Online Grid Scheduling Using Ant Algorithm

Kousalya K and Balasubramanie P

Abstract—**Grid computing is the next generation of distributed heterogeneous systems. It provides the highest productivity, utilizing the existing infrastructure. One of the most challenging issues in Grid computing is the efficiency of job scheduling in the distributed environment. Job scheduling in the grid environment is an NP complete problem. Heuristic approach is one of the best ways to solve the NP complete problem. The Scheduler considers the characteristics of resources such as geographic distribution, heterogeneity, different usage policies, non-uniform performance, varying loads and availability. The scheduler finds the suitable resource for each job. It leads to the effective utilization of grid resources. This paper proposes an adaptive scheduling algorithm, called Online Ant (OANT). The OANT uses the dynamic information of resources and jobs. The OANT algorithm uses the ant colony optimization heuristic technique. In this paper, performance of OANT is compared with other existing methods. It is found that the OANT algorithm can effectively and efficiently allocate jobs to proper resources**

Keywords—**Grid Computing, Scheduling, Ant Colony Optimization, Job Scheduling. Computational grid, Scheduling Algorithm**

I. INTRODUCTION

The grid computing technologies provide sharing of heterogeneous resources that are in geographically dispersed locations. So this type of technology is loosely coupled and has high computational capabilities and platform independent services across the entire internet. Because of these high computing capabilities, the computational grid easily solves the problems in science, engineering and commerce [1].To access the existing resources in grid computing, the users need a control and faultless access. It provides secure resource sharing in the distributed heterogeneous environment.So the researchers try to develop an intelligent computational grid environment instead of developing applications [2]. The Grid structure is changing almost all the time: some resources fail, some new resources enroll into the Grid and some resources resume working.

Grid Scheduling and management are the important issues in Grid computing. The scheduler must consider the dynamism of the Grid. The dynamics exists in both the networks and computational resources. In network dynamism, many parties can share the same network. So the network cannot provide guaranteed bandwidth all the time. In Computational resource dynamism, the availability and capability of computational resources will also change depending upon time. It means new resources may join or some of the resources may be unavailable due to some reasons. Similarly the resources may be shared by many users. Therefore capability is also changed due to time. Whenever the scheduler starts scheduling, it automatically detects and adds new resources and removes the unavailable resources in the later scheduling. After scheduling, if any unexpected failure occurs, rescheduling must be used to guarantee the reliability of Grid System

The scheduler in the grid collects the resource state information before the scheduling starts. The grid scheduling [3] is an optimal assignment of the set of tasks to the processing elements (resources) that are currently available in the grid environment and also improve the throughput of the entire system, but the resources are in multiple administrative domains. Good grid scheduling algorithm should be distributable, scalable, and fault tolerant. In general, the tasks are divided into two groups: Independent and dependent. The dependent tasks consider the results of predecessors. To evaluate the grid scheduling algorithms, makespan is one of the most important performance metrics. Makespan is the total time required to complete the metatask. The metatask is defined as the collection of independent tasks with no inter-task dependencies. Min-Min, Max-Min, Sufferage and XSuferage are the commonly used algorithms in the grid computing environment.

Grid scheduling is an NP-Complete problem. Heuristic optimization techniques are the best approach to solve NP-complete problems. The four basic heuristic methods for grid scheduling are Genetic Algorithm (GA) [4], Simulated Annealing (SA) [5], Ant Colony Optimization (ACO) and Tabu search (TS) [6]. The main focus of this paper is to develop a high throughput scheduling algorithm based on ACO.

The rest of the paper is organized as follows: Related work is discussed in section 2, in section 3 the ant colony system and proposed Online ant (OANT) are discussed. Analysis of the proposed algorithm (OANT) is presented in section 4. Conclusion and discussion are given in section 5.If your paper is intended for a *conference,* please contact your conference editor concerning acceptable word processor formats for your particular conference.

