INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

A PROPOSED DYNAMIC EACO SCHEDULING ALGORITHM IN GRID COMPUTING ENVIRONMENT WITH EFFECTIVE COMPARISON ANALYSIS OF OTHER HEURISTIC SCHEDULING ALGORITHMS

Dr.D.Maruthanayagam
Head, Department of Computer Science, Siri PSG Arts & Science College for Women, Sankari, Salem (Dt)

Abstract

In this paper we compared and evaluated the scheduling performances of existing heuristic scheduling algorithms with our novel Dynamic Enhanced Ant Colony Optimization (EACO). We take the algorithms for comparison Modified Ant Colony Optimization (MACO), MAXMIN and RASA scheduling algorithms. Our algorithm aims to minimize 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. The experimental results prove that the proposed task scheduling method has attained high precision and efficiency than the three hybrid methods through MACO, MAXMIN-ACO and RASA-ACO. Hence the proposed EACO task scheduling algorithm is capable of finding the optimal jobs to the optimal resources as well as achieving the minimum makespan and completion time.

Keywords: MACO Algorithm, MAX-MIN Algorithm, RASA Algorithm, GNLT and Dynamic EACO Algorithm.

1. Introduction

Reviews on the Various Heuristic Scheduling Algorithms

1.1 Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a paradigm for designing Meta heuristic algorithms for combinatorial optimization problems. The inspiring source of ACO is the foraging behavior of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it evaluates quantity and quality of the food and carries some of the food found to the nest. During the return trip, the ant deposits a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. The indirect communication between the ants via the pheromone trails allows them ***to find shortest paths*** between their nest and food sources. This behavior of real ant colonies is exploited in artificial ant colonies in order to solve discrete optimization problems [1, 2].The pheromone trail laying and following behavior of the ants induces a positive feedback process whereby trails with ***high concentration of***

pheromones become more and more attractive as more ants follow them. Researchers use the ACO to find the shortest paths and reach high concentration of pheromones, since ACO utilize Grid Computing scheduling operations to find shortest paths and to assign tasks in optimal resources [3]. ACO concepts when implemented in the proposed algorithm work the above biological terms like foods are called as resources; ants are called as jobs or tasks. The following sections explain few families of the ACO Algorithms tested and implemented in different manners in Grid Computing environment.

1.2 Modified Ant Colony Optimization (MACO)

The MACO algorithm is evaluated using the simulated execution times for a grid computing environment. Before starting the grid scheduling, the expected execution time for each task on each machine must be estimated and represented by an ET matrix. The collection of independent tasks with no data dependencies is called as a meta-task. Each machine executes a single task at a time. The meta-task size is one and the number of machines is 'm'. Each row of ET matrix consists of the estimated execution time for a job on each resource and every column of the ET matrix is the estimated execution time for a particular resource of list of all jobs in the job pool. ET_{ij} is the expected execution time of task t_i and the machine m_j . The time to move the executable and data associates with the task t_i includes the expected execution matrix ET_{ij} . For this algorithm, it is assumed that there are no inter-task communications, each task can execute on each machine, and the estimated expected execution times of each task on each machine is known [4][5].

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.*

1. *Calculate the heuristic information* (η_{ij})
   $\eta_{ij} = 1/Free(j)$
2. *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 MACO Algorithm

1.3. MAXMIN -Ant Colony Optimization (MAXMIN–ACO)

MAX–MIN Ant Colony Optimization is an improvement over the original Ant System. The Max-Min heuristic also begins with the set of all unmapped tasks. Then, the set of minimum completion times is found. Next, the task with the overall maximum completion time from M is selected and assigned to the corresponding machine. At last, the newly mapped task is removed from set of unmapped task, and the process repeats until all the tasks are mapped [6]. Intuitively, Max-Min attempts to minimize the penalties incurred from performing tasks with longer execution times**.** For example, assume that the Meta task being mapped has many tasks with very short execution times and one task with a

very long execution time. Mapping the task with the longer execution time to its best machine first allows this task to be executed concurrently with the remaining tasks (with shorter execution times) [7]

for *all tasks* T_i *in meta-task* M_v
  for *all resources* R_j
    $C_{ij}=E_{ij}+r_j$
do *until all tasks in* M_v *are mapped*
  for each task in M_v *find the earliest*
    completion time and the resource that obtains it
  find the best completion task Tij with the ***maximum*** *earliest completion time*
    if $\tau_{min} < T_{ij} < \tau_{min}$ then
      $T_{ij} = (1\text{-}p).T_{ij} + \Delta\ T_{ij}$ *best*
    End if
  assign best completion time and task Δ *Tij to the best resource* R_l
  that gives the earliest completion time
  update r_l
  update C_{il} *for all i*
end do

Figure -2: The MAXMIN-ACO Algorithm

1.4 Rasa-Ant Colony Optimization (RASA-ACO)

This algorithm is built two well-known task scheduling algorithms, Min-min and Max-min. Resource Aware Scheduling Algorithm **(RASA)** uses the advantages of the both algorithms and covers their disadvantages. To achieve this, RASA firstly estimates the completion time of the tasks on each of the available grid resources and then applies the Max-min and Min-min algorithms, alternatively. In this respect, RASA uses the Min-min strategy to execute small tasks before the large ones and applies the Max-min strategy to avoid delays in the execution of large tasks and to support concurrency in the execution of large and small tasks.

