An Enhanced Ant Algorithm for Grid Scheduling Problem

Kousalya.K and Balasubramanie.P,

Anna University, Kongu Engineering College, Tamilnadu, India

Summary

Grid computing is a form of distributed computing that coordinates and shares computation, application, data storage, or network resources across dynamic and geographically dispersed organizations. One primary issue associated with the efficient utilization of heterogeneous resources in a grid is grid scheduling. Grid Scheduling is a critical design issue of grid computing. It is a challenge because the capability and availability of resources vary dynamically. The complexity of scheduling problem increases with the size of the grid and becomes difficult to solve effectively. Hence a new area of research is developed to design optimal methods. It focuses on new heuristic techniques that provide an optimal or near optimal solution for large grids. In this paper, Ant Colony Optimization based grid scheduling algorithm for grid computing is proposed. The proposed scheduler allocates an application to a host from a pool of available hosts and applications by selecting the best match. In the evaluation study a number of intensive experiments with various simulation settings have been conducted. Based on the experimental results, the proposed algorithm confidently demonstrates its practicability and competitiveness with three previously proposed algorithms.

Key words:
Grid computing, Grid Scheduling, Ant System, Heuristics

1. Introduction

Grid Computing allows the integration and sharing of computers and computing resources, such as software, data and peripherals, in corporate networks. Grid-enabled networks stimulate cooperation among users and organizations, create dynamic and multi-institutional environment, provide and use the resources to achieve common or individual goals [1]. The usage of grids to solve CPU-intensive problems potentially benefits the entire society. With further development of grid technology, it is very likely that corporations, universities and public institutions will exploit grids to enhance their computing infrastructure. In recent years there has been a large increase in grid technologies research, which has produced some reference grid implementations. A vast number of researchers have been putting in a lot of effort to facilitate building and efficient utilization of grids. A significant grid is Globus toolkit [2]. The Security Infrastructure (GSI) in the Globus toolkit addresses and effectively deals with the security issue. However, there are still a considerable number of difficulties that should be over come for efficiently scheduling jobs in grids.

A Grid scheduler, often called resource broker, acts as an interface between the user and distributed resources and hides the complexities of Grid computing [3,4]. It performs resource discovery, negotiates for access costs using trading services, maps jobs to resources (scheduling), stages the application and data for processing (deployment), starts job execution, and finally gathers the results. It is also responsible for monitoring and tracking the progress of application execution along with adapting to the changes in the runtime environment of the Grid, variation in resource share availability, and failures. Essentially, the Grid broker does application scheduling on distributed Grid resources on which it does not have full control—the local scheduler has its own policies and performs actual allocation of resource(s) to the user job(s).

The previous work in scheduling on distributed systems such as clusters and supercomputers has focused on extracting the maximum throughput from the entire system [5,6]. Grid scheduling concentrates on improving response times in an environment containing autonomous resources whose availability dynamically varies with time. The Grid scheduler has to interact with the local schedulers managing computational resources and adapt its behavior to changing resource loads. Thus the scheduling is conducted from the perspective of the application or the user rather than that of the system.

Grid scheduling requires a series of challenging tasks. These include, searching for resources in the collection of geographically distributed heterogeneous computing systems and making scheduling decisions, taking into consideration quality of service. A grid scheduler differs from a scheduler for conventional computing systems in several respects. One of the primary differences is that the grid scheduler does not have full control over the grid. More specifically, the local resources are in general not controlled by the grid scheduler, but by the local scheduler. Another difference is that the grid scheduler cannot assume that it has a global view of the grid.

The demand for scheduling is to achieve high performance computing. It is very difficult to find an optimal resource allocation for specific job that minimize the schedule

length of jobs. The scheduling problem is a NP-hard problem [7] and it is not trivial.

The main goal is to schedule all the incoming applications to the available computational power. Meta heuristic approaches have shown their effectiveness for a wide variety of hard problems. These approaches produce best results in practice.

2. Literature Review

2 .1 Overview of Previous Algorithms

The resource scheduling in grid is a NP complete problem. Various algorithms have been designed to schedule the jobs in computational gird. The most commonly used algorithms are OLB, MET, MCT, Min-Min and Max-Min.

2.1.1 Opportunistic Load Balancing (OLB)

Without considering the job's execution time, it assigns a job to the earliest free machine. If more than one machine is free then it assigns the job in arbitrary order to the processor. This scheduler runs faster and assigns each job in the arbitrary order to the next available node. The advantage of this method is that it keeps almost all machines busy all possible time. Yet the solution is not optimal.

