ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering &Technology (IJARCET)
Volume 2, Issue 3, March 2013

A Novel Approach to Scheduling in Grid Environment: Enhanced Ant Colony Optimizer

D. Maruthanayagam

Research Scholar, Manonmaniam Sundaranar University,Tirunelveli, Tamilnadu, India

Dr.R. Uma Rani

Associate Professor, Department of Computer Science,Sri Sarada College for Women, Salem, Tamilnadu, India

Abstract: **In this paper we compared and evaluated the scheduling performances of existing heuristic scheduling algorithms with our proposed Enhanced Ant Colony Optimization (EACO). We take the algorithms for comparison Modified Ant Colony Optimization (MACO), MAX-MIN and RASA scheduling algorithms. We also presented a new job scheduling algorithm based on Resource Aware Scheduling Algorithm (RASA). Our algorithm aims to improve the makespan of the job scheduling, avoiding starvation, proper resource selection and allocation according to the system and network performance in dynamic Grid Environment. We achieved our goal with the help of the proposed simulator Grid Network Listing Tool (GNLT). It performed simulation process in real time environment. Thus, the task scheduling can be performed more effectively by achieving lower makespan time than the existing scheduling algorithms.**

Keywords: **Grid Scheduling, MACO Algorithm, MAX-MIN Algorithm, RASA Algorithm and EACO Algorithm.**

I. INTRODUCTION

One of the most complicated task in Grid computing is the allocation of resources for a process; i.e., mapping of jobs to various resources. This may be a NP-Complete (Non-deterministic Polynomial time) problem. For example, mapping of 50 jobs into 10 resources produces 500 possible mappings. This is because every job can be mapped to any of the resources. In our case the allocation is in terms of co-allocation which means that the job is executed on a number of resources instead of single resource. Here resource means nodes which are involved in the scheduling process. The other complexity of resource allocation is the lack of accurate information about the status of the resources. Load balancing and scheduling play a crucial role in achieving utilization of resources in grid environments [1].

Task scheduling is an integrated part of parallel and distributed computing. The Grid scheduling is responsible for resource discovery, resources selection, job assignment and aggregation of group of resources over a decentralized heterogeneous system. To get the resources information of single computer and scheduling is easy, such as CPU frequency, number of CPU's in a machine, memory size, memory configuration and network bandwidth and other resources connected in the system. It should optimize the allocation of a job allowing the execution on the optimization of resources.

The next stage is job scheduling, i.e., the mapping of jobs to specific physical resources, trying to maximize some optimization criterion. Most of the grid systems in the literature use performance-

guided schedulers, since they try to find a job to-resource mapping that minimizes the overall execution time [2] [3]. Many algorithms were designed for the scheduling of Meta tasks in computational grids. There are relatively a large number of task scheduling algorithms to minimize the total completion time of the tasks in grid computing involved systems [4]. Actually, these algorithms try to minimize the overall completion time of the tasks by finding the most suitable resources to allocate to the tasks. It should be noticed that minimizing the overall completion time of the tasks does not necessarily result in the minimization of execution time of each individual task.

This paper is organized as follows: Section 2 briefly reviews the Ant Colony Optimization **(ACO).** Section 3 discusses the Modified Ant Colony Optimization **(MACO)** techniques. Section 4 presents the **MAX-MIN** algorithm, **RASA** Algorithm and discusses the proposed **Enhanced ACO** technique, which is detailed in its subsections. Section 5 discusses about the Experimental Results and Discussion with simulation results. Section 6 concludes the paper.

2. ANT COLONY OPTIMIZATION

Ant Colony Optimization (ACO) was first introduced by Marco Dorigo as his Ph.D. thesis and was used to solve the TSP problem [5]. ACO was inspired by ant's behavior in finding the shortest path between their nests to food source. Many varieties of ants deposit a chemical pheromone trail as they move about their environment, and they are also able to detect and follow pheromone trails that they may encounter. With time, as the amount of pheromone in the shortest path between the nest and food source increases, the number of ants attracted to the shortest path also increases. This cycle continues until most of the ants choose the shortest path.

