Achieving of Tabu Search Algorithm for Scheduling Technique in Grid Computing Using GridSim Simulation Tool: Multiple Jobs on Limited Resource

Mohd Kamir Yusof
Faculty of Informatics, Universiti Sultan Zainal Abidin
Kampus Gong Badak
21300 Kuala Terengganu, Malaysia
Email: mohdkamir@udm.edu.my

Muhamad Azahar Stapa
Faculty Science Computer and Information System, Universiti Teknologi Malaysia
81310 Skudai, Johor, Malaysia
Email: azaharstapa@yahoo.com

Abstract

Grid computing is a form of distributed computing involves coordinating and sharing computing, application, data storage or network resources across dynamic and geographically dispersed organization. One of the scheduling techniques in Grid Computing is Tabu Search algorithm. A good scheduling algorithm is normally shows lower value of total tardiness and schedule time. The implementation Tabu Search algorithm was tested and evaluated on universal datasets using GridSim tool. The results indicate performance of tardiness is directly related to number of machines up to certain number of resources. Small and medium company can use grid in operation process because it saves cost and times.

Keywords: *Grid Computing, Grid Scheduling, GridSim, Multiple Jobs, Tabu Search.*

1. Introduction

Grid computing is a growing network application that can be expected to become an important and required component of the new global knowledge of the 21st century. The term 'Grid' was ideal to explain this environment as an analogy to the electric power grid that is a major pervasive, readily available resource that empowers multiple different devices, systems and environments at distributed site. Grid computing is about getting computers to work together. Grid will help to promote the internet to a true computing platform, combining the qualities of services of enterprise computing with the ability to share heterogeneous distributed resources everything from application, data, storage and servers [1]. Grid also refers to an infrastructure that enables the integrated, collaborative use of high-end computers, networks, database, and scientific instruments owned and managed by multiple organizations. Grid applications often involve with large amount of data or computing and often require secure resource sharing across organizational boundaries, and are thus not easily handled by todays "Internet and Web Infrastructures."

Nowadays, Grid Scheduling has been used in Grid environments. Grid scheduling is defined as the process of make scheduling decisions relating resources over multiple administrative domains. This process can include searching multiple administrative domains to use a single machine or scheduling a single job to use multiple resources at a single site or multiple sites. The scheduling problem, in general, has been studied broadly in many areas,

such as transportation systems, industrial control, and medical operations. Today the scheduling in a Grid computing involves much manual administrative work. In Grid Scheduling have two important meanings. Firstly, scheduling is a decision making function: to determine a schedule. Secondly, scheduling is a body of a theory; it is a collection of principles, models, technique and logical conclusion. Scheduling function is the allocation of resources over time to perform a collection of task raised in a variety of situations. There are four primary stages of the system approach [2]:

(i) Formulation stage is where the problem is identified and the criteria to guide decision making are determined.

(ii) Analysis stage is the detailed process of examining the elements of a problem and their relationships: It is also aimed at identifying the decision variables and relationships among them and the constraint they must obey.

(iii) Synthesis is the process of building alternative solution to the problem.

(iv) Evaluation is the process of comparing these feasible alternatives and selecting a desirable course of action.

Tabu Search algorithm is one of scheduling techniques in grid computing. TS is a meta-heuristic scheduling method used to guide optimization algorithm in the search for a globally optimal solution. The basic principle of TS is to pursue Local Search (LS) whenever it encounters a local optimum by allowing non-improving moves; cycling back to previously visited solutions is prevented by the use of memories, called Tabu lists, that record the recent history of the search, a key idea that can be linked to Artificial Intelligent concepts [3]. TS deal with various techniques for making the search more effective. These include method for exploiting better information that becomes available during search and creating better starting points, as well as more powerful neighborhood operators and parallel search strategies. Another important trend in TS (this is, in fact, a pervasive trend in the whole meta-heuristic field) is hybridization, i.e. using TS in conjunction with other solution approaches such as Genetic Algorithm (Fleurent and Ferland, 1996; Crainic and Gendreau, 1999), Lagrangean relaxation (Grunert, 2002), Constraint Programming (Pesant and Gendreau, 1999), column generation (Crainic, Gendreau and Farvolden, 2000) and integer programming technique [3].