2.1.2 Minimum Execution Time (MET)

The first available machine is assigned a job with the smallest execution time. It neither considers the ready time nor the current load of the machine. Also, the availability of the resources at that instant of time is not taken into account. The resources in grid system have different computing power. Allocating all the smallest tasks to the same fastest resource redundantly creates an imbalance condition among machines. Hence this solution is static.

2.1.3 Minimum Completion Time (MCT)

It uses the ready time of the machine to calculate the job's completion time (ready time of the machine + execution time of the job). It calculates the completion time of current job in the earliest available machines. From the list, the job with smallest completion time is selected and is assigned to that machine. This means the assigned job may have a higher execution time than any other job. This algorithm calculates the completion time of current unfinished job in only one earliest available node. But the same job may be completed in lesser time in some other machine which is available at that time.

2.1.4 Min-Min

It starts with a set of unmapped tasks. The minimum completion time of each job in the unmapped set is calculated. This algorithm selects the task that has the overall minimum completion time and assigns it to the corresponding machine. Then the mapped task is removed from the unmapped set. The above process is repeated until all the tasks are mapped. When compared with MCT, Min-Min considers all the unmapped tasks during their mapping decision. The smaller makespan can be obtained when more tasks are assigned to machines that complete them the earliest and also execute them the fastest.

2.1.5 Max-Min

First it starts with a set of unmapped tasks. The minimum completion time of each job in the unmapped set is found. This algorithm selects the task that has the overall maximum completion time from the minimum completion time value and assigns it to the corresponding machine. The mapped task is removed from the unmapped set. The above process is repeated until all the tasks are mapped. On comparison with MCT, Max-Min considers all unmapped tasks during their mapping decision. The Max-Min may produce a balanced load across the machine. When compare to Max-Min Min-Min is the best one.

2.1.6 Drawbacks of the above stated algorithms

Though the above mentioned algorithms have various advantages, they also have some pitfalls. The drawback of Min-Min is that, too many jobs are assigned to a single grid node. This leads to overloading and the response time of the job is not assured. OLB does not assure load balance. In MCT calculation of minimum completion time for a job is longer. Other algorithms are very difficult to implement. In paper [4], the job moving from one machine to another machine is discussed. So the traffic in the grid system will be automatically increased. In paper [3] communication cost is considered.

The problem of scheduling a grid is complex. So a number of researchers research in this area. They are trying to find an optimal solution and harness the existing resources effectively. The main aim of scheduling is to improve the overall system performance. Min-Min, Max-min, fast greedy tabu search and ant system are some of the heuristic algorithms which create a static environment. They must predict the execution time and workload in

advance. In paper [8], they have proposed a simple grid simulation architecture using ACO. They have used response time and average utilization of resources as the evaluation index. In the paper [9] and [10], they have proposed ACO algorithms, which could improve the performance like job finishing ratio. In paper [11], the job is moved from one machine to another machine, so that the traffic in the grid system will be automatically increased. In paper [10], communication cost is considered and in paper [12], six different ant agents are used. To solve the grid scheduling problem, ACO is one of the best algorithms.

2.1.7 Ant Algorithm

The ant algorithm is also based upon heuristic approach. It is based on the behavior of real ants. Each ant deposes the chemical pheromone on its path when it searches for food from its nest. When each ant moves in a particular direction, the strength of chemical pheromone increases. With this, other ants could also trail along.

This inspired the discovery of ACO algorithm. This algorithm uses a colony of artificial ants that behave as co-operative agents in a mathematical space where they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. This approach which is population based has been successfully applied to many NP-hard optimization problems.

3. Problem Description

Grid computing is a dynamic environment and allocates the jobs to the resources effectively. The important challenge in grid scheduling is that, no one has the ability to control all the jobs completely. The other challenges are, the resources dynamic nature and the difference between the expected execution time and the actual time in algorithm. The main aim of the scheduler is to allocate the jobs to the available nodes. The best match must be allocated from the list of available jobs and the list of available resources. The selection is based on the prediction of the computing power of the resource [13].

The grid users expect to run their jobs efficiently. The efficiency depends upon two criteria; one is makespan and the other is flow time. These two criteria are very much important in the grid system. The makespan measures the throughput of the system and flow time measures its QOS [14,15].

The expected execution time ET is the expected time to complete the matrix. The element ETij of the ET matrix is defined as the amount of time taken to complete i^{th} job in the j^{th} resource. The jobs are owned by different users. Each job has to be completely preempted. All jobs are interdependent. Each and every resource has its own computing characteristics. All the resource may be dynamically added or removed from the grid. They use the expected time to compute (ET) the model [16]. Between ET value and actual time taken to complete a job there is a difference but calculate or assume that the values in ET matrix are the completion time for that job.