***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* $\rho = 0.05$*(pheromone evaporation value)*
T0 = 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})
   $\eta_{ij} = 1/Free(j)$
b. *Calculate current pheromone trail value*
   $\triangle T_{ij} = 1-\rho/F_k$
   where $F_k = max(free(j))$;
c. *Update the Pheromone Trail Matrix*
   $T_{ij} = \rho\ T_{ij} + \triangle T_{ij}$
d. *Calculate the Probability matrix*

$P_{ij} = T_{ij} . \eta_{ij} (1/ET_{ij}) / \Sigma\ T_{ij} . \eta_{ij} (1/ET_{ij})$

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

Figure 1: The Procedure of Ant Colony Optimization for task scheduling.

3. MODIFIED ANT COLONY OPTIMIZATION

In paper [6], they use one ant. To overcome this disadvantage a new algorithm is proposed. In this method, the probability matrix (P_{ij}) is modified and several ants are used. 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 uses the corresponding scheduling list.

The inclusion of ET_{ij} execution time of the i^{th} job by the j^{th} machine (predicted) in the calculation of probability, that the j^{th} machine will be free, has shown a positive result in performance improvement. This improvement is in terms of the decrease in makespan time. The result produced by the algorithm is little better than the algorithm in the paper [6]. The paper [7, 8] used the same formula (9) but the only difference was, in paper [8], instead of adding ET_{ij}, execution time of the i^{th} job by the j^{th} machine (predicted), in the calculation of probability, adding CT_{ij}, like formula (1) still produces better results.

$$P_{ij} = \tau_{ij} \cdot \eta_{ij} (1/CT_{ij}) / \Sigma \tau_{ij} \cdot \eta_{ij} (1/CT_{ij}) \text{----------(1)}$$

***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 $\rho = 0.05$(pheromone evaporation value)*
T0 = 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})*
   $\eta_{ij} = 1/Free(j)$

b. *Calculate current pheromone trail value*
   $\triangle T_{ij} = 1 - \rho / F_k$
   where Fk = max(free(j));

c. *Update the Pheromone Trail Matrix*
   $T_{ij} = \rho T_{ij} + \triangle T_{ij}$

***Definition:** The P_{ij} 's value has been modified to include the CT_{ij}*

d. *Calculate the Probability matrix*

$$P_{ij} = \tau_{ij} \cdot \eta_{ij} (1/CT_{ij}) / \Sigma \tau_{ij} \cdot \eta_{ij} (1/CT_{ij})$$

Figure 2: The Procedure of Modified Ant Colony Optimization for task scheduling.

4. MAX-MIN Algorithm

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. The Max-Min algorithm gives priority to the larger tasks. This algorithm firstly assigns the large or in other words time consuming tasks to the resources and then assigns the small ones [10]. The Max-Min seems to do better than the Min-Min algorithm whenever the number of shorter tasks is much more than the longer ones, but in the other cases, early execution of the large tasks might increase the total response time of the system. Also, in the Max-Min algorithm, the small tasks may wait for large ones to be executed.

```
for all tasks Ti in meta-task Mv
    for all resources Rj
        Cij=Eij+rj
do until all tasks in Mv are mapped
      for each task in Mv find the earliest
          completion time and the
resource that obtines it
      find the task Tk with the maximum
earliest  completion time
      assigne task Tk to the resource Rl
that gives the earliest completion time
      delete task Tk from Mv.
      update rl
      update Cil for all i
end do
```

Figure 3: The Procedure of MAX-MIN Algorithm for task scheduling.

4.1. Resource Aware Scheduling Algorithm (RASA)

This algorithm builds a matrix C where C_{ij} represents the completion time of the task T_i on the resource R_j. If the number of available resources is odd, the min-min strategy is applied to assign the first task, otherwise the max-min strategy is applied. The remaining tasks are assigned to their appropriate resources by one of the two strategies, alternatively. For instance, if the first task is assigned to a resource by the min-min strategy, the next task will be assigned by the max-min strategy.