Simple application has been develop by using TS algorithm. The purpose of this development is to test efficiency of TS algorithm in Grid Computing using multiple jobs on resource limited.

2. Methodology and Implementation

Tabu Search (TS) algorithm has been selected to apply in Grid Computing using multiple jobs on limited resource. The algorithm for TS illustrate as below:

```
K:=1
Generate initial condition
WHILE the stopping is not met DO
    Identify N(s). (Neighbourhood set)
    Identify T(s,k). (Tabu set)
    Identify A(s,k). (Aspirant set)
```

```
        Choose the best s' ∈ N(s,k) = {N(s) –
T(s,k)} ∪ A(s,k)
        Memories s' if it improves the previous
best known      solution
        s :s'.
        k :k+1.
    END WHILE
```

Figure 1. Tabu Search Algorithm

The most important function in TS is neighborhood function. The neighborhood function is a mapping which defines for each solution x a subset N(x) of solutions called neighborhood. Each solution of N(x) is called a neighbor of x. A local search algorithm starts from some initial solution and moves from neighbor to neighbor as long as possible while decreasing the objective of function value. To find a good solution (optimal or near optimal) see figure 2.

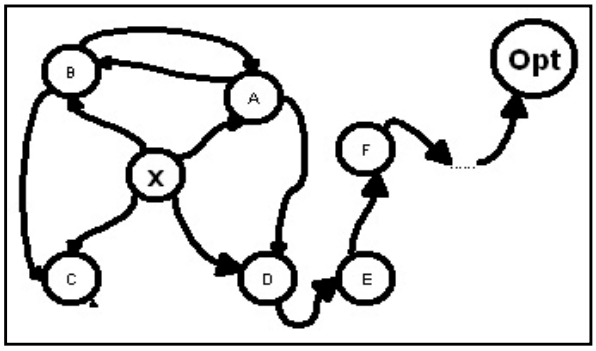

Figure 2. Mapping Solution

The TS approach seeks to counter the danger of entrapment by incorporating a memory structure that forbids or penalizes certain moves (i.e. render them tabu) that would return to recently visited solution. TS are a variable neighborhood method.

$$N(s) \rightarrow N(s,k) = N(s) - T(s,k) \qquad (1)$$

For short term memory to keep track of the *t* most recent solutions T(s,k) = the t most recent solution for keep track of the differences. Keep track of the *t* most recent "reserve moves". Let $M_k = \{\Delta 1, \Delta 2, \ldots, \Delta t\}$ denote the set of the last t moves. $T(s,k) = \{\Delta 1^{-1}(s), \ldots\ldots, \Delta D1^{-1}(s)\}$ $\Delta 1^{-1}(s)$ is the solution obtained from s by resorting some of the attributes that changed when move Δi was applied for aspiration level conditions. This may forbid attractive unvisited solutions.

i. *To get an effective implementation of TS*

S = Current solution
F(s) = Objective function
N(s) = Neighborhood
T(s,k) = Tabu list

ii. *S: From one to many*

S = s1 ∪ s2 ∪ ..

Each time define a neighborhood N(s,k) that depends on S: N(S,k) ∪ N(s2,k) ∪ …..

Intensification: in S there are the best found solutions.
Diversification: in S there the "different" found solutions.

iii. *F(s): Penalty approach*

Change the objective function value by adding an extra term:

Intensification: the term will penalize solutions far from the current solutions.
Diversification: the term will penalize solutions close to the current solutions.

iv. *Satisfiability problems*

Have several constraints. A penalty function is created for every constraint. The goal is to minimize the weighted sum of the penalty functions.

a. Diversification: decrease the highest weights.
b. Intensification: increase the highest weights.
Alternatives: decrease (increase) the weight of a constraint that always satisfied (violated).

v. *N(s): Change neighborhood function*

Idea: by changing the neighborhood function you have different local minima → good to diversify.

