2013 IEEE International Conference on Systems, Man, and Cybernetics

Hyper-Heuristic Based Resource Scheduling in Grid Environment

Rajni Aron and Inderveer Chana
Computer Science and Engineering Department
Thapar University
Patial, India
Email: rajni@thapar.edu, inderveer@thapar.edu

Ajith Abraham*
IT4Innovations, VSB-Technical University of Ostrava,
Czech Republic
Machine Intelligence Research Labs (MIR Labs), WA, USA
E-mail: ajith.abraham@ieee.org

***Abstract*—An efficient management of the resources in Grid computing crucially depends on the efficient mapping of the jobs to resources according to the user's requirements. Grid resources scheduling has become a challenge in the computational Grid. The mapping of the jobs to appropriate resources for execution of the application in Grid computing is an NP-Complete problem. In this paper, hyper-heuristic based resource scheduling algorithm is designed to effectively schedule the jobs on available resources in a Grid environment. The performance of the proposed algorithm is evaluated using the GridSim toolkit. Empirical results illustrate that our algorithm outperformed the existing algorithm by minimizing cost and makespan of user's submitted applications.**

***Keywords*-Resource Scheduling; Heuristic Methods; Grid Computing.**

I. Introduction

Grid computing is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [1]. The ultimate goal of Grid computing is to provide the computing facility to users like power Grid without knowing the detailed characteristics of the source. The base of Grid computing is a resource. To manage resources in Grid environment is a challenging task. So Grid resource management has become one of the most important areas of Grid computing. Grid resource management can be defined as a process of identifying requirements of the resources, matching resources to the applications, allocating those resources and finally scheduling and monitoring the Grid resources over time in order to run Grid applications as efficiently as possible. Grid resource management system is required to perform resource management decisions which include resource provisioning and scheduling, while maximizing the Quality of Service (QoS) metrics delivered to the clients. Grid scheduling is the main key challenge of the Grid resource management system because in this, application should be mapped to the appropriate resource while fulfilling the user's requirements.

Grid scheduling is defined as the process of making scheduling decisions involving allocating jobs to resources over multiple administrative domains [2]. The mapping of jobs to appropriate resources for execution of application in Grid computing is an NP-Complete problem [7]. NP-complete problems are often solved using heuristic methods. Heuristic approaches can be easily applied to Grid scheduling problems because Grid scheduling has various important issues such as heterogeneity of resources, dynamic and autonomous nature of Grid resources and finally the resource providers and resource consumers have different policies for execution of their applications. Hyper-heuristic can be seen as a high-level methodology, which when given a particular problem instance or a class of instances and a number of low-level heuristics, automatically produced an adequate combination of the provided components to effectively solve the given problems [3].

The paper is structured as follows: Section 2 discusses related work. In section 3, a description of Grid resource scheduling model has been presented. In section 4, we present Tabu search based hyper-heuristic with great deluge resource scheduling algorithm for Grid environment. Section 5 presents the experimental setup used for the performance evaluation and results. We provide the conclusion in section 6.

II. Related Works

Many traditional Grid scheduling algorithms have been proposed that have some features in common, that are performed in multiple steps to solve the problem of matching jobs/applications needs with resource availability while providing QoS. To find the best pair of jobs and resources is an NP-complete problem.

Abraham et al. used nature's heuristics namely Genetic Algorithm (GA), Simulated Annealing (SA) and Tabu Search (TS) for scheduling of jobs on computational Grids. Authors illustrated that GA performs better than TS and SA for scheduling of the jobs to exact resources but hybrid heuristic algorithms perform better than GA approach as it minimizes the time required for scheduling the job [5]. A fuzzy reputation based ant algorithm for Grid scheduling has been designed in[14]. Authors used fuzzy logic trust model for trust value aggregation through fuzzification. Chen et al. have used universal utility optimization function to design economic Grid resource scheduling algorithm. They considered time and cost parameters to design resource scheduling algorithm [15]. Authors used meta-heuristic to design scheduling algorithm instead of hyper-heuristic.