In the next round the task assignment begins with a strategy different from the last round. For instance if the first round begins with the max-min strategy, the second round will begin with the min-min strategy. Jobs can be farmed out to idle servers or even idle processors. Many of these resources sit idle especially during off business hours. Policies can be in places that allow m jobs to only go to servers that are lightly loaded or have the appropriate amount of memory/processors characteristics for the particular application. As RASA consist of the max-min and min-min algorithms and both have no time consuming instructions, the time complexity of RASA is $O(mn^2)$ where m is the number of resources and n is the number of tasks (similar to Max-min and Min-min algorithms) [10].

```
for all tasks Ti in meta-task Mv
  for all resources Rj
        Cij=Ej+rj
do until all tasks in Mv are mapped
  if the number of resources is even then
      for each task in Mv find the
  earliest completion time and
      the resource that obtines it
      find the task Tk with the
  maximum  earliest  completion time
       assigne task Tk to the resource Rl
that   gives the earliest completion time
       delete task Tk from Mv
       update rl
       update Cil for all i
  else
      for each task in Mv find the
   earliest completion  time and
      the resource that obtines it
       find the task Tk with the
   minimum  earliest  completion time
      assigne task Tk to the resource Rl
that gives the earliest completion time
      delete task Tk from Mv
      update rl
      update Cil for all i
 end if
end do
```

Figure 4: The Procedure of RASA Algorithm for task scheduling.

4.2. The Proposed EACO Algorithm

EACO fully run under **dynamic grid** environments this information that can be retrieved from a many servers includes operating system, processor type and speed, the number of available CPUs and software availability as well as their installation

locations. EACO aims to improve the makespan of the job scheduling, avoiding starvation, proper resource selection and allocation according to the system and network performance in dynamic Grid Environment.

As RASA consists of the max-min and min-min algorithms and both have no time consuming instructions. **ACO** and **RASA** algorithms **(EACO)** incorporate in which intend to optimize workflow execution times on grids. Our experimental results of applying the proposed EACO on scheduling independent tasks within grid environments demonstrate the applicability of EACO achieving schedules with comparatively lower makespan than existing heuristic scheduling algorithms. We compared the scheduling performances of MACO, MAX-MIN and RASA with proposed EACO have also been detailed here.

***Step 1**: Collect all necessary information about the jobs (n) and resources (m) of the system*
__for__ each tasks T_i and resources R_j allocations
***Step 2:** __do__ until all tasks in Dynamic mapped*
Find and calculate network and system speeds
Remove idle systems from scheduling
Calculate Estimate CT=Executed Time (ET) + Ready Time (RT)
***Step 3:** Select the task (i) and resource (j) randomly.*
***Step 4:** Repeat the following until all jobs are executed.*
Find the resource free times
Find the Jobs with the
Minimum and Maximum earliest completion time
***Step 5:** Find out best probability makespan P from ET and CT*
*pET = 1/ ET*Avg(maxET – minET)*
*pCT = (Best Resource * ET) + Avg(maxCT –minCT)*
Assign task T to the resource R that gives
the better completion time from min and max
end while

Figure 5: The Procedure of proposed EACO Algorithm for task scheduling.

5. Experimental Results and Discussion

Specification of the resources is according to resources speed (MIPS) and bandwidth (Mbps), specification of the tasks depends on instructions and data (MIPS) completion time of the tasks on each of the resources. Here we used four resources R1, R2, R3 and R4 & five tasks T1, T2, T3, T4 and T5 are in the mapping and the grid manager is supposed to schedule all the tasks within the resources of 20 iterations. Table 1 is shown the ready times and executed times of the resources and tasks.

Table 1: Ready and executed times of the Resources and Tasks.

Task / Resource	Task1		Task2		Task3		Task4		Task5	
	RT	ET	RT	ET	RT	ET	RT	ET	RT	ET
Resource1	02.44	16.44	16.66	22.24	38.42	48.44	64.86	76.78	82.36	98.22
Resource2	04.66	18.88	24.22	32.68	34.68	44.48	62.64	74.24	84.26	100.32
Resource3	06.88	20.44	22.48	30.24	36.24	46.68	66.66	78.32	86.12	102.24
Resource4	08.42	22.12	26.20	34.24	40.12	52.54	68.62	80.06	88.22	104.38

Consider in an environment, Task1 (T1) is to be allotted to one of the four resources. Among them Resource1 (R1) is not in working condition. Resource2 (R2) is engaged with another job. Among the available resources R3 & R4, the mapping tool (GNLT) selects R4 since it has high processor speed than R3.