Example 1: TSP (2-opt, 3-opt)
Example 2: Job Shop (moving jobs, shifting bottleneck)

vi. *The Neighborhood function used is based on moving operations*

Figure 3 and figure 4 shows the moving the job in Tabu Search for minimize time for running the jobs.

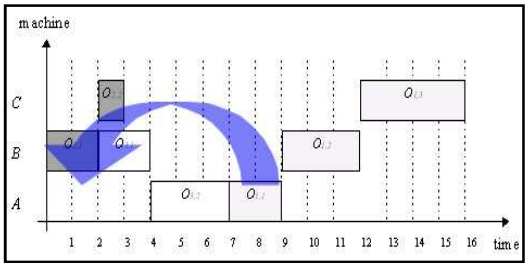

Figure 3. Moving the job in Tabu Search

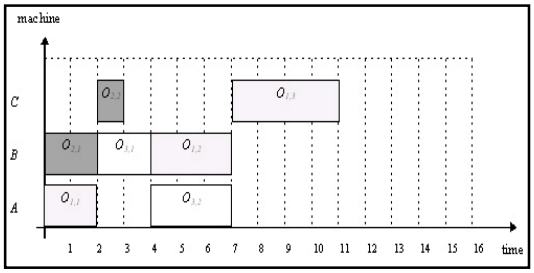

Figure 4. After moving in Tabu Search

vii. *N(s): Change neighborhood size*

Intensification: isolates regions of the neighborhood containing moves with desirable features.

Diversification: enlarge the neighborhood

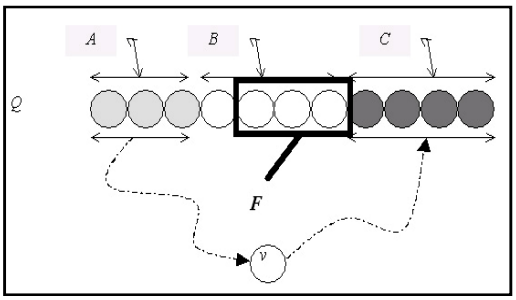

Figure 5. Neighborhood size

In figure 5 shows *F* is a subset of feasible insertions in which is guaranteed that there is an optimal insertion of *v*.

viii. *T(s, k): Tabu list types*

The choice depends on the problem.

Hint: test the tabu status of the move in constant time

a. Small number of different attributes
b. Each move is characterized by a constant number of attributes (use array or matrix and store the iteration number) Otherwise; use balanced binary tree

Example: Tabu search in constant time

Neighborhood: obtained by moving operation *O* from *A* to machine *B*.
Attribute: (*O*, *B*).
Tabu list type: *N** m matrix TM

Updating: If a move with attribute (*O, B*) is performed → TM (*O, A*) = iter + T.
Check: A move with attribute (*O, B*) is tabu if TM(*O,B*) > iter

ix. *T(s, k): Tabu list size*

No single rule (even an empirical one) gives good sizes.

a. Grow the tabu list size with size of the problem
b. Charge the tabu list size with the solution quality

By testing:

a. Find the minimum length that avoids cycles
b. Find the maximum length that avoids big deterioration of solution quality.

Example shows on figure 6 and figure 7.