978-1-4799-0652-9/13 $31.00 © 2013 IEEE
DOI 10.1109/SMC.2013.187

In [16], a QoS guided min-min heuristic is presented which can guarantee the QoS requirements of particular tasks and minimize the makespan at the same time. They have considered a single objective problem. Carretero et al. used GA for job scheduling in large-scale Grid applications. They have done several variations for GA operators in order to identify which worked best for the problem [17]. Gonzalez [6] et al. used ad-hoc (immediate and batch mode) scheduling methods to design hyper-heuristic approach for scheduling of jobs on the Grid nodes. A scheduling model for resource scheduling has been designed using heuristic methods by Bhanu et al. [4]. They have used Longest Job Faster Resource (LJFR) heuristic and Shortest Job Faster Resource (SJFR) heuristic method for resource scheduling. They did not consider the cost and makespan for independent job scheduling in the Grid environment. Our proposed implementation of hyper-heuristic based resource scheduling algorithm minimizes the cost and makespan simultaneously.

A. Grid Scheduling: Problem Formulation

To find the best resource to a corresponding job is a tedious task and the problem of finding the best resource - job pair according to user's application requirement is a combinatorial optimization problem.

In order to formalize the problem instance, we have used the Brun et al. [7] computational model. Under the ETC simulation model for problem formulation, we have considered the following constraints: 1) Each job to be scheduled for application's execution has a unique id. 2) Jobs are independent and indivisible and 3) Arrival of jobs for execution of application is random and jobs are placed in the queue of unscheduled jobs. The problem of finding the best resource - job pair according to user's application requirement is a combinatorial optimization problem. So, it is required to mathematically formalize to get an optimal solution. To consider this problem, we have taken a set of independent jobs $\{j_1, j_2, j_3, ...j_m\}$ to map on a set of heterogenous and dynamic resources $\{r_1, r_2, r_3.....r_n\}$.

$R = \{r_k | 1 \leq k \leq n\}$ is the collection of resources and n is the total number of resources. $J = \{j_i | 1 \leq i \leq m\}$ is the collection of jobs and m is the total number of jobs. The estimated time to compute value of each application/ job on each resource is assumed to be given by the user's supplied information, experimental data, job profiling and analytical benchmarking.

We have used a weighted sum function of makespan and cost to deal with their simultaneous optimization. We have transformed a bi-objective problem into mono-objective by using the weighed function.

B. Objective Function

In Grid scheduling, the main goal of the providers is to minimize the makespan where as the goal of the user is to minimize the cost for Grid application. Fitness value is thus calculated as:

$FitnessFunction = \theta cost + \delta makespan$
cost = Min$(c(r_k, j_i))$ for $1 \leq k \leq n, 1 \leq i \leq m$
makespan= Min (F_{j_i}) for $j_i \in J$

where $0 \leq \theta < 1$ and $0 \leq \delta < 1$ are weights to prioritize components of the fitness function.

Cost(r_k, j_i) is the cost of job j_i which executes on resource r_k. Makespan is the finishing time F_j of latest job and can be expressed as Expected Time to Compute (ETC) job j_i on resource r_k . For calculating makespan, it is useful to define the completion time of a machine. Completion time indicates the time in which the machine/resource can complete the execution of all the previous assigned jobs in addition to the execution time of job j_i on resource r_k, as defined below.

$completion(r_k) = avail\_time_{r_k} \pm ETC(j_i, r_k)$

We can use the value of completion time to compute the makespan. This mapping is done with an objective of minimizing cost and makespan simultaneously.

III. Tabu search based Hyper-heuristic with Great Deluge Resource Scheduling Algorithm

A hyper-heuristic operates at a higher-level of abstraction. It selects a low-level heuristic that should be applied at any given time, depending upon the characteristics of the region of solution space currently under exploration [3]. The work presented in this paper is based on the four low-level heuristics as discussed in [10][13]. The process of hyper-heuristic is divided into two parts namely heuristic selection and heuristic acceptance. Heuristics selection method is very simple to select the low-level heuristics. Heuristics acceptance can be divided into deterministic and non-deterministic. In the proposed algorithm, we have used tabu search [8] as heuristic selection method and great deluge [9] as heuristic acceptance. Tabu search with great deluge based hyper-heuristic algorithm for resource scheduling deals with feasible solution for scheduling of resources in Grid environment. For job execution of independent job, best heuristic is found and then the procedure is repeated until all user's jobs have been scheduled.