II. RELATED WORKS

In the current grid environment, a lot of scheduling algorithms [7, 8] are designed by the research developer. The algorithm produces sub optimal solutions because grid scheduling is an NP complete problem [9]. The existing heuristic scheduling is divided into two categories. One is dynamic scheduling and the other is static scheduling. In

Manuscript received February, 2009

K. Kousalya is with the Kongu Engineering College, Perundurai, India Phone 04294 226560

Dr P. Balasubramanie is with the Kongu Engineering College, Perundurai, India

dynamic scheduling, a task is scheduled to a resource whenever the job reaches the scheduler.

A. Dynamic Algorithms

Minimum execution time (MET)[6] is a dynamic scheduling algorithm. It assigns the task to the resource having the least amount of execution time. The algorithm does not consider whether the resource is currently available or not. So it serves load imbalance across the resources. This is one of the heuristic methods that are implemented in SmartNet [10]. Minimum Completion Time (MCT) [6] is also a dynamic scheduling algorithm. If the algorithm receives a job from the user, it immediately calculates the completion time of that job to all the machines that are currently available in the pool. Then the algorithm assigns that job to the resource having the earliest completion time. This algorithm may allocate a job to the resource that does not have minimum execution time. This heuristic method is also implemented in SmartNet [10].

B. Static Algorithms

In this method, tasks are collected as a set. These sets are mapped at prescheduled times called mapping events.

Min-Min is one of the popular algorithms. The Min-Min algorithm [6] has two phases. In the first phase, it calculates the minimum expected completion time for each task with respect to all machines in the set. In the second phase, it selects the task having the overall minimum expected completion time and assigns it to the corresponding resource. Then the currently scheduled task is removed from the set and repeats the above two phases until all the tasks in the set are completely scheduled [7]. So this method automatically minimizes the makespan and balances the load to an extent. At times, too many jobs are assigned to a single grid node and this will lead to system overloading and the response time of the job is not assured. This is the main disadvantage of Min-min method. Max-min [6] is very similar to Min-Min. The first phase of Max-Min and Min-Min are same but in the second phase the Max-Min selects the task that has the overall maximum expected completion time and assigns it to the corresponding resource. Min-min or Max-min will produce better results depending upon the expected execution time of unassigned tasks in the set. If minimum number of large tasks and too many short tasks are in the set, then the Max-min will produce better makespan, resource utilization rate and load balancing.

The QoS Guided Min-Min method considers QoS, which will affect the effectiveness of the grid [11]. The bandwidth and network are the two QoS constraints to basic Min-Min heuristics. This algorithm first assigns high QoS tasks then the low QoS tasks. It will produce worst result if all the tasks are high QoS or low QoS. The Segmented Min-Min heuristic algorithm has three steps. In the first step, the tasks are ordered by their expected completion time. Then in the second step, segments are in the ordered sequence. Finally apply Min-Min to those segments. This algorithm produces better results than Min-Min when the expected execution time is dramatically different.

In this paper [12], the grid simulation architecture using ACO is proposed. The response time and average utilization of resources are used as the evaluation indices. In this paper [13], the algorithm could improve the performance by increasing the job finishing ratio.

Characteristics of Ant Algorithm

Dorigo M. introduced the Ant algorithm in 1996, which is a new heuristic, predictive scheduling algorithm. It is based on the real ants. When an ant looks for food, ant deposits some amount of pheromone on the path, thus making a trail of this substance. If an ant tries to move from one place to another then it encounters a previously laid trail. The ant can detect the pheromone trail and decide with high probability to follow it. This ant also reinforces the trail with its own pheromone. When more ants are following the trail, then the pheromone on shorter path will be increased quickly. The quantity of pheromone on every path will affect the possibility of other ants to select that path. At last all the ants will choose the shortest path. In this paper [12], the experimental results show that the ant algorithm has produced an optimum solution. The ACO algorithm has been used to solve many NP problems, such as TSP, assignment problem, job-shop scheduling and graph coloring successfully. So the ant algorithm is suitable to be used in Grid computing task scheduling. In the grid environment, the algorithm can carry out a new task scheduling by experience, depending on the result in the previous task scheduling. In the grid computing environment, this type of scheduling is very much helpful. So, the ant algorithm for task scheduling in Grid Computing is proposed in this paper.