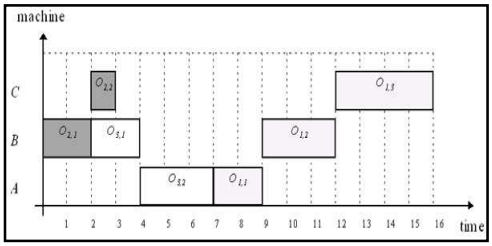

Figure 6. Tabu list size

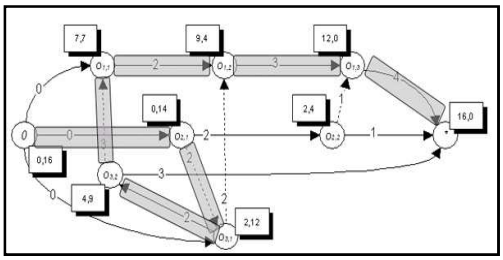

Figure 7. Critical path

The Tabu status length T is dynamically defined for each operation O and each solution. It equal to the number of operations of the current critical path P plus the number of alternatives machines available operation O, i.e.

T: $|P| + |M(O)|$ (2)

$|P|$ ~ instance size
$|P|$ ~ quality of the current solution

In order to diversity the search it may be unprofitable to repeat the same action often if the number of candidate actions is "big" or the solution quality is low, in some

sense, when |P| is "big". Furthermore, the tabu list status length of O is augmented by $|M(O)|$ in order to diversify the machine assignment of O [4].

3. Experimental Results

In this section discuss about implementation and results which were obtained from the successful runs of Grid Simulation tools. The results retrieved from identification of data will be analyzed and distributed by looking at the total tardiness and schedule time.

3.1. Implementation Description on GridSim

GridSim is the simulation tool environment that can simulate scheduling and execution of different types of non preemptive jobs in both static and dynamic fashion on resources composed of parallel and heterogeneous machines. System administrator demands on resource utilization can be satisfied by schedule time minimization and user requirements can be handled through optimization of the total tardiness of all jobs. The simulation environment allows an easy testing for the scheduling algorithms.
Simulator is modular, composed of independent entities which correspond to the real world. It consists of the centralized scheduler, job generator, job submission, and the Grid resources. These entities communicate together by message passing. Currently Grid users are not directly simulated but a job generator attached to the job submission system used to simulate job arrivals. Description of entities is depicted in table 1.

Table 1. Description of entities communications scheme

Job submission system	**Simulate job arrivals.**
Job generator	It generates new synthetic job that appear during simulation run.
Scheduler	Responsible for schedule generation and further optimization. This schedule may change in time as some jobs are already finished while new ones appear. Keep the scheduler extensible it was designed as a modular entity composed of three main parts.
Grid resource	Responsible for the job execution. The resource is selected by the scheduler and job is then submitted by the job submission system.

3.2. Communication Scheme

Common communication scheme between the job submission system, scheduler, and Grid resource is depicted in figure 8. Job submission system submits job descriptions to the scheduler. The scheduler uses the list of all available Grid resources and their parameters such as the number of CPUs and their rating.

The scheduler is centralized,; therefore it has information about and access to all available resources in the system. On the basis of this information scheduler generates separate schedule for each Grid resource. Using these schedules and also information of jobs currently in execution the scheduler is able to approximate various parameters of the schedule such as schedule time or expected tardiness for the jobs before their execution and completion. This allow the scheduler to compare tardiness and schedule time criteria and select the better one.

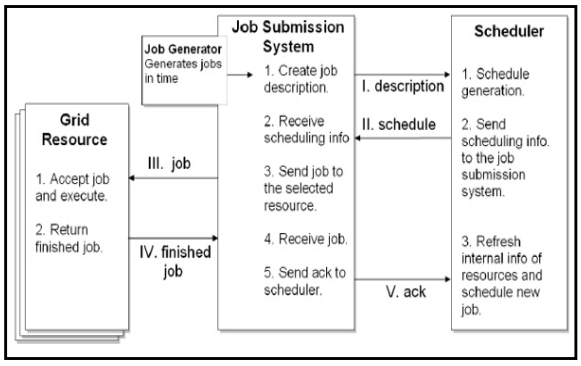

Figure. 8. Communication Scheme [17]