The ET matrix will have N x M entries, where N is the number of independent jobs to be scheduled and M is the number of resources which is currently available. Each job workload is measured by million of instructions and the capacity of each resource is measured by MIPS. The Ready time ($Ready_m$) indicates the time resource 'm' would have finished the previously assigned jobs. The completion time of i^{th} job on the j^{th} machine is

$$CT_{ij}=Ready_j+ET_{ij} \quad (1)$$

$Max_s(CT_{ij})$ is the makespan of the complete schedule. Makespan is used to measure the throughput of the grid system. The main objective of this algorithm is to minimize the makespan. The grid scheduling problem is a NP-complete problem. In general the existing heuristic mapping can be divided into two categories. One is on line mode and the other one is batch mode. In the on line mode, the scheduler is always in ready mode. Whenever a new job arrives to the scheduler, it is immediately allocated to one of the existing resources required by that job. Each job is considered only once for matching and scheduling.

In the batch mode, the jobs and resources are collected and mapped at prescheduled time. In this mode, it takes better decision because the scheduler knows the full details of the available jobs and resources. The proposed algorithm is also heuristic algorithm for batch mode.

The result of the algorithm will have four values (task, machine, starting time, expected completion time). The number of jobs available for scheduling is always greater than the available number of machines in the grid. The machine Mj's free time will be known using the function free (j). The starting time of job t_i on resource Mj is

$$Bi = free(j) + 1 \quad (2)$$

Then the new value of free(j) is the starting time plus ET_{ij}. In the algorithm, use the minimization function to find out the best resource

$$F = max(free(j)) \quad (3)$$

And use the following heuristic information

$$\eta_{ij} = \frac{1}{Free(j)} \quad (4)$$

Using the formula (4) find out the highest priority machine which is free earlier. Use three to four ants. Each ant starts from random resource and task (they select ET_{ij} randomly j^{th} resource and i^{th} job). All the ants are maintaining a separate list. Whenever they select next task and resource, they are added into the list. At each iteration the ants calculate the minimize function 'F_k (kth ant)' and Pheromone trail updates the value.

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

In this algorithm two set of tasks are maintained. One is set of scheduled tasks and the other is set of arrived and unscheduled tasks. The algorithm starts automatically, whenever the set of scheduled jobs become empty.

According to the paper [17], the first task to be performed and the machine in which it is performed is chosen randomly. Next, the task to be run and the machine in which it is to be run is computed by the following formula

$$Pij = \frac{T_{ij} \cdot \eta_{ij}}{\Sigma T_{ij} \cdot \eta_{ij}} \quad (6)$$

Where

- η_{ij} is the attractiveness of the move as computed by some heuristic information indicating a prior desirability of that move
- T_{ij} is the pheromone trail level of the move, indicating how profitable it has been in the past to make that particular move(it represents therefore a posterior indication of the desirability of that move)
- $P^k_{i,j}$ - is the probability to move from a state i to a state j is depending on the combination of above two values:

The above formula (6) has the disadvantage, that all the columns in the probability matrix has the same probability value. This decides the best resource, but the task is chosen to be the first non zero value of the column. In paper [18], they use one ant. To overcome this disadvantage a new algorithm is proposed. In this method, modify the probability matrix ($P^k_{i,j}$) and use several ants, and the number of ants used is less than or equal to the number of tasks. From all the possible scheduling lists find the one having minimum makespan and use that ant's scheduling list.

Here two kinds of ET matrices are formed, one of them consists of currently scheduled jobs and the other consists of jobs which have arrived but not scheduled. The scheduling Algorithm is executed periodically. At the time of execution, it finds the list of available resources (processors) in the grid environment, form the ET matrix and start scheduling.

3. 1 Scheduling Algorithm

The execution time matrix ETij of task t_i on machine $m_{j,}$ is defined as the amount of time taken by $m_{j,}$ to execute t_i

Given that m_j has no load t_i assigned. The expected completion time is (CT_{ij})

$$CT_{ij} = B_i + ET_{ij} \quad (7)$$

where

B_i = Beginning time of t_i on machine m_j

The function free[j] – return time when the machine Mj will be free.

$$free[j] = Bi+ETij+1 \quad (8)$$

Use the objective function Fk = max(free[j]) over the solution constructed by an ant k and added pheromone by an ant k

The result will be in the following format (task, machine, starting time, Ending time)

Step 1: Collect all necessary information about the jobs (n) and resources (m) of the system in matrix $ET_{mxn.}$

Step 2: Set all the initial value

ρ = 0.05(pheromone evaporation value)
T_0 = 0.01(initial pheromone deposit value)
Free[0.. m-1] = 0(one dimensional matrix of size m)
k = m(number of ants= no. of tasks)

Step 3: For each ant (to prepare the scheduling list) do the following **steps 4 and 5**

Step 4: Select the task (i) and resource (j) randomly.