III. Problem Description

This paper uses two types of schedulers. One is local scheduler and another one is grid scheduler (OANT). The local scheduler uses all the system information and the local scheduling of resources. The OANT is placed in all the nodes that are involved in the grid scheduling. The node consists of one or more local schedulers and one or more resources. OANT in one node interacts and coordinates to their neighbor grid schedulers and their local scheduler. One of the major differences between OANT and local scheduler is that the OANT does not own the resources directly. OANT has the details of jobs which have to enquire the neighboring OANT.

In this method, each job is act like an Ant. After a job is successfully allocated to a particular node, the Ant will deposit a pheromone on the path it traveled. The pheromone is nothing but its numeric information about current ANT's performance. Each job maintains a separate list from the starting node to the destination node. The destination node is having the enough resource to execute the job. All the OANTs also maintain the pheromone information. Using this information only the remaining job selects a successful path from the set of available paths. There is no centralized control on OANTs.

In this method, a set of jobs can be scheduled parallely and asynchronously. When a job wants to use the grid, the job is submitted to one of the OANTs. The current OANT decides whether the current job is allocated to the local scheduler or moved to the neighboring OANT. If more than one neighboring OANTs are available, select the best OANT.

These are all done using the job's requirement, the set of OANTs that are already visited, the neighboring OANT's functionality and the pheromone information of current OANT. When the job is successfully allocated to one of the Local Schedulers, the job evaluates the cost of the path and deposits the pheromone value to all the OANTs that participate on the path. During deposition in OANTs, some amount of previous pheromone value must be evaporated.

In Grid computing, the resource can be added or removed, or the computational time may be changed from time to time. For that reason the pheromone evaporation must be considered over a period of time. Only then, the Ant (job) forgets the past and considers the current situation. The pheromone evaporation is used to prevent all the jobs being allotted to the same resource. This is the main advantage of pheromone evaporation. The main aim of grid scheduling is to reduce the makespan. Makespan is the total time needed to complete a group of jobs from the beginning of the first job to the completion of the last job.

In this paper, we focus on the decision about how to allocate a set of jobs to the resources optimally among the Super Schedulers. Problems like how to deal with the situation when a job does not find a solution or how the Super Schedulers communicate with each other are not discussed in the paper. Researches on those problems are being done on their way.

IV. Implementation

In this paper, the jobs are moved from one OANT to another OANT until the job finds a best local scheduler. The maximum numbers of OANTS in the grid have their own job queue. The users who want to execute their job in grid environment submit their jobs in the nearest OANT queue. All the OANTs collect their neighboring local schedulers load and the bandwidth available between the OANTS. The current OANT calculates the probability value of current OANT and their neighboring OANTs using the following probabilistic equation (1).

$$= \frac{T_{ij} \cdot \eta_{ij}}{\Sigma T_{ij} \cdot \eta_{ij}} \quad (1)$$

where,

Current OANT

Current OANT's neighboring OANTS

In the equation (1) the η_{ij} is the heuristic information.

$$\eta_{ij} = \frac{BWj}{L_j * ET_{ij}} \quad (2)$$

where,

BWj - Bandwidth of current and neighboring OANTS

Lj - Current Load of current and neighboring OANTS

ETij - Expected Execution Time of current job in current and neighboring OANTS

The heuristic is an algorithm that gives up finding the optimal solution for the improvement of run time. In the equation (1), Tij is the pheromone trail update value. The initial pheromone value (T0) is

$$T_o = \frac{1}{L_j} \quad (3)$$

After the job finds their local scheduler, the scheduled job deposits some pheromone value on pheromone trail update variable. During updation, some of the past pheromone values are automatically evaporated using the formula. The updation must be done only in the OANTS which are visited by this job.