Refer to the constructed schedule and the current simulation time, the scheduler responds to the submission system with scheduling system with scheduling information, i.e. provides information which resources are selected to select to execute the job. It is important to notice that this communication is asynchronous. Job submission system does not wait for the response from the scheduler and sends the job descriptions as the new jobs that are available from the job generator. On the other hand the scheduler sends the scheduling information to the job submission system according to the current load of the resources. The scheduler works with the job description while the job stays with the job submission system. This helps to save scheduler's network bandwidth and prevents the scheduler to become a bottleneck of the whole system. Once some job is finished, the job submission system sends the acknowledge message to the scheduler. The scheduler will update its internal information of the current resource load. It is also an impulse to check whether another job should be sent on a resource to prevent the resource from being idle.

3.3. Analysis of Results

The result shows the difference schedule time, total tardiness vs. number of machine. Tests were done from two to eight machines simulated in GridSim. Single machine situation was not tested because Tabu Search only distributed jobs between one than machine. Figure 9 shows the result in graph analysis. The experiments were run tests for 400 jobs. The release date is taken as a hard constraint in the sense that the job is started to be executed while the due date is a soft constraint which may be compiled at the end of the scheduling. The data sets were generated synthetically and results were averaged from similar 20 data sets. The

scheduler optimizes the total tardiness of all jobs compares the result between tardiness and schedule time to achieve optimization.

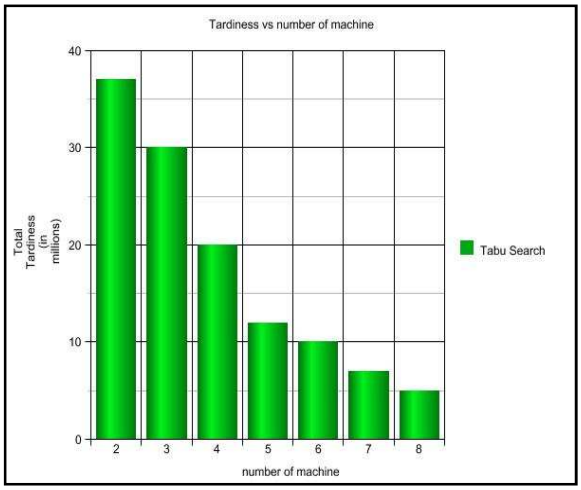

Figure 9. Total tardiness for Tabu Search in dynamic situation

Same number of jobs is tested for all the simulations. With machines increase the number of the size of schedule decreases due to the jobs previously scheduled were already finished. The algorithm in each run works with the smaller number of jobs. It is natural that in such cases the calculation of expected tardiness for unfinished jobs on one machine is faster than more than one machine.

In the first scenario the Tabu Search was run for each newly arrived job. In the other two cases the Tabu Search was run on two (2) until eight (8) machine with same size of data sets. The figure 10 and figure 11 shows the total tardiness and the schedule time required to generate the schedule when using Tabu Search. In figure 12 the graph pattern for the number of machines and the number of jobs are related each other. When the Tabu Search optimization is run after every five (5) or ten (10) jobs or task, the total tardiness is considered stable number of machines for all.

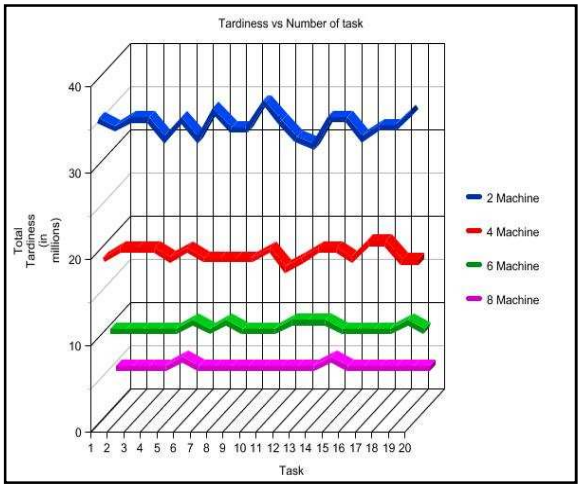

Figure 12. Total tardiness for Tabu Search after number of machines

Figure 12 shows tardiness for 20 jobs with different numbers of machine. At eight (8) machines the optimum value has been archived. From six (6) machines to eight (8) machines that total tardiness achieve optimal level of reduction. Al this level, increasing number of machine for more than eight (8) will not make significant different for the amount of 20 jobs. The value of tardiness and schedule time will effected when more than 20 jobs are running at same number of machine. The different tardiness value between two (2) machines and four (4) machines is almost while between four (4) to eight (8) machines, the different value is significant reduce.