A. Pseudo code of Algorithm

In this Section, we present the pseudo code of hyper-heuristic based algorithm for resource scheduling in the Grid environment. Low-level heuristics can be simple or complex and can be implemented as follows : 1) First of all, select the job to be scheduled. The heuristic selects a job from the list of unscheduled jobs and schedule it to the best available resource that is filtered from the resource provisioning list. 2) Move job j_i from its current resource to some other resource. 3) Randomly select a job and swap it with some other job. 4) Finally, remove a randomly selected job from the job

Algorithm 1: Tabu search based Hyper-heuristic with Great Deluge Resource Scheduling Algorithm

```
Data: number of jobs and number of available
      resource.
Result: mapping of the each job to the resources.
begin
    initialize Resource list[Number of Resources]
    initialize joblist[Number of Jobs]
    Input n= number of heuristics
    Initialize a random feasible solution
    joblist= get job to schedule()
    resourcelist= get available resources()
    select the best heuristic from non tabu list
    initialize the heuristic list h[h1,h2,h3...hn]
    boundary= F_current for m = 0 → m = k do
        i= selectheuristic(h_c)
        if F_i < F_current then
            Apply h_c
            F_current = F_i
        add h_c into the tabu list
        untill terminating condition satisfied
        Repeat for Heuristic
        while there are unscheduled jobs in the queue
        do
            for every resource is in resource list do
                get the next job from queue
                compute the fitness to schedule the job
                on the current resource
                schedule the job on the resource on the
                basis of fitness
        Repeat the every step until all jobs are allocated
```

pool already scheduled. This is the only heuristic which will move the search into an infeasible region because any job may be unscheduled. We make sure that the search can move back into its feasible region by un-scheduling the job that has other valid resources so that it can move into the next iteration.

- Job list and a random feasible solution is initialized. Then, resource list is obtained from resource provisioning unit after provisioning of user's requests [11].
- The task to choose best heuristic from low-level heuristics is started.
- The value of boundary is set with current feasible solution for accepting the best heuristic from non tabu list. It calls each heuristic which is not tabu.
- Bruke et al. [9] acceptance criteria of great deluge has been used in this algorithm. After the selection of heuristic, it's solution is compared with current solution.
- If the solution is less than the boundary value, then the heuristic is accepted for resource scheduling.
- After heuristic selection, heuristic will assign each job to resource from the queue of unscheduled jobs.
- Resource scheduling is performed till there are no unscheduled jobs in the queue.

IV. Performance Evaluation and Discussion

GridSim toolkit provides facilities of modeling, simulation of resources and network connectivity with different capabilities, configurations and domain [12]. Resources capability can be defined in the form of Millions Instructions Per Second (MIPS) as per Standard Performance Evaluation Corporation (SPEC) benchmark. Multiple user entities can submit tasks for execution simultaneously.

For experimental results, heterogeneous type of resources are considered. In general, each resource may contain a different number of machines, and each machine may have one or more than one Processing Elements (PE). In our results, we have assumed that each application/task which is submitted to the Grid may require varying processing time and input size and such type of task is defined in the form of Gridlets. Table 1 shows the characteristics of resources and Gridlets, that we have used for all our experiments. For our evaluation, we have derived a suitable workload from real machine traces. These traces have been obtained from Grid workload archive website [1]. 2000 user applications are generated according to the Lublin workload model [19]. The model specifies the arrival time, number of CPUs required, and execution time μ of each application. This model is derived from existing workload traces for rigid jobs and incorporates correlations between job runtimes, job sizes, and daytime cycles in job inter-arrival times. Using this generated workload, we have generated ETC matrix which is computed as the ratio of workload and computing capacity of machine vectors. No of jobs * no of resources gives the size of the matrix and its components are defined as $ETC(j_i, r_k)$. Rows of the ETC matrix demonstrate the estimated execution time for a job on each resource and the columns demonstrate the estimated execution time for a particular resource. $ETC(j_i, r_k)$ is the expected execution time of job j_i and the resource r_k. Each job can execute on each resource, and the estimated execution times of each job on each resource is known.

ETC matrices are classified into consistent and inconsistent matrices. Consistent matrix means that whenever a resource r_k executes the job j_i faster in comparison to r_l then the resource r_k executes all the jobs faster than r_l. Inconsistent matrix means that r_q may be faster in job execution than r_s for some cases and slower for others [7].

[1]More information about the real trace used can be obtained from the Grid Workload Archive at http://gwa.ewi.tudelft.nl/pmwiki/

Table I
SCHEDULING PARAMETERS AND THEIR VALUES

Parameter	Value
Number of Resource	70-150
Number of Gridlets	2000
Length of Job	1000 - 6000
Bandwidth	3000 or 7000 B/S
Number of machine per resource	1
Number of PEs per machine	1-5
PE ratings	10-60 MIPS
Cost per job	3 G−5G
File size	100 + (10-30%) MB
Job Output size	250 + (10-40%)MB

A. Performance Evaluation criteria

We selected two matrices, namely makespan and cost for evaluating the performance. The former indicates the execution time where as the latter indicates the cost per unit resources that are consumed by the users for the execution of their applications. The cost is measured in Grid dollars (G$).