$$T_{ij} = T_{ij} + (1-\rho)\, \triangle T_{ij} \quad (4)$$

where

$(1-\rho)$ – the evaporation value between 0 to 1

$\triangle T_{ij}$ is the additional pheromone value and it is different for different ACO algorithms

$$\triangle T_{ij} = \frac{1}{NEi - 1} \quad (5)$$

The proposed algorithm starts only if the OANTs queue has some set of jobs. The initialization part of the algorithm is as follows. The algorithm collects the details about available OANTs and their local schedulers and neighboring local schedulers load, bandwidth of available network, expected execution time of the queued jobs. The variable free is a one dimensional matrix of size m (no. of OANTs) and the initial value is current load of their local schedulers.

Algorithm 1 Algorithmic frame for a OANT Algorithm

Begin

Calculate the initial pheromone value ($\mathbf{T_o}$)

$\mathbf{T_o = 1/L_j}$

For each **OANTS** queue not empty do

For each **job** in the **OANT** do

Marked current OANT is visited by job_j

While **true**

If job visited to all the **OANTs**

Remove the job_j from the grid

Exit from the while loop

Endif

Calculate the heuristic information ($\boldsymbol{\eta_{ij}}$)

$\boldsymbol{\eta_{ij} = BW_j/(L_j * ET_{ij})}$

Calculates the probability value of **Current OANT** and **Neighboring OANT**

$$\mathbf{P_{ij}} = \frac{T_{ij} \cdot \eta_{ij}}{\Sigma T_{ij} \cdot \eta_{ij}}$$

If **current OANT's** local scheduler has the highest probability

Add ($\mathbf{job_i}$, $\mathbf{resource_j}$, **free[j], free[j]+**$\mathbf{ET_{ij}}$) to the output list.

Calculate $\triangle T_{ij} = 1/ NE_i - 1$
Update the pheromone trail value using
$T_{ij} = T_{ij} + (1- \rho) \triangle T_{ij}$
Else If one of the neighbored OANT has the highest priority
Marked current OANT is visited by job_j
Move the current job to that neighboring node
Endif
Endwhile
Endfor
Endfor
From the output list find out the makspan
End

V. Experimental Results

In online methods, minimum execution time and minimum competition time [6] are the currently used technologies in grid scheduling. The above methods simply allocate the resources without knowing the current workload, CPU load and the bandwidth of the networks. In this case, some jobs cannot finish their execution. It may have to resubmit and reschedule the same job again and again.

In the paper [14], communication delay is the only parameter to find the heuristic information. The paper [14] does not consider the neighboring local scheduler's workload and current CPU speed. Here the results are compared with the Super scheduler [14].

To simulate the various heterogeneous problems, different types of ET matrix using benchmark simulation model [6] are defined.

The ET matrix considers three factors: task heterogeneity, machine heterogeneity and consistency. The task heterogeneity depends upon the various execution times of the jobs. The two possible values are defined high and low. Similarly the machine heterogeneity depends on the running time of a particular job across all the processors and again has two values: high and low. In real scheduling, three different ET consistencies are possible.

They are consistent, inconsistent, and semi consistent. The instances of bench mark problems are classified into twelve different types of ET matrices. Each consists of 100 instances. The instances depend upon the above three factors as task heterogeneity, machine heterogeneity and consistency. Instances are labeled as u_x_yyzz.k where

u - is a uniform distribution, used to generate the matrix.

x – is a type of consistency

c- consistent

s-semi consistent

i-inconsistent

An ET matrix is said to be consistent if a resource Ri executes a task Ti faster than the resource R_k, and R_i executes all other jobs faster than R_k. An ET matrix is said to be in-consistent if a resource R_i executes some jobs faster than R_j and some slower. A semi consistent ETC matrix is an inconsistent matrix which has a sub matrix of a predefined size.

yy- is used to indicate the heterogeneity of the jobs(hi – high, lo-low)

zz-is used to indicate the heterogeneity of the resources (hi-high, lo-low)

All the instances consist of 512 jobs and 16 machines. For each method the makespan is computed. It allows a fair comparison of the presented methods

VI. Performance Evaluation

The makespan of the OANT and Super scheduler's are stacked in area chart and is given in Fig.1 and the corresponding table is given in table 1.

