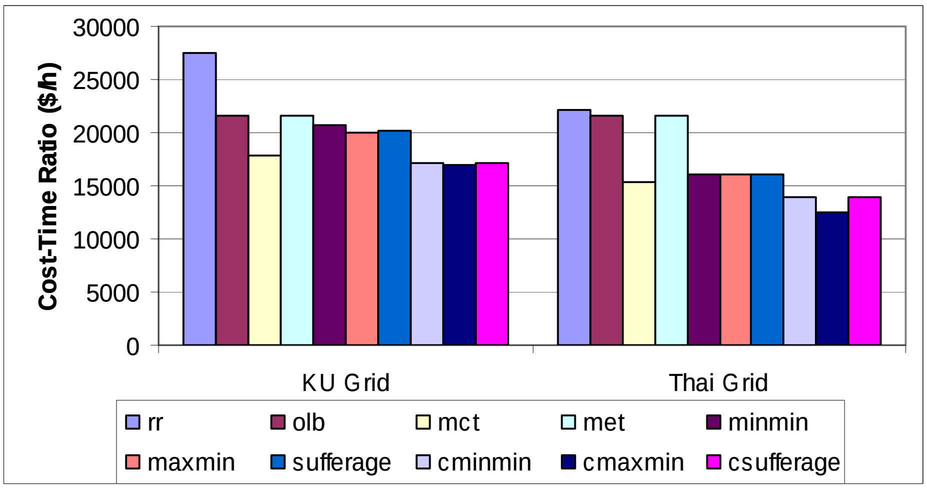

Fig. 3. Cost-Time Ratios of KU Grid and Thai Grid

Fig. 1 clearly shows that CMinMin, CMaxMin, and CSufferage scheduled all tasks to complete using cheaper cost comparing to all others algorithms including on-line scheduling algorithms. However, these schedules take slightly longer makespan (see Fig. 2) comparing to others because they usually try to minimize positive time between deadlines and finish time. Generally, a task might be

scheduled to finish at deadline while another task is scheduled to finish at its deadline before previous task. However, cost-time ratio in Fig. 3 shows the clear result. Proposed algorithms are outperforming comparing to other algorithms both on-line and batch.
In term of scalability, all algorithms perform well to take shorter time to finish all tasks when more machines are available.

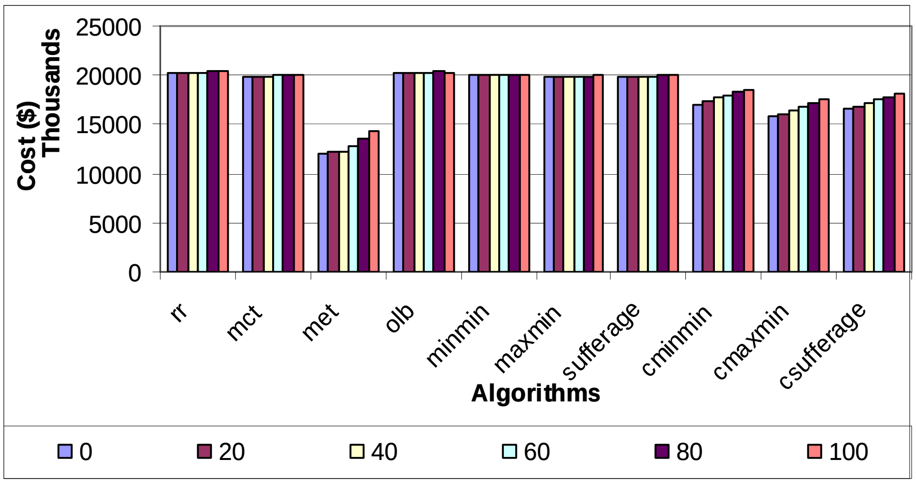

Fig. 4. Cost of different prediction error

According to all algorithms described in this paper, they are relying heavily on estimated execution time to predict completion time at every step. Generally, higher prediction error intends to bring lower performance because execution time and completion time are considered as a major parameter to match and to map each task on each machine. If inaccurate parameters have been obtained, these values will be propagated to next step forever.
Fig. 4 shows cost of running longer tasks on KU Grid with different level of prediction error, says 0% to +/-100%, in exponential distribution.