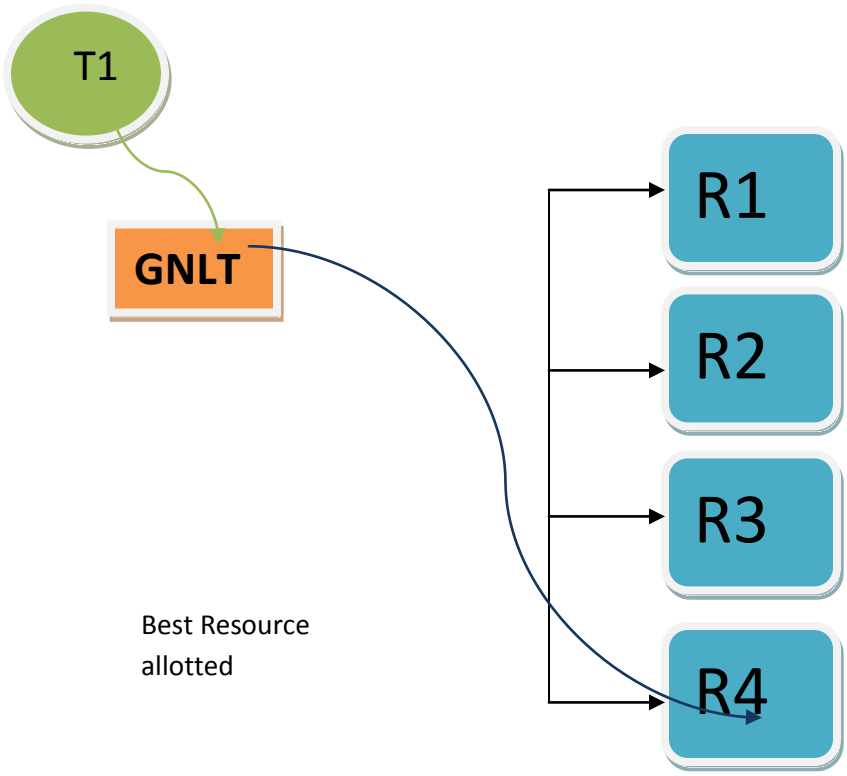

Figure 6: Allotted best resource.

The performance of the proposed algorithm EACO is evaluated under five different jobs and four resources and the results are compared against the conventional

algorithms such as MACO, MAX-MIN and RASA. The results that are obtained in different experiments are given in following Table2.

Table 2: Compare Probability Makespan Time for **MACO**, **MAXMIN-ACO, RASA –ACO** and proposed **EACO**.

Iterations	Task/Resource	MACO	MAXMIN- ACO	RASA- ACO	EACO
1.	T1/R1	107	103.56	100.00	84.56
2.	T1/R2	97.66	82.44	42.59	37.17
3.	T1/R3	75.12	59.78	26.26	17.78
4.	T1/R4	59.45	54.68	17.98	14.67
5.	T2/R1	44.66	39.59	15.95	10.9
6.	T2/R2	70.4	69.67	67.39	55.78
7.	T2/R3	72.45	72	71.60	59.56
8.	T2/R4	53.65	52.78	51.79	48.78
9.	T3/R1	30.44	28.56	26.86	20.45
10.	T3/R2	32.21	30.56	29.96	26.78
11.	T3/R3	29.67	27.56	26.19	24.56
12.	T3/R4	23.78	20.89	19.82	17.45
13.	T4/R1	18.34	13.24	9.80	7.56
14.	T4/R2	18.22	12.98	10.15	8.67
15.	T4/R3	17.34	12.26	8.86	5.89
16.	T4/R4	17.44	12.12	8.22	5.34
17.	T5/R1	16.45	11.55	5.79	3.68
18.	T5/R2	16.33	10.89	5.40	2.68
19.	T5/R3	15.78	10.89	5.07	2.22
20.	T5/R4	15.23	9.45	4.74	2.12

Our proposed scheduling algorithm which achieves makespan time 14.67 seconds for Task1 with Resource R4 compare than R1,R2 and R3.Similarly T2,T3,T4 and T5 getting minimum makespan times with optimal resource.