	OANT	SSCHEDULE
CHTHM	150714824.73	179539649.04
CHTLM	1584786.75	1854560.26
CLTHM	4757394.12	6088846.44
CLTLM	45622.43	62785.82
ICHTHM	20927117.36	155220213.87
ICHTLM	252364.70	1489698.69
ICLTHM	685712.55	4951086.54
ICLTLM	7823.16	54146.63
PCHTHM	56484374.76	171144747.28
PCHTLM	512879.33	1721587.59
PCLTHM	1706585.50	5590235.08
PCLTLM	18550.81	58555.03

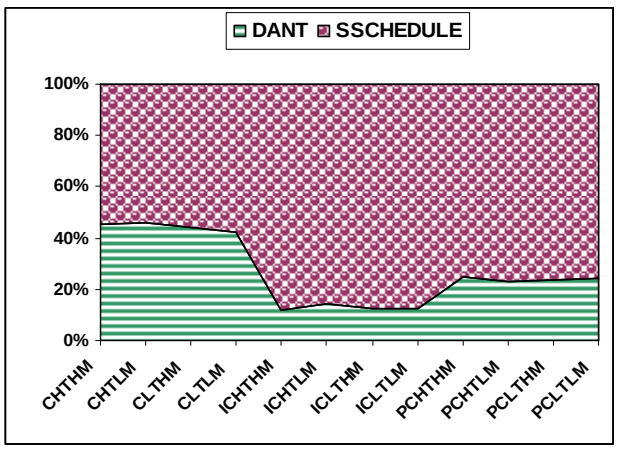

Fig 1. Graphical representation of makespan values (arbitrary time units)

The above Figure shows that the relative size of OANT algorithm is very low as compared with Super Scheduling. The OANT algorithm produces excellent results for inconsistent matrices. It also produces the best result for the semi consistency type problems

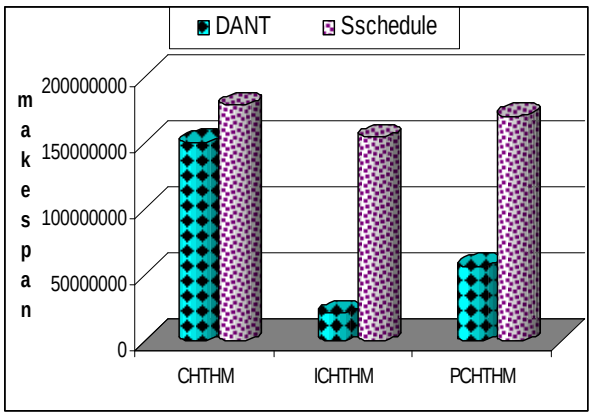

Fig 2. Graphical representation of makespan values of High Task High Machine (arbitrary time units)

Because of task and job's heterogeneity, this paper has four different sets. They are High Task High Machine, Low Task High Machine, High Task Low Machine, and Low Task Low Machine. The comparisons between OANT and super scheduler [14] is shown in Figure 2, 3, 4, 5 of High Task High Machine, Low Task High Machine, High Task Low Machine, Low Task Low Machine respectively. The hardware/software configuration used is irrelevant because the execution times are given in their time complexity.