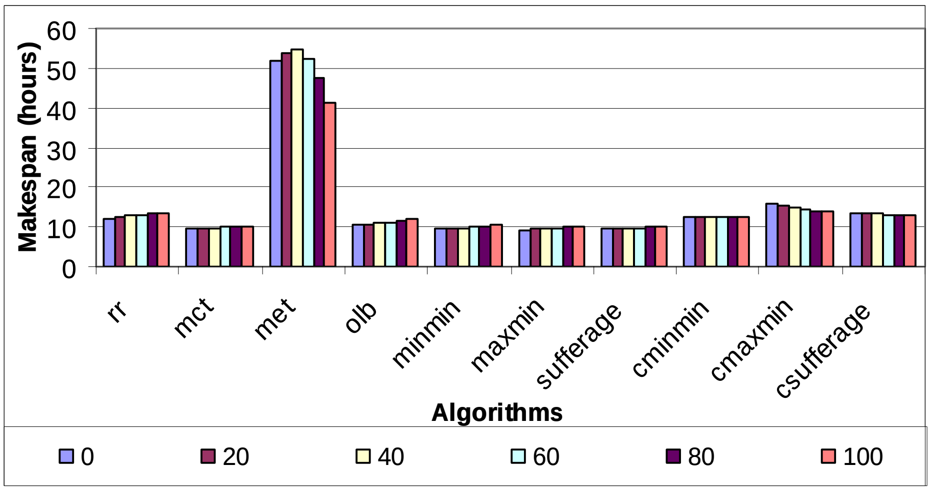

Fig. 5. Makespan of different prediction error

As a result, the prediction error only slightly effects to RR, MCT, OLB, MinMin, MaxMin, and Sufferage but increases rental cost for MET, CMinMin, CMaxMin, and CSufferage. The reason is 4 folds. Firstly, the first group of heuristics, says RR, does not consider any time related parameters so there is no effect at all. Secondly, the second group, e.g. MCT, makes decision based on completion time which error is easily propagated until simulation finished. However, their objective is mainly to minimize makespan so these costs are just uncontrollable variable. Fig. 5 shows their makespans are slightly increased due to prediction error. Thirdly, MET only determines execution time so it heavily rely on accurate prediction. In this case, makespan is increased until 40% and fall down because the error function is generated by exponential distribution. Higher percentage intends to give more stable prediction that parallel with correct values. Lastly, proposed algorithms reflect normal behavior that cost increases if error is increased. However, the final results shown in Fig. 6 clearly shows that proposed algorithms with proposed model outperform other algorithms even on high prediction error environment.

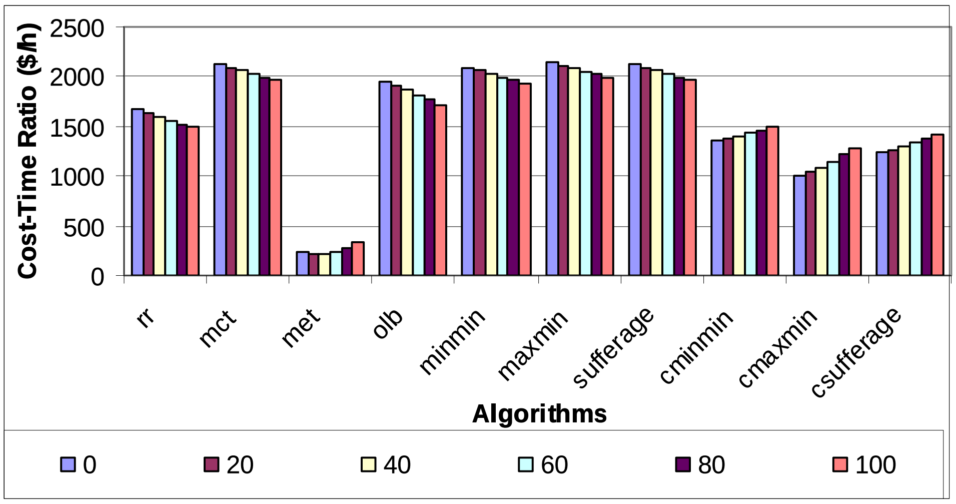

Fig. 6. Cost-time ratio of different prediction error

V. Conclusions

Grid is going to be next generation standard of internet protocol like TCP/IP today. It is widely used in many areas that resources are shared geographically distributed across administrative domain. To utilize Grid efficiently, one or more Grid scheduler is required somewhere to serve end-user like a single machine do in the traditional system. This infrastructure gives functionality to provide on-demand resources automatically. However, nothing is free of charge in the real world. Most scheduling heuristics today are designed to minimize makespan that usually intends to rapidly increased rental cost (a fast machine is more expensive than slower one).

This paper proposed an extensible performance model of resources scheduling problem for large-scale Grid system. The proposed model consists of time, cost, and deadline. Basic idea of basic win-win heuristic is to minimize both time and cost so that task owner gets results back as fast as possible with minimum price. However, time might increase when decreases cost and in contrast, cost might increase when time decreases. Deadline is the third factor to balance time and cost. In addition, three modified heuristics are proposed to show usage of the model.

Since computational Grid is uncontrollable infrastructure, proposed heuristics and other well-known heuristics are evaluated in HyperSim, discrete event simulation library. The experimental results clearly show that (1) the three modified heuristics using proposed model outperform their original heuristics and other heuristics in term of lower price per unit of time (cost-time ratios). (2) All heuristics including proposed ones perform better on larger system because more tasks executed simultaneously before their deadline. (3) Prediction accuracy of execution time takes slightly effect to performance for short-time applications.