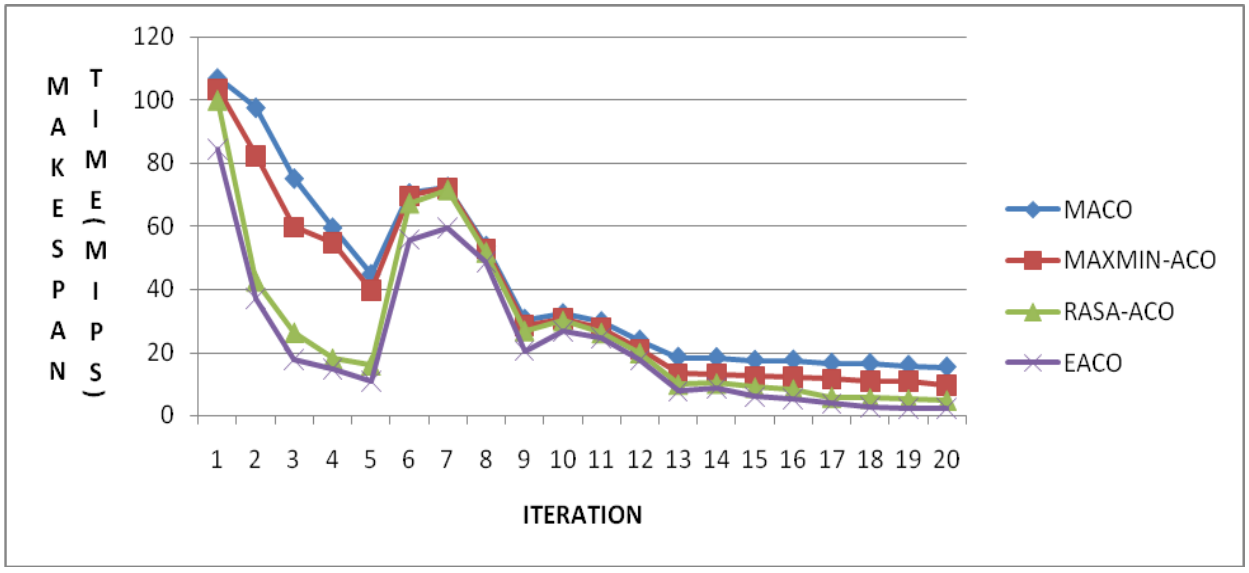

Figure :7 Comparative Probability Makespan Time of MACO,MAXMIN-ACO,RASA- ACO and EACO.