The tardiness parameter shows that the result can get the optimum value at the uses of eight (8) machines. This result can give benefit to the company or user to used grid at limited resources, this solution will reduce the cost of processing and implementation time for the small and medium company.

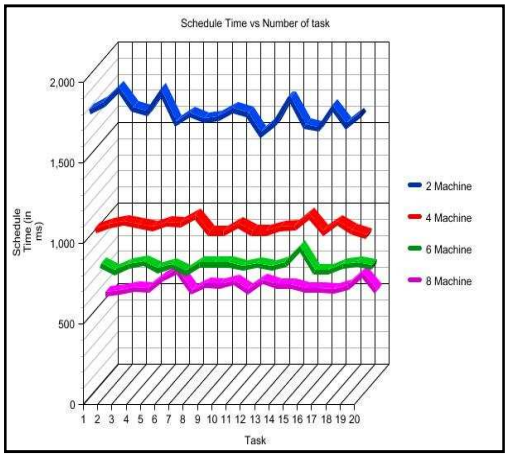

Figure 13. Schedule time for Tabu Search after number of machines

Figure 13 shows the time required to schedule different number of task when using Tabu Search. Tabu Search becomes stable with the growing number of machines. From 4 to 8 machines the values of schedule time stay at the level with the small difference value.

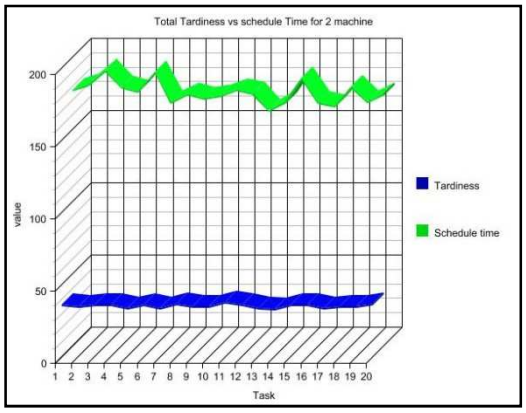

Figure. 14. Total tardiness vs. schedule time for two machines.

Figure 14 shows the value of tardiness and schedule time for two highly marked machines is highly difference. Two machines does not properly distributed task at the optimum time. 20 jobs circulations in the process should wait respective turns. Long waiting period is required for distribution task to the source. While resources must bear the burden process heavy to resolve 20 jobs with limited machines.

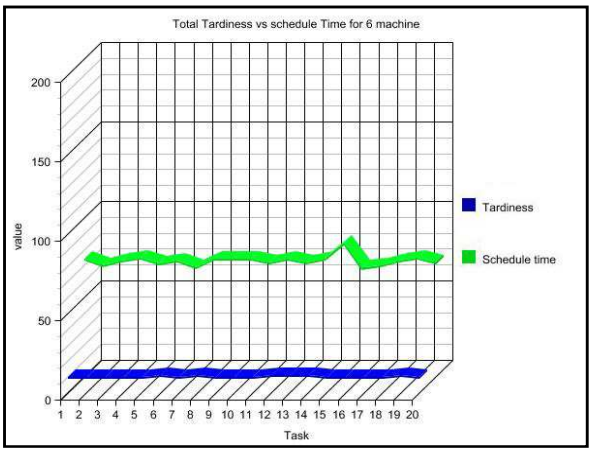

Figure 15. Total tardiness vs. schedule time for six (6) machines

In figure 15, the use of six (6) machines for distribution of 20 jobs will reduce the value tardiness as well as schedule time. Value of schedule time fall to the level 80 compared with two (2) machines previously at the level 200. At this level the end user can get good process time and result for resource at same level with using eight (8) machines