Other interesting topics are to study effect of inaccurate measurement of other many parameters, e.g. network bandwidth, network latency, and so on. Moreover, data staging could be performed during other task is being run to save lots of time. However, the run task might be slower than usual especially accessing storage. To model this behavior correct, it is necessary to model data access of the application. In this paper, there is only one application type. In realistic, many kinds of application are submitted to the scheduler at the same time. Quality of Service can be added to prioritize each task based on end-user.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal Supercomputer Applications*, vol. 15, pp. 200-222, 2001.

[2] I. Foster and C. Kesselman, *The Grid: Blueprint for a Future Computing Infrastructure*: Morgan Kaufmann, 1998.

[3] I. Foster and C. Kesselman, "Globus: A Toolkit-Based Grid Architecture," in *The Grid: Blueprint for a Future Computing Infrastructure*, I. Foster and C. Kesselman, Eds.: Morgan Kaufmann, 1998, pp. 259-278.

[4] M. Muthucumaru, A. Shoukat, S. Howard Jay, H. Debra, and F. F. Richard, "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems," in *Proceedings of the 8th Heterogeneous Computing Workshop*. San Juan, Puerto Rico: IEEE Computer Society, 1999.

[5] S. Y. Lee and C. H. Cho, "Load Balancing for Minimizing Execution Time of a Target Job on a Network of Heterogeneous Workstations," in *Proceedings of the 6th Workshop on Job Scheduling Strategies for Parallel Processing*. Cancun, Mexico: Springer-Verlag, 2000.

[6] 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*. Cancun, Mexico: IEEE Computer Society, 2000.

[7] P. Christopher, L. Paul, and G. Mark, "Practical Heterogeneous Placeholder Scheduling in Overlay Metacomputers: Early Experiences," in *Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing*. Edinburgh, Scotland: Springer-Verlag, 2002.

[8] C. Su-Hui, C. A.-D. Andrea, and K. V. Mary, "The Impact of More Accurate Requested Runtimes on Production Job Scheduling Performance," in *Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing*. Edinburgh, Scotland: Springer-Verlag, 2002.

[9] I. D. B. Anca and H. J. E. Dick, "Local versus Global Schedulers with Processor Co-allocation in Multicluster Systems," in *Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing*: Springer-Verlag, 2002.

[10] S. Vijay, K. Rajkumar, S. Srividya, and P. Sadayappan, "Distributed Job Scheduling on Computational Grids Using Multiple Simultaneous Requests," in *Proceedings of the 11 th IEEE International Symposium on High Performance Distributed Computing HPDC-11 20002 (HPDC'02)*: IEEE Computer Society, 2002.

[11] D. Abramson, J. Giddy, and L. Kotler, "High Performance Modeling with Nimrod/G: Killer application for the global Grid," in *Proceeding of the International Parallel and Distributed Processing Symposium*. Cancun, Mexico: IEEE Computer Society Press, 2000.

[12] S. V. Sathish and J. D. Jack, "A Metascheduler For The Grid," in *Proceedings of the 11 th IEEE International Symposium on High Performance Distributed Computing HPDC-11 20002 (HPDC'02)*: IEEE Computer Society, 2002.

[13] C. Henri, O. Graziano, B. Francine, and W. Rich, "The AppLeS parameter sweep template: user-level middleware for the grid," in *Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*. Dallas, Texas, United States: IEEE Computer Society, 2000.

[14] E. Carsten, H. Volker, and Y. Ramin, "Economic Scheduling in Grid Computing," in *Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing*. Edinburgh, Scotland: Springer-Verlag, 2002.

[15] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Englewood Cliffs, NJ: Prentice Hall, 1995.

[16] S. Phatanapherom, P. Uthayopas, and V. Kachitvichyanukul, "Fast Simulation Model for Grid Scheduling using HyperSim," in *Proceedings of Winter Simulation Conference 2003*. New Orlean, 2003.

[17] V. Varavithya and P. Uthayopas, "ThaiGrid: Architecture and Overview," in *Proceeding of South East Asia High Performance Computing 2001*. Kasetsart University, Bangkok, Thailand, 2001.

SUGREE PHATANAPHEROM is a research assistant in HPCNC at Kasetsart University, Thailand. He received his M.Eng in Computing Engineering from Kasetsart University. His recent work has involved Grid resource scheduler, simulator, and algorithms. His email address is <sugree@hpcnc.cpe.ku.ac.th>.

PUTCHONG UTHAYOPAS is an Assistant Professor in Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Thailand. He is also being the core member of ApGrid organization and participating actively in the construction of Asia Pacific grid testbed. His email address is <pu@ku.ac.th>.