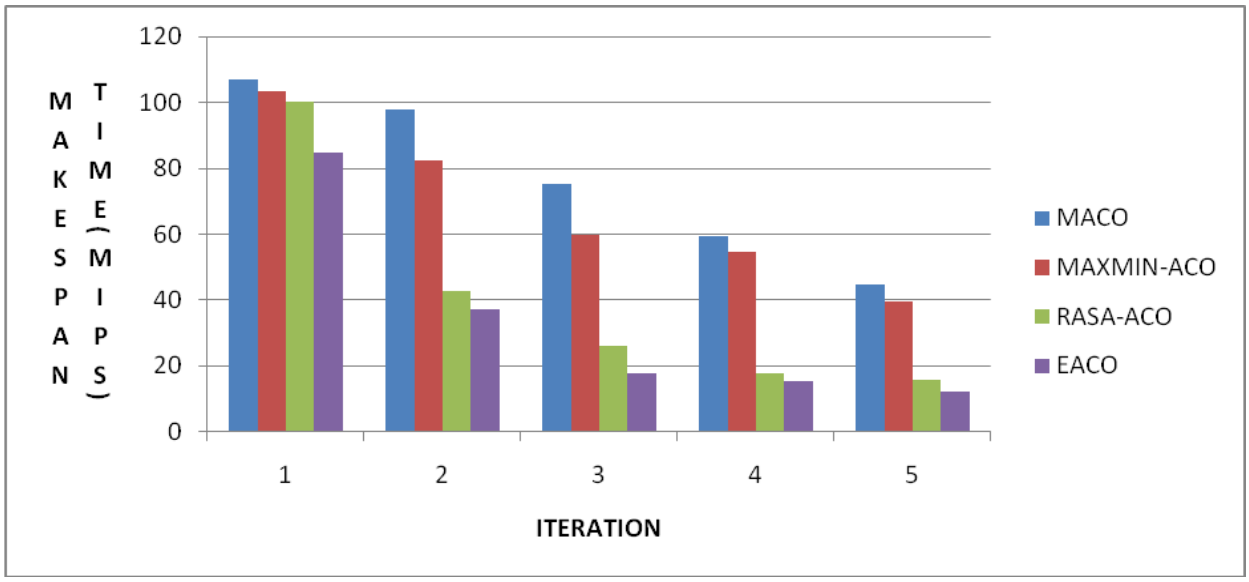

Figure :8 Five iteration's Comparative Probability Makespan Time of MACO,MAXMIN-ACO,RASA- ACO and EACO.

Table 3: Compare Probability Completion Time for **MACO**, **MAXMIN-ACO, RASA –ACO** and Proposed **EACO.**

Iterations	Task/Resource	MACO	MAXMIN- ACO	RASA- ACO	EACO
1.	T1/R1	108.00	104.67	100.00	85.66
2.	T1/R2	98.65	83.45	50.55	38.12
3.	T1/R3	76.09	60.89	34.03	28.88
4.	T1/R4	60.98	55.78	25.81	16.76
5.	T2/R1	45.89	40.69	19.49	13.98
6.	T2/R2	40.87	35.67	16.61	12.78
7.	T2/R3	38.98	33.78	14.49	10.66
8.	T2/R4	26.98	21.78	12.91	9.43
9.	T3/R1	22.87	17.67	10.79	8.55
10.	T3/R2	21.80	16.60	9.94	7.00
11.	T3/R3	20.79	15.59	9.21	7.44
12.	T3/R4	20.22	15.02	8.60	7.12
13.	T4/R1	19.45	14.25	7.47	6.34
14.	T4/R2	19.12	13.92	7.09	6.05
15.	T4/R3	18.44	13.24	6.75	5.04
16.	T4/R4	18.42	13.22	6.45	5.12
17.	T5/R1	17.54	12.65	5.71	4.66
18.	T5/R2	17.43	11.90	5.51	3.44
19.	T5/R3	16.99	11.22	5.32	3.12
20.	T5/R4	16.12	10.24	5.16	3.80

Our proposed scheduling algorithm which achieves completion time 16.76 seconds for Task1 with Resource R4 compare than R1,R2 and R3.Similarly T2,T3,T4 and T5 getting minimum completion times with optimal resource.