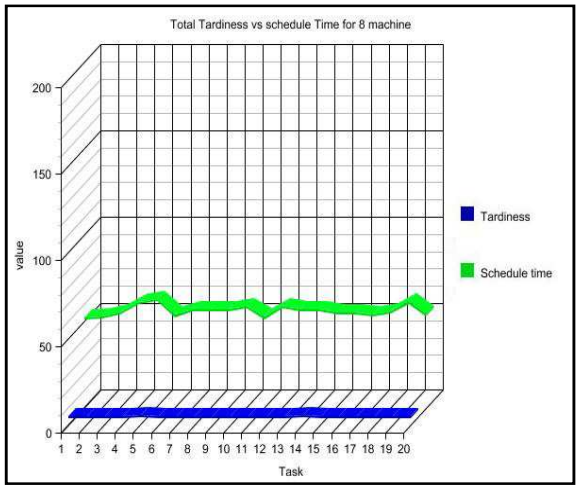

Figure 16. Total tardiness vs. schedule time for eight (8) machines

Figure 16 demonstrates the value of tardiness getting smaller from values before it. Used eight (8) machines can show the value schedule time stay at the level 60 compared to level 80 when six (6) machines are used. This different value of tardiness and schedule time is very important, to identify the better scheduling technique. The suitable implementation on a real grid is achieved when the different value tardiness and schedule time become small.

The value of tardiness and schedule time can give big different for selected best scheduling technique. The minimum of tardiness among selected scheduling techniques can give the different job distributions to get the minimum time process in grid environment and can save the cost of the process. Distributed the job to the resource with good schedule time and finish before due date is the objective to the end user, the cost of money and the save of time.

4. Conclusion

It is clear that the total tardiness strongly depends on the number of available machines. Total tardiness becomes higher when the number of machine small for scheduling technique. Tabu Search makes improving moves in all tested situations. When the Tabu Search optimization is run after 20 jobs, the total time decrease is acceptable, while the resulting schedule is still very good among all the compared schedules. As expected, the time required to perform Tabu Search in the dynamic situation becomes stable with the growing number of available machines.

The test used the same number of jobs for all the simulations, i.e. with more machines the size of the schedule decreases because the jobs previously scheduled are already finished and the algorithm in each run works with the smaller number of jobs. This result can be good parameter to select the best scheduling to use. The cost and time to implementation on real grid can save before the end user select which technique to use. Small and medium company can use grid for operation process at limited number of machines, this company can save the cost for buying the high power of machines too.

Acknowledgement

Special thanks are owned to my friend Che Mat Ismail and Masturah binti Haji Rashid for support and advice.

References

[1] L. Zhang, J. Chung, Q. Zhou. Developing Grid Computing Applications.http://www.106.ibm.com/developers/grid/library/gr-grid1/, Oct. 2002 (Discover Grid Computing, developer Works Journal) February 2003, 14-19.

[2] Deris, S. Studies on Intelligent Optimization Techniques for Planning, Scheduling. And Timetabling. Graduate School College of Engineering, Osaka Prefecture University, Japan: Thesis PhD 1997.

[3] M. Gendreau. An Introduction to Tabu Search. Canada: Universite de Montreal, 2002.

[4] K. Dowsland, Variants of Simulated for Practical Problem Solving. V. Rayward-Smith editor, Applications of Modern Heuristics Methods, Henley-on-Thames: Alfred Water Ltd. 1995.

[5] A. L. Pariera, V, Muppavarapu, and S.M. Chung. Role-Based Access Control for Grid Database Services Using the Community Authorization Service. IEEE Trans. On Dependable and Secure Computing, vol 3, no. 2, 2006, pp. 156-166.