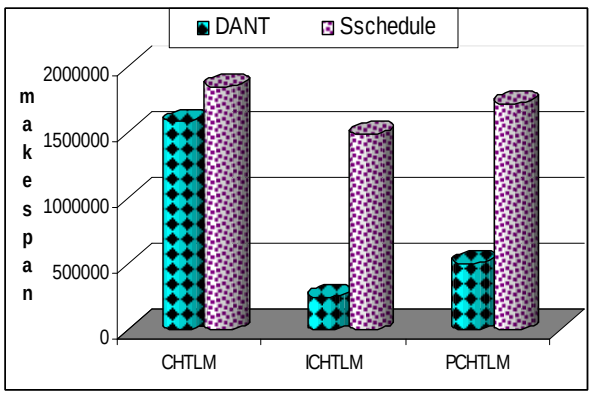

Fig. 3. Graphical representation of makespan values of High Task Low Machine (arbitrary time units)

TABLE 2: PERCENTAGE DECREASE IN MAKESPAN VALUE BY AC IN COMPARISON WITH AWEOC AND AE (VALUES IN %)

Problem Type	Decreases of Makespan %
CHTHM	16.05%
CHTLM	14.55%
CLTHM	21.87%
CLTLM	27.34%
ICHTHM	86.52%
ICHTLM	83.06%
ICLTHM	86.15%
ICLTLM	85.55%
PCHTHM	67.00%
PCHTLM	70.21%
PCLTHM	69.47%
PCLTLM	68.32%

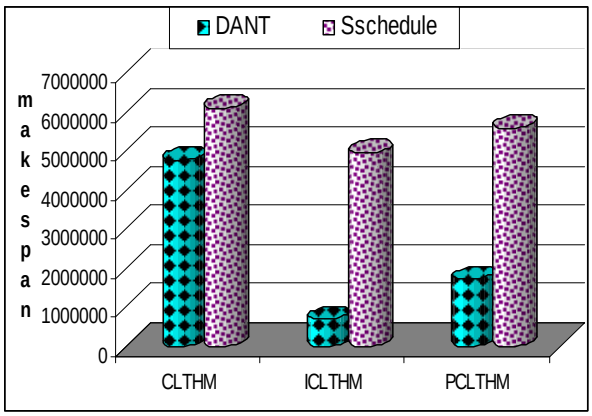

Fig. 4. Graphical representation of makespan values of Low Task High Machine (arbitrary time units)

As compared to OANT and Super Scheduler there is, a major difference in the reported makespan values, yields better results for 12 out of 12 considered instances (Table 2).

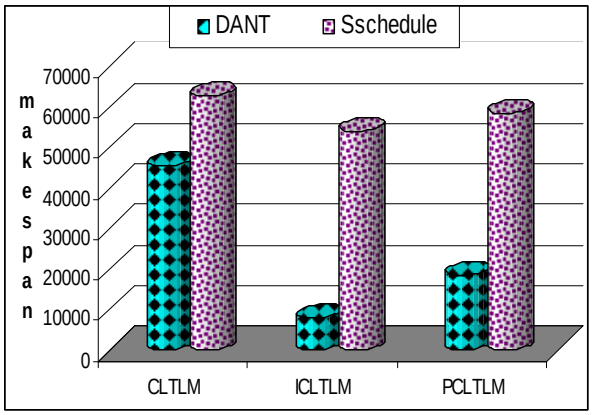

Fig 5. Graphical representation of makespan values of Low Task Low Machine

The heuristic techniques seen above, the OANT Ant algorithm performs above sixty percentage better than the Super scheduling [14] ant algorithm in all possible cases on an average. Thus, addition of ETij in the calculation of probability matrix, that is inclusion of completion time of the ith job by the jth machine (predicted), has shown a positive result in performance improvement. This improvement is in terms of decrease in makespan time.

VII. CONCLUSION AND FUTURE WORK

Using the Ant Algorithm, this paper tries to allocate all the submitted jobs to the available resources successfully. In this method, each and every job knows the current grid environment and takes decision using the environment information. The jobs are also responsible for changing the gird environment, because each job tries to deposit some amount of pheromone on their successful path. The job uses the previous job's pheromone information during its allocation. So, this ACO method can allocate the jobs to the resources effectively in the dynamic environment.

These methods consider bandwidth and CPU Load. But Cost and memory space are not considered. In future research, bandwidth, CPU Load, Cost and memory spaces can be considered. Try to modify the heuristic information in the

probably matrix to get further minimized makespan. If the job does not match all the resources in that path then simply remove that job from the grid. In future, put that job to another queue and find solution for that job. In this paper the job may be submitted to the nearest queue. But in future, advanced reservation, the job reserved the required resource in advance and uses that resource in the allocated time period.