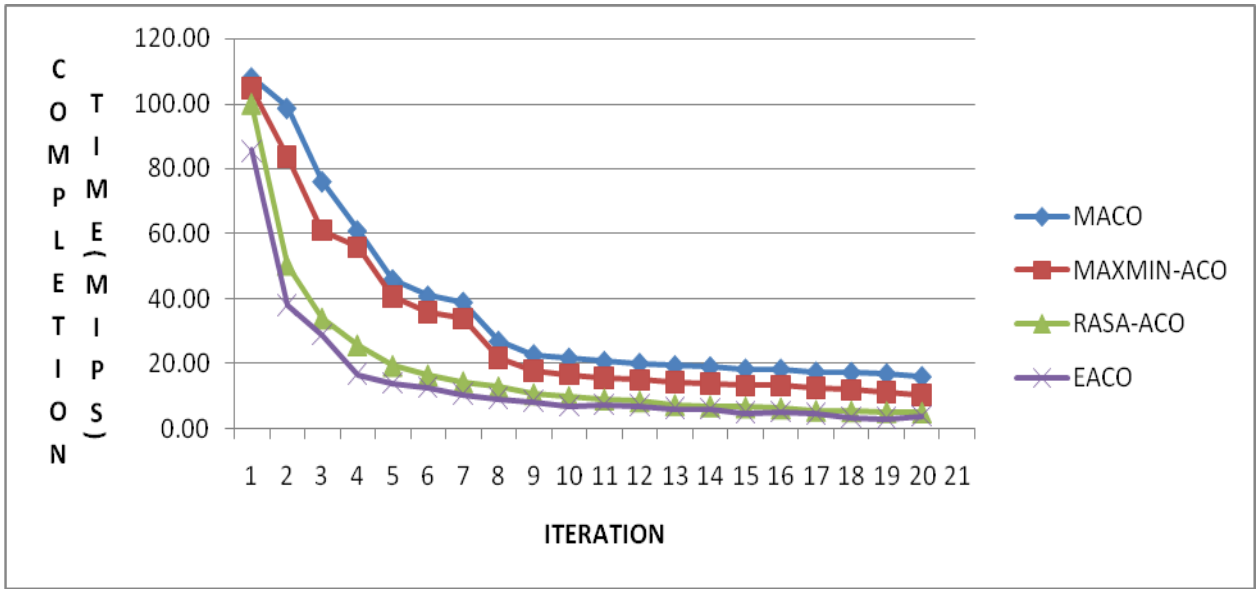

Figure :9 Comparative Probability Completion Time of MACO,MAXMIN-ACO,RASA- ACO and EACO

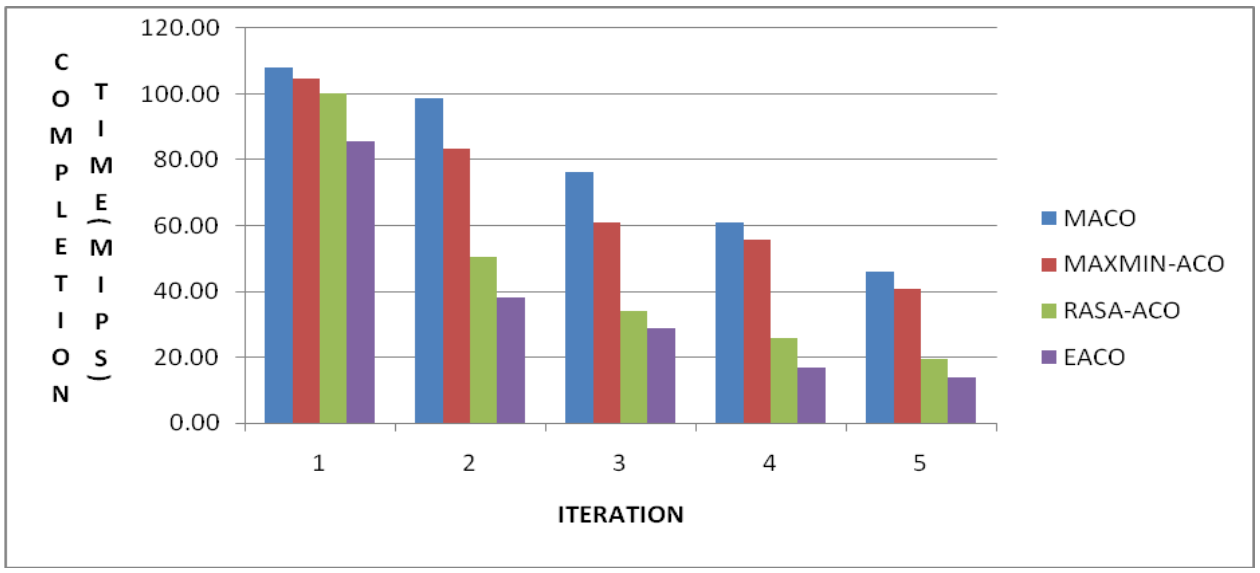

Figure :10 Five iterartion's Comparative Probability Completion Time of MACO,MAXMIN-ACO,RASA- ACO and EACO

5.1. Mapping tool

In this paper, we seek to provide a simulation platform capable of consuming the workload of actual jobs provided by grid environment, so as to allow the studying of performance and behavioral aspects of the system, as well as identifying any potential weakness within the system. It is necessary for the simulator to possess the ability to replace an entire functional grid environment, simulating all the resources and networking infrastructure which is used in a grid environment of today. As a result, the GNLT simulator was conceived.

The GNLT simulation tool helps the scheduler to send the jobs good and efficient site and also complete the jobs in time or before the job completion time. Here, we invoke the Enhanced Ant Colony Optimization (EACO) technique for finding shortest path. We give jobs to three sites randomly and then analyze the shortest path site. The GNLT analysis and list the performance details of the resources before mapping with the machines.