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 consists 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) [8].

for *all tasks* T_i *in meta-task* M_v
for *all resources* R_j
$C_{ij}=E_j+r_j$
do *until all tasks in* M_v *are mapped*
if *the number of resources is even then*
for each task in M_v *find the*
earliest completion time and
the resource that obtains it
find the task T_k *with the*
maximum earliest completion time
assign task T_k *to the resource* R_l *that gives the earliest completion time*
delete task T_k *from* M_v

update r_l
update C_{il} *for all i*
else
for each task in M_v *find the*
earliest completion time and
the resource that obtains it
find the task T_k *with the*
minimum earliest completion time
assign task T_k *to the resource* R_l *that gives the earliest completion time*
delete task T_k *from* M_v
update r_l
update C_{il} *for all i*
end if
end do

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

This work evaluated the comparison of the probability makespan and completion time of existing algorithms MACO, MAXMIN-ACO and RASA-ACO for job scheduling. The above all methods did not concentrate on the efficient factors like high through put, dynamic mapping and scheduling based on system speed and network performance etc. during the task scheduling function. The lack of such factors in the scheduling operation has mechanically decreased the performance. To overcome such drawbacks in the existing techniques, new efficient optimization algorithms have been emerged in the field of grid computing. So, another efficient task scheduling algorithm is required to achieve minimum makespan and completion time in grid scheduling [9].

2. Proposed Work

2.1 Proposed Dynamic Grid Scheduling Algorithm-(EACO)

The main aim of this paper is to present a better task scheduling method by solving the drawbacks that currently exist in the review of the algorithms. The proposed work has considered some efficient factors such as (i) High through put (ii) Dynamic mapping and (iii) Scheduling based on system speed and network performance. The proposed Enhanced Ant Colony Optimization (EACO) method professionally chooses the task to the resources depends on ready time and execution time of the task. The main objective of the proposed system is to minimize the makespan time and minimize the completion time.

Enhanced Ant Colony Optimization (EACO) fully run under **dynamic grid** environments. This proposed algorithm retrieves the details of many resources for efficient job allocation. EACO aims at improving the makespan of the job scheduling, avoiding starvation, and proper resource selection and allocation according to the system and network performance in dynamic grid environment.

In this work, the EACO ensure high throughput in grid computing environment. EACO can be used to schedule many independent tasks with many resources, dealing them out to the different computer (resource) processors in the grid. As soon as a processor finishes one task, the next task arrives and finished. In this way, hundreds of tasks can be performed in a very short time.

Dynamic scheduling is usually applied when it is difficult to estimate the cost of applications or jobs. Dynamic task scheduling has two major components: system state estimation (other than cost estimation in static scheduling) and decision making. System state estimation involves collecting state information throughout the grid and constructing the estimation. On the basis of the estimate, decisions are made to assign a task to a selected resource. Since the cost for an assignment is not available, a natural way to keep the whole system health is balancing the loads of all resources.

Step 1: *Collect all necessary information about the jobs (n) and resources (m) of the system for each tasks Ti and resources Rj 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- 4: Procedure of proposed EACO Algorithm for task scheduling

The grid computing can offer a resource dynamic load balancing effect by scheduling grid jobs on machines with low utilization. The two reasons for happen; first, an unexpected peak can be routed to relatively idle machines in the grid and the second, if the grid is already fully utilized, the lowest priority work being performed on the grid. Then, it can be temporarily suspended or even cancelled and performed again later to make usage for the higher priority work.

In this work, if the size of the job is known, if it is a kind of job that can be sufficiently split into sub jobs, and if enough resources are available after preempting lower priority work, a grid can bring a very large amount of processing power to solve the problem. The advantage of dynamic load balancing over static scheduling is that the system need not be aware of the run-time behavior of the application before execution. It is particularly useful in a system where the primary performance goal is maximizing resource utilization, rather than minimizing runtime for individual jobs. If a resource is assigned too many tasks, it may invoke a balancing policy to decide whether to transfer some tasks to other resources, and which tasks to transfer. According to who will initiate the balancing process, there are two different approaches: sender-initiated where a node that receives a new task but doesn't want to run the task initiates the task transfer, and receiver-initiated where a node that is willing to receive a new task initiates the process.