REFERENCES

[1] Foster. I Kesselman C.Tuecke S, The anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer, April 2001, 15(3):pp 200-22

[2] Xin Bai, Han Yu, GuoQiang Wang, Youngchang Ji, Gabriela.M. Marinescu, . Coordination in Intelligent Grid Environments. Proceedings of the IEEE, VOL 93, No. 3, 2005

[3] [3] Luo Hong, Mu De-jun, Deng Zhi-qum and Wang Xiao-dong, A Review of Job scheduling for Grid Computing, Application Research of Computers, 2005, 22(5), pp. 16-19

[4] Aggarwal, M.; Kent, R.D.; Ngom, A, " Genetic algorithm based scheduler for computational grids" International Symposium on High Performance Computing Systems and Applications, 2005.. Volume15, No.18 pp: 209 – 215

[5] Fidanova.S, " Simulated Annealing for Grid Scheduling Problem", International IEEE Symposium on Modern Computing, 2006.

[6] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen and R. Freund, 2001, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing, Vol.61, No.6, pp. 810- 837, 2001.

[7] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems, " 8th IEEE Heterogeneous Computing Workshop (HCW '99), pp. 30-44, San Juan, Puerto Rico, April 1999.

[8] F. Dong and S. G. Akl, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems, " Technical Report of the Open Issues in Grid Scheduling Workshop, School of Computing, University Kingston, Ontario, January 2006.

[9] D. Fernández-Baca, "Allocating modules to processors in a distributed system, " IEEE Transactions on Software Engineering, pp. 1427-1436, November 1989.

[10] R. F. Freund, M. Gherrity, S. Ambrosius, M. Camp-bell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel, "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet, " 7th IEEE Heterogeneous Computing Workshop (HCW'98), pp. 184–199, March 1998.

[11] X. He, X. Sun and G. Laszewski, "A QoS Guided Min-Min Heuristic for Grid Task Scheduling, " Journal of Computer Science and Technology, Special Issue on Grid Computing, pp. 349-363, Cancun, Mexico, May 2000.

[12] Zhihong XU, Xiangdan HOU, Jizhou SUN, " Ant- Algorithm-Based Task scheduling in Grid Computing", Montreal, In Proceeding of the IEEE Conference on Electrical and Computer Engineering, pp. 1107-1110, 2003.

[13] H. Yan, X. Shen, X. Li and M. Wu, "An Improved Ant Algorithm for Job Scheduling in Grid Computing", In Proceedings of the IEEE International Conference on Machine Learning and Cybernetics, pp. 2957-2961, 2005.

[14] Li Liu, Yi Yang, Lian Li and Wanbin Shi, " Using Ant Optimization for super scheduling in Computational Grid, IEEE proceedings of the 2006 IEEE Asia-pasific Conference on Services Computing (APSCC' 06)

Kousalya.K received the B.E. and M.E. degrees in Computer Science and Engineering from Bharathiar University, Coimabatore, India, in 1993 and 2001, respectively. She is currently doing her PhD degree in Anna University, Chennai, India. Currently she is an Assistant Professor in the department of Computer Science and Engineering, Perundurai, Tamilnadu. Her areas of interest are Grid Computing, Compiler Design and Theory of Computation. She has published papers in National, International Conferences and in International Journal.

Dr.P.Balasubramanie has obtained his PhD degree in theoretical computer science in the year 1996 from Anna University, Chennai. He was awarded junior research fellow by CSIR in the year 1990. Currently he is a professor in the Department of Computer Science and Engineering, Kongu Engineering College, Perundurai, Tamilnadu. He has published more than 50 research articles in International/National Journals. He has also authored six books. He has guided 3 PhD scholars and guiding 15 research scholars. His areas of interest include theoretical computer science, data mining, image processing and optimization Techniques.