B. Results

To validate our algorithm, 2000 jobs/applications and 70 - 150 resources are considered. We used an average of fifty runs in order to guarantee statistical correctness. We present the simulation result using Brun et al. simulation model with the Gridsim discrete event simulation so as to test the performance of hyper-heuristic based algorithm. To simulate the Grid environment, execution time for every job on resource is obtained from the Expected Time to Compute (ETC) matrix.

Test case 1: Performance for the High & Low-Heterogeneous Case

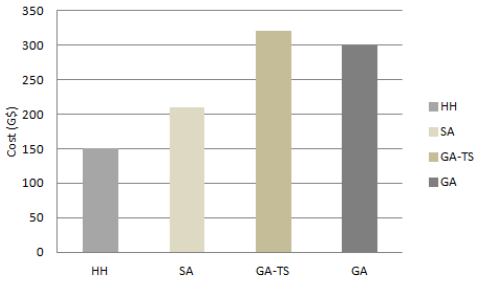

Figure 1. Comparison result for inconsistent and low machine heterogeneity

In this case, we evaluate the makespan and cost of the Grid applications in two different scenarios as (i) Same number of applications/jobs are sent and (ii) Different number of applications are sent. The pricing of resources may or may not be related to CPU speed. Thus, minimization of both makespan and cost of an application may conflict with each other depending on the price of the resources. Fig 1 - 8 show the makespan and cost of Tabu search based hyper-heuristic with great deluge resource scheduling vs Genetic Algorithm (GA), Simulated Annealing (SA) and GA-TS algorithms respectively. The most important characteristic applicable to real world scenarios is that how each algorithm responds to different heterogeneity of jobs and resources. A comparison of different makespans for both high and low resource/machine heterogeneity has been shown. In this analysis, high resource heterogeneity is simulated by each resource having a random number of PEs between 7 and 30. The low resource heterogeneity is simulated by resources having the number of PEs between 1 and 5. By analyzing the results in these figures, we can conclude that when

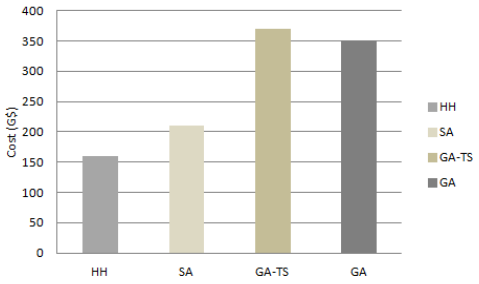

Figure 2. Comparison result for consistent and low machine heterogeneity

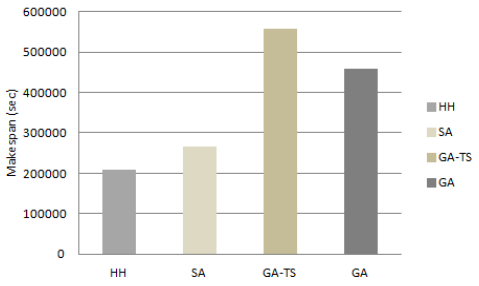

Figure 3. Comparison result for inconsistent and low machine heterogeneity

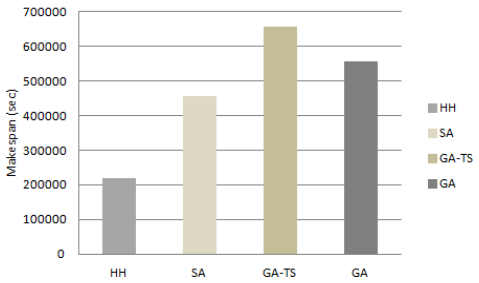

Figure 4. Comparison result for consistent and low machine heterogeneity

the resource heterogeneity is low, tabu search based hyper-heuristic with great deluge resource scheduling algorithm outperforms all the other approaches.