Step 5: Repeat the following until all jobs are executed.

a. Calculate the heuristic information (η_{ij})

$$\eta_{ij} = \frac{1}{Free(j)} \quad (9)$$

If a machine is free earlier then the corresponding machine will be more desirable

b. Calculate current pheromone trail value

$$\triangle T_{ij} = \frac{1-\rho}{F_k} \quad (10)$$

where $Fk = \max(free(j))$;

c. Update the Pheromone Trail Matrix

$$T_{ij} = \rho T_{ij} + \triangle T_{ij} \quad (11)$$

d. Calculate the Probability matrix

(12)

$$Pij = \frac{T_{ij} \cdot \eta_{ij} (1/ET_{ij})}{\Sigma T_{ij} \cdot \eta_{ij} (1/ET} \quad \text{where}$$

- η_{ij} is the attractiveness of the move computed by some heuristic information indicating a prior desirability of that move.
- T_{ij} is the pheromone trail level of the move, indicating how profitable it has been in the past to make that particular move(it represents therefore a posterior indication of the desirability of that move)
- ET_{ij} - Execution Matrix.

e. Select the task with highest probability's of 'i' and 'j' as the next $task_i$ to be executed on the $resource_j$

Step 6: Find the best feasible solution using all the ants scheduling List

4 Computational Results

Here the results are compared with the various implementations of OLB, MET, MCT, and existing Ant algorithms. To simulate the various heterogeneous problems, different type of ET matrix using benchmark simulation model [16] are defined. The ET matrix considers three factors: task heterogeneity, machine heterogeneity and consistence. 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 the 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 task heterogeneity, machine heterogeneity and consistence. 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 execute a task Ti faster than the resource Rk, then Ri will execute all other jobs faster than Rk. An ET matrix is said to be in-consistent if a resource Ri may execute some jobs faster than Rj 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 (h-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.

The computation results are given in tables 1, 2,3,4,5,6,7,8 and 9. The results are obtained from immediate mode methods for makespan like MCT, MET and two heuristic methods (existing ant algorithm [17] and proposed ant algorithm), for a set of 12 instances of the bench mark [17]. From the bench mark problem, chose three groups of four instances having consistent, semi consistent and inconsistent ET matrices. The selected instances are having the different types of heterogeneity of jobs and heterogeneity of resources.

5 Performance Evaluations

The tables 1,3,5,7 show the comparison of proposed ant algorithm with OLB,MCT,MET and existing ant algorithm in High Task High Machine, Low Task High Machine, High Task Low Machine, Low Task Low

Machine respectively and the corresponding graphs are shown in the Figure 1,2,3,4. The percentage decreases in the makespan value by the proposed ant algorithm when compared to OLB, MCT, MET and Existing ant are listed in the tables 2 ,4, 6, 8 in High Task High Machine, Low Task High Machine, High Task Low Machine, Low Task Low Machine respectively.

The entire makespan of the computational results are given in table 9. In table 9, the name of the instance is in the first column, the best makespan obtained by OLB is in the second, MCT is in the third and MET is in the fourth column respectively. , The best makespan obtained by existing ACO is in the fifth and proposed ACO is in sixth column. Figure 5 shows the comparison of all the five algorithms' makespan. The proposed algorithm performs better than immediate mode methods like MCT, MET, OLB and also existing ant algorithm.

The proposed ant algorithm performs on an average of ten percentage better than other algorithms.