[6] R. Buyya, D. Abramson, and J. Giddy. A Case for Economy Grid Architecture for Service-Oriented Grid Computing. Proceedings of the International Parallel and Distributed Processing Symposium: 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), April 23, 2001, San Francisco, California, USA, IEEE CS Press, USA, 2001.

[7] I. Foster and C. Kesselman. Globus: A Metacomputing Intrastructure Toolkit. International Journal of Supercomputer Applications. 11(2): 115{128, 1997}.

[8] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed Resource Management for High Throughput Computing. Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, July 28-31, 1998, Chicago.

[9] H. Dail, G. Obertelli, F. Berman, R. Wolski, and Andrew Grimshaw. Application-Aware Scheduling of a Magnetohydrodynamics Application in the Legion Metasystem. Proceedings of the 9th Heterogeneous Computing Workshop, May 2000.

[10] S. Smallen, W. Cirne, J. Frey, F. Berman, R. Wolski, M. Su, C. Kesselman, S. Young, and M. Ellisman. Combining Workstationsand Supercomputers to Support Grid Applications: The Parallel Tomography Experience. Proceedings of the 9th Heterogeneous Computing Workshop, May 2000.

[11] T. Eren and E. Guner. Minimizing Total Tardiness in a Scheduling Problem with a Learning Effect. Science Direct, vol 31, issue 7, july 2007, pp. 1351-1361.
[12] Lin, S., Goodman, E. and Punch W. (1997). A Genetic Algorithm Approach to Dynamic Job-Shop Scheduling Problems, In: Proc. 7th International Conf. on Genetic Algorithms, Morgan Kaufmann Publishers, San Francisco, pp. 481-488.
[13] Negnevitsky, M. (2005). Artificial Intelligence A Guide to Intelligent Systems, Addison-Wesley, pp. 222-223
[14] Dorigo M, Di Caro G. The ant colony optimization meta-heuristic. In: Corne D, Dorigo M, Glover F, editors. New ideas in optimization. London, UK: McGraw Hill; 1999. p. 11–32.
[15] W. Cheung, H. Zhou. Using Genetic Algorithms and Heuristics for Job Shop Scheduling with Sequence-Dependent Setup Times, Annals of Operations Research 107, 65–81, 2001 Kluwer Academic Publishers.
[16] Jia Yu and R. Buyya, A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms , Grid Computing and Distributed Systems (GRIDS) Laboratory Dept. of Computer Science and Software Engineering The University of Melbourne, VIC 3010 Australia.
[17] C. Fayad, Jonathan M. Garibaldi and D. Ouelhadj, Fuzzy Grid Scheduling Using Tabu Search, IEEE. 1-4244-1210-2/07, 2007.
[18] Shajulin Benedict & V. Vasudevan, Improving scheduling of scientific workflows using Tabu Search for Computational Grids, Information Technology Journal 7(1):91-97, 2008.
[19] D. Klusa·cek, L. Matyska, and H. Rudova. Local Search for Grid Scheduling. Doctoral Consortium at the International Conference on Automated Planningand Scheduling (ICAPS'07), Providence, RI, USA, 2007.
[20] D. Abramson, R. Buyya, and J. Giddy. A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker, Future Generation Computer Systems (FGCS), 18(8): 1061-1074, Elsevier Science, The Netherlands, October 2002.

Mohd Kamir Yusof obtained her Master of Computer Science from Faculty of Computer Science and Information System, Universiti Teknologi Malaysia in 2008. Currently, he is a Lecturer at Department of Computer Science, Faculty of Infomatics, Universiti Sultan Zainal Abidin, Terengganu, Malaysia.

Muhamad Azahar Stapa obtained her Master of Computer Science from Faculty of Computer Science and Information System, Universiti Teknologi Malaysia in 2008. Currently, he is a teacher a SMK Teknik Kangar, Perlis, Malaysia.