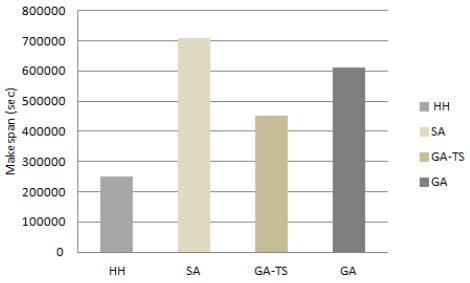

Figure 5. Comparison result for inconsistent and high machine heterogeneity

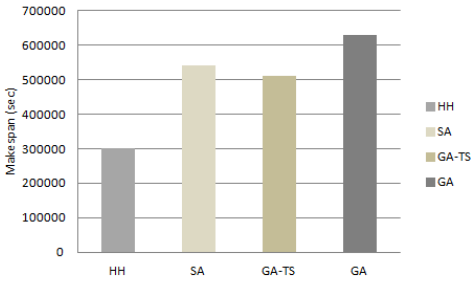

Figure 6. Comparison result for consistent and high machine heterogeneity

As discussed earlier, Gridlets which are sent to the Grid are supposed to be independent of each other. The characteristics of Gridlets are sent to the Grid to compare the makespan of different algorithms. The results show that in case of GA, SA and GA-TS algorithms, if we send the same number of applications/jobs to the Grid, makespan and cost increases whereas in the case of Tabu search based hyper-heuristic with great deluge resource scheduling algorithm both makespan and cost decreases.

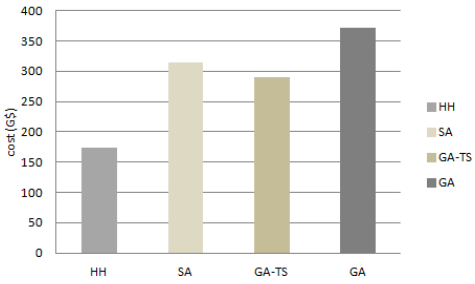

Figure 7. Comparison result for inconsistent and high machine heterogeneity

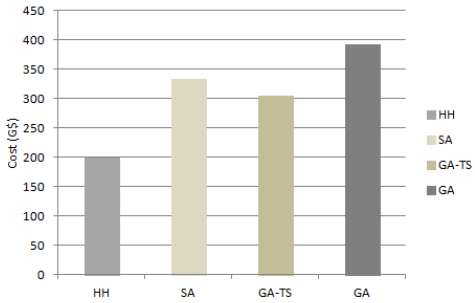

Figure 8. Comparison result for consistent and high machine heterogeneity

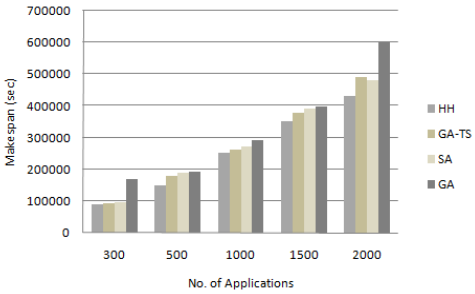

Figure 9. Effect of the number of application on the makespan

Testcase 2: Effect of the Number of Jobs

We have also performed experiments to determine the effect of increasing the number of applications on cost and makespan. We have around hundred-node simulated Grid with two thousand jobs being sent to the Grid. From the experimental results shown in Figure 9, we can conclude that the time taken to execute an application reduces by using Tabu search based hyper-heuristic with great deluge resource scheduling algorithm. Figure 10 shows that cost per application increases as the number of submitted application increases. The existing algorithm based application's execution resulted in a schedule which is expensive in comparison to Tabu search based hyper-heuristic with great deluge resource scheduling algorithm as the number of applications increases.

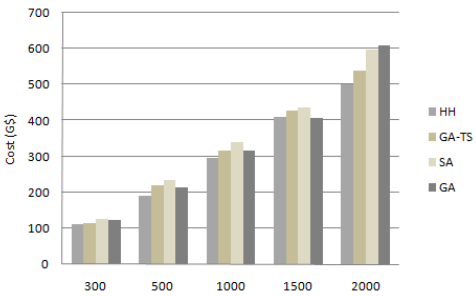

Figure 10. Effect of the number of application on the cost

From the above results, we observed that application execution using the Tabu search based hyper-heuristic with great deluge resource scheduling algorithm provides the following advantages: The makespan of the proposed algorithm is lower than the GA,SA and GA-TS. The time variation in execution of applications is about 5-10 %, compared to the existing algorithm of 50- 60% using the same set of applications. This time variation is quite significant. It also maintains cost for user's application execution.