Table 1: Makespan values for Braun et al. benchmark (in arbitrary time units) by algorithms MCT, MET, Existing-Ant, Proposed Ant on HTHM when they are consistent, inconsistent (IC) and partially consistent (PC)

High Task High Machine			
	CONSISTENT	**IC**	**PC**
OLB	14376662.18	26102017.618	19464875.910
MCT	11422624.49	4413582.982	6693923.896
MET	47472299.43	4508506.791	25162058.136
Existing-Ant	13496496.72	5703005.083	8765320.713
Proposed Ant	12485079.88	3659080.427	7722087.294

Table 2 Percentage decrease in makespan value by Proposed Ant algorithm in comparison with other algorithms (values in percentages)

	High Task High Machine		
	CONSISTENT	**IC**	**PC**
OLB	13.16	85.98	60.33
MET	-9.30	17.10	-15.36
MCT	73.70	18.84	69.31
Existing Ant	7.49	35.84	11.90

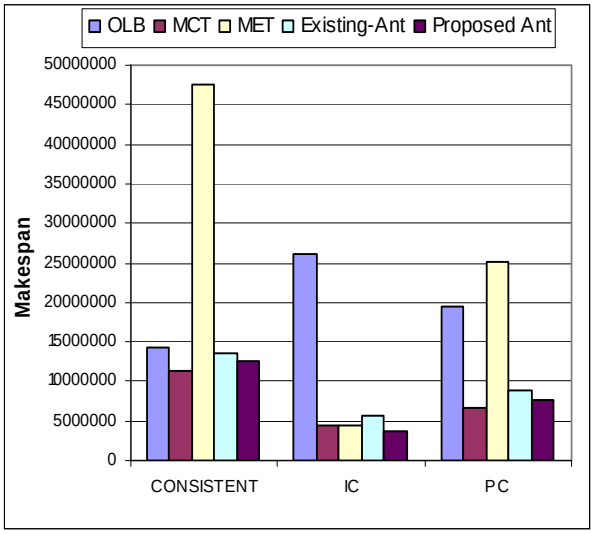


Fig. 1: Graphical Representation of Table 1

Table 3: Makespan values for Braun et al. benchmark (in arbitrary time units) by algorithms MCT, MET, Existing-Ant, Proposed Ant on LTHM when they are consistent, inconsistent (IC) and partially consistent (PC)

Low Task High Machine			
	CONSISTENT	**IC**	**PC**
OLB	477357.019	833605.654	603231.467
MCT	378303.624	143816.093	186151.286
MET	1453098.003	185694.594	674689.535
Existing-Ant	370797.065	169621.082	260822.210
Proposed Ant	319142.292	125062.742	207160.951

Table 4 : Percentage decrease in makespan value by Proposed Ant algorithm in comparison with other algorithms (values in percentages)

	Low Task High Machine		
	CONSISTENT	**IC**	**PC**
OLB	33.14	85.00	65.66
MET	15.64	13.04	-11.29
MCT	78.04	32.65	69.30
Existing Ant	13.93	26.27	20.57

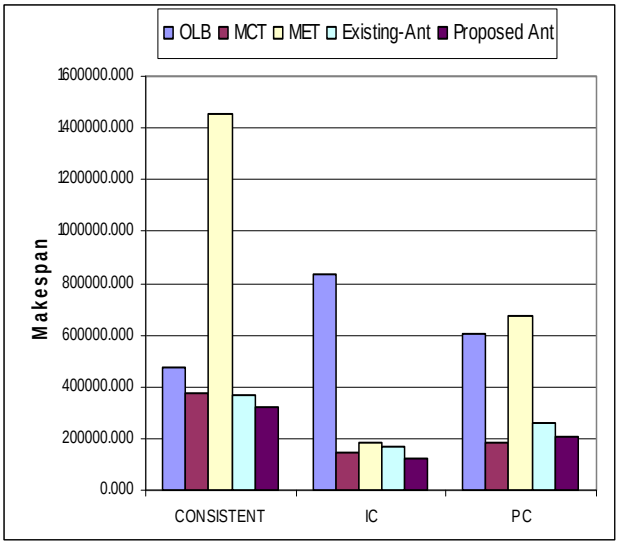


Fig. 2: Graphical representation of table 3.

Table 5: Makespan values for Braun et al. benchmark (in arbitrary time units) by algorithms MCT, MET, Existing-Ant, Proposed Ant on HTLM when they are consistent, inconsistent (IC) and partially consistent (PC)

High Task Low Machine			
	CONSISTENT	**IC**	**PC**
OLB	221051.823	272785.200	250362.113
MCT	185887.404	94855.913	126587.591
MET	1185092.968	96610.481	605363.772
Existing-Ant	117674.295	48855.353	82109.419
Proposed Ant	98017.149	32256.535	55977.881