During the experimentation process, totally 1-month experiments were conducted, each with a different seed feeding the GNLT simulator. On the other hand, our GNLT simulator testing was performed on a single dedicated dual-core 2.4GHz machine with limited processing capabilities, limiting our processing threads to a maximum of 600. With the high resource utilization nature of this simulation attempts to squeeze out more threads often resulted in instability to the system due to overloading of the hardware. We believe that this phenomenon can be alleviated if we can create multiple instances of the GNLT simulator running in distributed mode.

6. CONCLUSION

In this paper, the proposed method has achieved the minimum makespan and completion time. The drawbacks of existing scheduling algorithms were solved by considering some efficient factors in the job scheduling process. Thus, the proposed technique has achieved high performance in allocating the available jobs to the accurate resources and also attained a high efficiency. The performance of the proposed task scheduling algorithm was compared with three existing algorithms namely MACO, MAXMIN, and RASA with experimental results that prove that the proposed job scheduling technique has attained high accuracy and efficiency than the three existing techniques. Hence, the proposed EACO task scheduling algorithm is capable of finding the optimal jobs to the resources and also achieving the minimum completion time.

REFERENCES

[1]. Xiao L, Zhu Y, Ni L M. and Xu Z, Gridis: An incentive based grid scheduling. IPDPS'05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Raman, R., Livny, M., Solomon, M.: Matchmaking: Distributed resource management for high throughput computing. Int. Symp. on High Performance Distributed Computing (1998).

[3] Huedo, E., Montero, R.S., Llorente, I.M.: An Experimental Framework for Executing Applications in Dynamic Grid Environments. NASA-ICASE Technical Report 2002-43 (2002).

[4] Mohammad Khanli, L. and M. Analoui, Resource Scheduling in Desktop Grid by

Grid-JQA. The 3rd International Conference on Grid and Pervasive Computing, IEEE, 2008.

[5] K. Kousalya and P. Balasubramanie,"To Improve Ant Algorithm's Grid Scheduling Using Local Search. (HTTP://WWW.IJCC.US), VOL. 7, NO. 4, DECEMBER 2009.

[6] Stefka Fidanova and Mariya Durchova, "Ant Algorithm for Grid Scheduling Problem", Large Scale Computing, Lecture Notes in Computer Science No. 3743, Springer, germany, pp. 405-412, 2006.

[7] Kousalya.K and Balasubramanie.P, "Resource Scheduling in Computational Grid using ANT algorithm", In Proceedings of the International Conference on Computer Control and Communications, Pakistan,2007.

[8] Kousalya.K and Balasubramanie.P, "An Enhanced ant algorithm for grid scheduling problem", International Journal of Computer Science and Network Security,Vol. 8, No.4, pp.262-271, 2008.

[9] Li Liu, Yi Yang, Lian Li and Wanbin Shi, "Using Ant Optimization for Super Scheduling in Computational Grid", In Proceedings of the IEEE Asiapasific Conference on Services Computing, 2006.

[10] Saeed Parsa and Reza Entezari-Maleki RASA: A New Task Scheduling Algorithm in Grid Environment World ppliedSciences Journal 7 (Special Issue of Computer & IT): 152-160, 2009,ISSN 1818.4952.

Authors Profile

D.Maruthanayagam received his M.Phil, Degree from Bharathidasan University, Trichy in the year 2005. He has received his M.C.A Degree from Madras University, Chennai in the year 2000.He has published 6 papers in International journals and 6 papers in National and International conferences. He is working as a Head, Department Computer Science, Siri PSG Arts and Science College for Women, Sankari, Salem, Tamilnadu, India. His areas of interest include Computer Networks, Grid Computing and Mobile Computing.

Dr.R.Uma Rani received her Ph.D., Degree from Periyar University, Salem in the year 2006. She is a rank holder in M.C.A., from NIT, Trichy. She has published around 65 papers in reputed journals and National and International conferences. She has received the best paper award from VIT, Vellore, Tamil Nadu in an International conference. She has done one MRP funded by UGC. She has acted as resource person in various National and International conferences. She is currently guiding 5 Ph.D., scholars. She has guided 20 M.Phil, scholars and currently guiding 4 M.Phil, Scholars. Her areas of interest include information security, data mining, fuzzy logic and mobile computing.