V. Conclusions and Future Works

In this paper, we proposed a novel hyper-heuristic based scheduling algorithm for scheduling of jobs in Grid environment so as to minimize the cost and time by minimizing the makespan. Empirical results show that Tabu search based hyper-heuristic with great deluge resource scheduling algorithm outperforms in comparison to hybrid heuristics in all cases. The proposed algorithm not only minimizes the time and cost but it also minimizes the makespan and cost. In future, we would like to incorporate trust of node and reliability of the node /resources at the time scheduling of resources.

Acknowledgments

This work was supported in the framework of the IT4 Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 by operational programme Research and Development for Innovations funded by the Structural Funds of the European Union and state budget of the Czech Republic, EU.

References

[1] Foster, I. and Kesselman, C., "The Grid: Blueprint for a Future Computing Infrastructure", Morgan Kaufmann Publishers, USA, 2004.

[2] Khateeb A. A., Abdullah R. and Rashid A. N. "Job type approach for deciding job scheduling in Grid computing systems", journal of computer science, pp:745-750, 5(10),2009.

[3] Burke,E.K., Hyde,M., Kendall, G., Ochoa, G., Ozcan,E. and Qu,R., "Hyper-heuristics: A survey of the State of the Art",Technical report, University of Nottingham, 2009.

[4] Bhanu, S.M.S and Gopalan,N.P., "A Hyper-Heuristic Approach for Efcient Resource Scheduling in Grid", Int. J. of Computers, Communications Control, Vol. 3, No.3, pp. 249-258, 2008.

[5] Abraham,A., Buyya,R. and Nath,B., "Nature's Heuristics for Scheduling Jobs on Computational Grids". The 8th IEEE Conference on Advanced Computing and Communications, Cochin, India, 2000.

[6] Gonzalez, J.A., Serna, M. and Xhafa,F., "A Hyper-heuristic for schedulingin dependent jobs in Computational Grids", International conference on software and data technologies,ICSOFT(2007).

[7] Braun, T.D., Siegel, H.J., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", Journal of Parallel and Distributed Computing, Vol. 61, No. 6, pp.810837, 2001.

[8] Glover,F. and Laguna,M., "Tabu Search", Kluwer,Boston,1997.

[9] Dueck, G. "New optimisation heuristics for the great deluge algorithm and the record-to-record travel", Journal of Computational Physics, Vol. 104, pp. 86-92, 1993.

[10] Cowling, P., Kendall, G. and Han,L., "An Investigation of a Hyperheuristic Genetic Algorithm Applied to a Trainer Scheduling Problem", In Proceedings of the IEEE Congress on Evolutionary Computation, pp.1185-1190, 2002.

[11] Rajni, Chana, I., "Formal QoS Policy based Grid Resource Provisioning Framework", Journal of Grid Computing, 10(2) (2012) 249-264, 2012.

[12] Buyya R. and Murshed,M.: "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing", Concurrency and Computation: Practice and Experience (CCPE), Vol 14, pp.1175-1220, ISSN: 1532-0626, Wiley Press, New York, USA, November - December (2002).

[13] Burke, E.K., Kendall,G., Landa Silva, J.D., O'Brien,R. and Soubeiga,E., "An Ant Algorithm Hyperheuristic for the Project Presentation Scheduling Problem", Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC), Edinburgh, UK, 2nd-5th September 2005, pp 2263-2270.

[14] Zhurong Zhou, Wei Deng, Linrui Lu, "A Fuzzy Reputation Based Ant Algorithm for Grid Scheduling," cso, vol. 1, pp.102-104, 2009 International Joint Conference on Computational Sciences and Optimization, 2009

[15] Chen, J., "Economic Grid Resource Scheduling Based on Utility Optimization",pp:522-525, IITSI 2010

[16] X. He, X. Sun and G. Laszewski, A QoS Guided Min-Min Heuristic for Grid Task Scheduling, in J. of Computer Science and Technology, SpecialIssue on Grid Computing, Vol.18, No.4,pp.442–451, July 2003.

[17] Carretero, J. and Xhafa,F., "Use of Genetic Algorithms for Scheduling Jobs in Large scale Grid applications", Okio Technologies IR Ekoonominis Vvstymas, Technological and Economic development of Economy, Vol 12, No.1 pp 11-17, 2006.

[18] Carretero, J. and Xhafa,F. and abraham,A. "Genetic Algorithm based Schedulers for Grid Computing Systems", International Journal of Innovative Computing, Information and Control, Vol 3, No.6, 2007.

[19] U. Lublin, D. Feitelson, The workload on parallel supercomputers: Modeling the characteristics of rigid jobs, Journal of Parallel and Distributed Computing 63(11) (2003) 1105-1122.