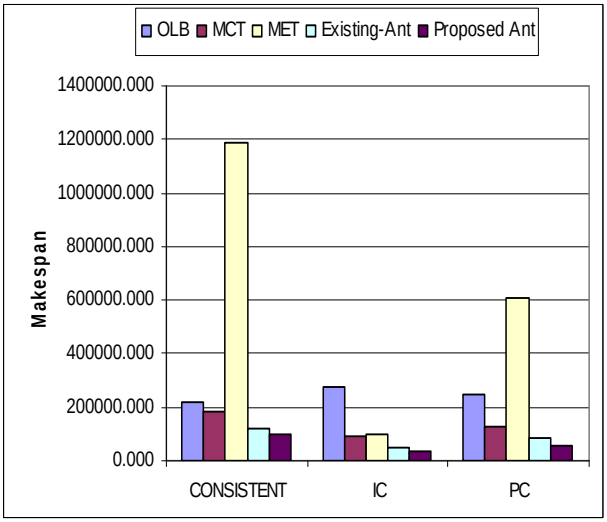


Fig. 3 : Graphical representation of table 5

Table 6: Percentage decrease in makespan value by proposed ant algorithm when compared to other algorithms. (Values in percentage)

	High Task Low Machine		
OLB	55.66	88.18	77.64
MET	47.27	65.99	55.78
MCT	91.73	66.61	90.75
Existing Ant	16.70	33.98	31.83

Table 7: Makespan values for Braun et al. benchmark (in arbitrary time units) by algorithms MCT, MET, Existing-Ant, Proposed Ant on Low Task High Machine when they are consistent, inconsistent (IC) and partially consistent (PC)

Low Task Low Machine			
	CONSISTENT	**IC**	**PC**
OLB	7309.595	89380.269	8938.389
MCT	6360.054	3137.350	4436.117
MET	39582.297	3399.284	21042.413
Existing-Ant	4208.342	1605.428	2914.206
Proposed Ant	3675.097	1073.454	1890.407

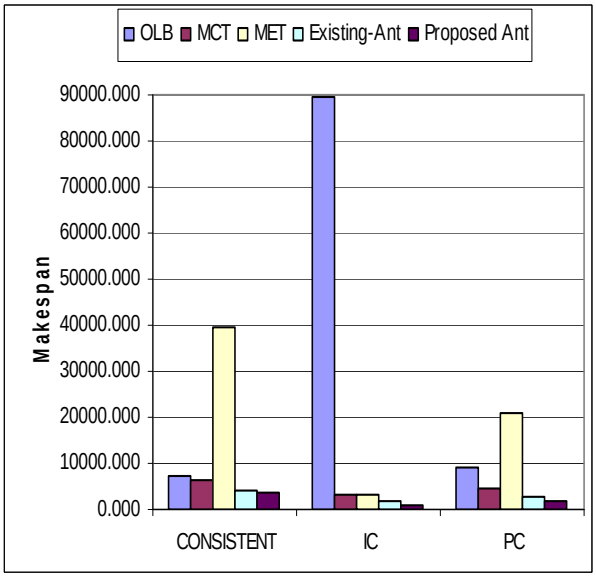


Fig. 4 Graphical representation of Table 7

Table 8 : Performance of proposed ant algorithm is shown with the help of decrease in makespan value (in percentage) in comparison with other algorithms.

	Low Task Low Machine		
OLB	49.72	98.80	78.85
MET	42.22	65.78	57.39
MCT	90.72	68.42	91.02
Existing Ant	12.67	33.14	35.13

Hence, as a result of experimental evaluation discussed so far it is very clear that the heuristic technique in general, performs much better when compared to OLB, MCT and MET. Among the heuristic techniques seen above, the Proposed Ant algorithm performs 10 percentage better than the existing ant algorithm in all possible cases on an average. Thus, addition of ET_{ij} in the calculation of free(j), that is inclusion of execution time of the ith job by the jth machine(predicted) in the calculation of probability, that the jth machine will be free, has shown a positive result in performance improvement. This improvement is in terms of decrease in makespan time.

6 Conclusions and Future Work

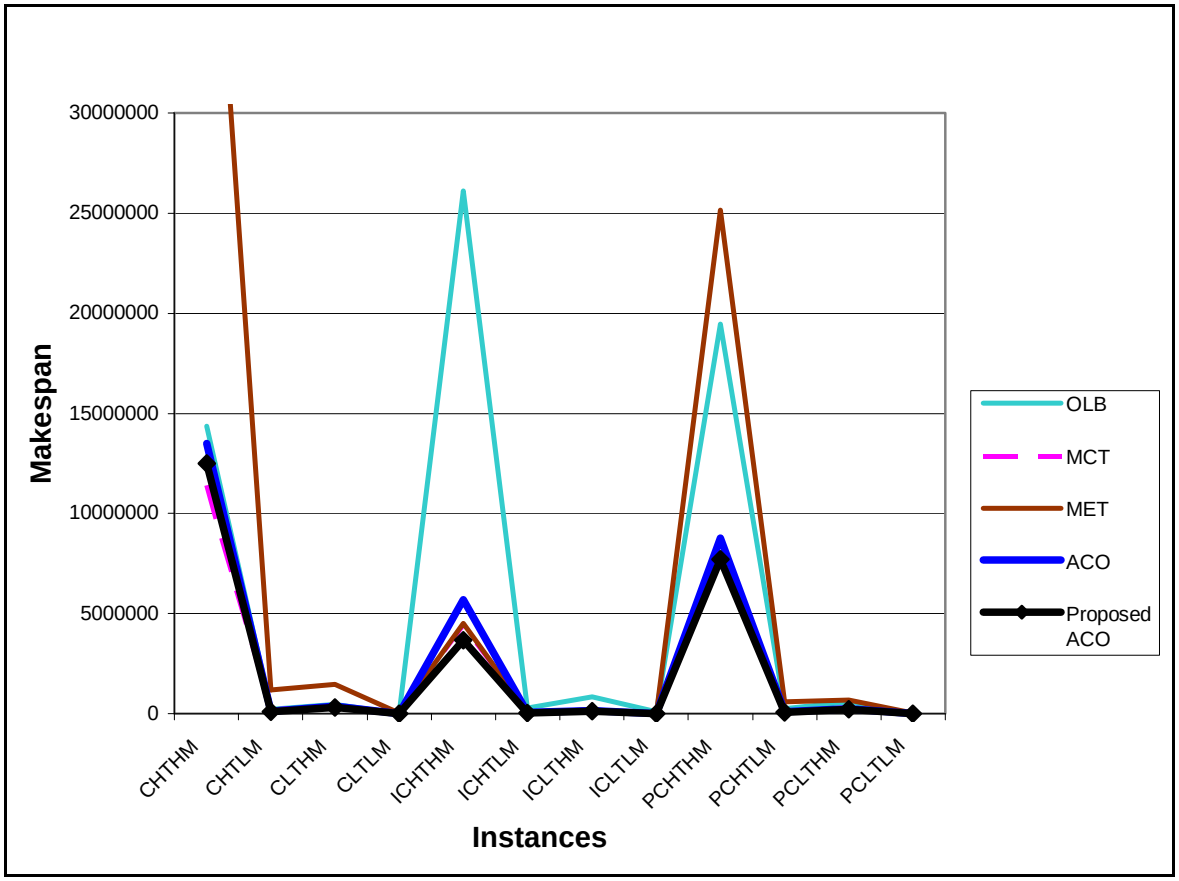
Selecting the appropriate resources for the specific task is the one of the challenging work in computational grid. This paper shows how to schedule the jobs using ACO method in computational grid system. It provides a real distributed real time system with no global control for schedulers. This method will sense the current environment and aware the contexts to decide what do to next. The resource allocation decision is not directly made by the grid system. The algorithm can adopt the system environment freely at runtime. It uses the previous information and allocates the resource optimally and adaptively in the scalable, dynamic and distribute-controlled environment.

In the study, the algorithm is designed and compared to different grid environments. Using ACO can get good workload balancing results. The proposed ACO algorithm can consistently find better schedules for several benchmark problems as compared to other techniques in the literature.

In the Grid environment the proposed ant algorithm will achieve high throughput as compared with previous ant system [8]. In this algorithm, the jobs execution time is the one of the major input parameter. The ACO algorithms can be improved the solution by combining them with local search techniques. But future research will be done using the following factors CPU workload, Communication delay and so on. The next research direction is to create different heuristic based algorithms for problem arising in grid computing. The one more future work is automatically changing the amount of pheromone evaporation and deposit depending upon the performance of the grid system. The techniques used may have to diverge somewhat from those described here, but the results presented here suggest that there is considerable scope for future research in this area.

Table 9 : Makespan values for Braun et. al. benchmark (in arbitrary time units)

	OLB	MCT	MET	ACO	Proposed ACO
u_c_hihi.0	14376662.175	11422624.494	47472299.429	13496496.722	12485079.884
u_c_hilo .0	221051.823	185887.404	1185092.968	117674.295	98017.149
u_c_lohi .0	477357.019	378303.624	1453098.003	370797.065	319142.292
u_c_lolo .0	7309.595	6360.054	39582.297	4208.342	3675.097
u_i_hihi.0	26102017.618	4413582.982	4508506.791	5703005.083	3659080.427
u_i_hilo .0	272785.200	94855.913	96610.481	48855.353	32256.535
u_i_lohi .0	833605.654	143816.093	185694.594	169621.082	125062.742
u_i_lolo .0	89380.269	3137.350	3399.284	1605.428	1073.454
u_s_hihi.0	19464875.910	8693923.896	25162058.136	8765320.713	7722087.294
u_s_hilo .0	250362.113	126587.591	605363.772	82109.419	55977.881
u_s_lohi .0	603231.467	186151.286	674689.535	260822.210	207160.951
u_s_lolo .0	8938.389	4436.117	21042.413	2914.206	1890.407



Graph 5: Graphical Representation of Table 13

References

[1] I. Foster, C. Kesselman, and S. Tuecke. "The anatomy of the Grid: Enabling Scalable Virtual Organizations". International Journal of Super computing Applications, pp.200-222, Fall. 2001.

[2] I. Foster and C. Kesselman, "Globus: A Meta computing Infrastructure Toolkit," Int'l J Super computer App., 1997, pp. 115–128.

[3] Abramson D, Giddy J, Kotler L. "High performance parametric modeling with Nimrod/G: Killer application for the global Grid " Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2000). Cancun, Mexico, 1–5 May 2000. IEEE Computer Society Press: Los Alamitos, CA, 2000.

[4] Buyya R, Abramson D, Giddy J. "Nimrod/G: An architecture for a resource management and scheduling system in a global computational Grid". Proceedings of the 4th International Conference and Exhibition on High Performance Computing in Asia-Pacific Region (HPC ASIA 2000), Beijing, China, May 2000. IEEE Computer Society Press: Los Alamitos, CA, 2000.

[5] Casavant TL, Kuhl JG. "A taxonomy of scheduling in general purpose distributed computing". IEEE Transactions on Software Engineering 1988; 14(2).

[6] Feitelson DG, Rudolph L (eds.). Proceedings of the 5th IPPS/SPDP'99 Workshop "Job Scheduling Strategies for Parallel Processing" (JSSPP 1999), San Juan, Puerto Rico, April 1999 (Lecture Notes in Computer Science, vol. 1659). Springer: Heidelberg, 1999.

[7] D. Fernandez-Baca(1989) "Allocation Modules to processors in a Distributed System", IEEE Transactions on Software Engineering. Vol.15(11): Pages 1427-1436

[8] Z. Xu, X. Hou and J. Sun, "Ant Algorithm-Based Task Scheduling in Grid Computing", Electrical and Computer Engineering, IEEE CCECE 2003, Canadian Conference, 2003.

[9] E. Lu, Z. Xu and J. Sun, "An Extendable Grid Simulation Environment Based on GridSim", Second International Workshop, GCC 2003, volume LNCS 3032, pages 205–208, 2004.
[10] H. Yan, X. Shen, X. Li and M. Wu, "An Improved Ant Algorithm for Job Scheduling in Grid Computing", In Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, 18-21 August 2005.
[11] 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)
[12] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," in 7th IEEE Heterogeneous Computing Workshop, pp. 79–87, Mar. 1998.
[13] Gong L., Sun X.H., Waston E.: "Performance Modeling and Prediction of Non-Dedicated Network Computing", IEEE Transaction on Computer, **51** 9 (2002) 1041–1055.
[14] Maheswaran M., Ali S., Siegel H.J., Hensgen D., Freund R.: "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems", 8th IEEE Heterogeneous Computing Workshop (HCW'99), San Juan, Puerto Rico, (1999) 30–44.
[15] Pinedo M.:Scheduling: "Theory, Algorithms and Systems", Prentice Hall, Englewood Clifts, NJ, (1995).
[16] Braun, T.D., Siegel, H.J., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., et al. (2001) 'A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems', J. of Parallel and Distr. Comp., Vol. 61, No. 6, pp.810–837
[17] Stefka Fidanova and Mariya Durchova," Ant Algorithm for Grid Scheduling Problem", Large Scale Computing, Lecture Notes in Computer Science No. 3743, Springer, germany, 2006, 405-412.

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 Ph.D degree in Anna Univercity, Chennai, India. Currently she is a assistant professor in the department of computer science and Engineering, Perundurai, Tamilnadu. Her area of interest are grid computing, Compiler Design and Theory of Computation. She has presented papers in National and International Conferences.

Dr.P.Balasubramanie has obtained his Ph.D 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 Ph.D scholars and guiding 15 research scholars. His area of interest include theoretical computer science, data mining, image processing and optimization Techniques.