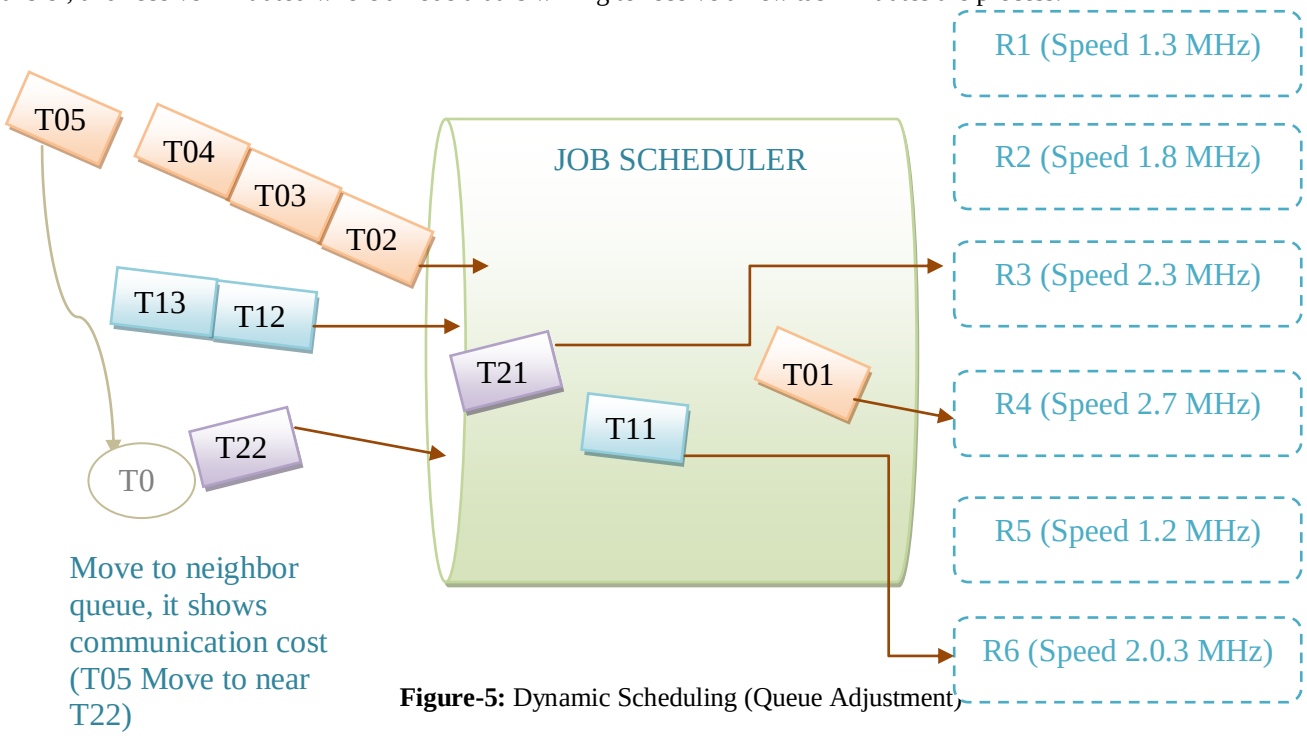


Figure-5: Dynamic Scheduling (Queue Adjustment)

Queue adjustment has shown in the above figure-5 in the grid computing system, this could be very costly due to the considerable communication delay. So, some adaptive local rebalancing heuristic can be applied. For example, tasks are initially distributed to all resources, and then, instead of computing the global rebalance, the rebalancing only happens inside a "neighborhood" where all resources are better interconnected. This approach has several advantages: the initial loads can be quickly distributed to all resources and started quickly; the rebalancing process is distributed and scalable; and the communication delay of rebalancing can be reduced since task shifting only happens among the resources that are "close" to each other.

3. Implementation of Proposed EACO with GNLT

The Grid Network Listing Tool (GNLT) helps to schedule the jobs towards good and optimal resources in efficient manner. This work, invokes the Enhanced Ant Colony Optimization (EACO) technique for finding shortest path with GNLT. The testing tool **GNLT (Grid Network Listing Tool)** was used to evaluate the performance. **GNLT** can be improved using some form of operating systems, hardware, software, different storage capacities, CPU speeds, network connectivity and technology needs. Tool has to find out the performance of the machines and then testing the processor speed of all machines in the particular network sites.

The GNLT finds the performance of the machines like processor speed, RAM speed and idle machines detail in the grid network environments before scheduling process. This simulation tool has to find out the performance of the machines and then testing the processor speed of all machines in the particular network sites. For example, there are three sites like A, B and C. Users jobs are entered in to the job scheduler. The job scheduler has to give the jobs to the above mentioned sites randomly. In this time, apply the proposed EACO algorithm for best scheduling for those incoming jobs. This tool helps that moment to easily find the particular site machines whether they are well equipped or not. Additionally, it displays the machine performance details (such as processor speed, RAM speed) in that particular site. Figure-6 shows the searching process of the best resource for task scheduling in grid networks.

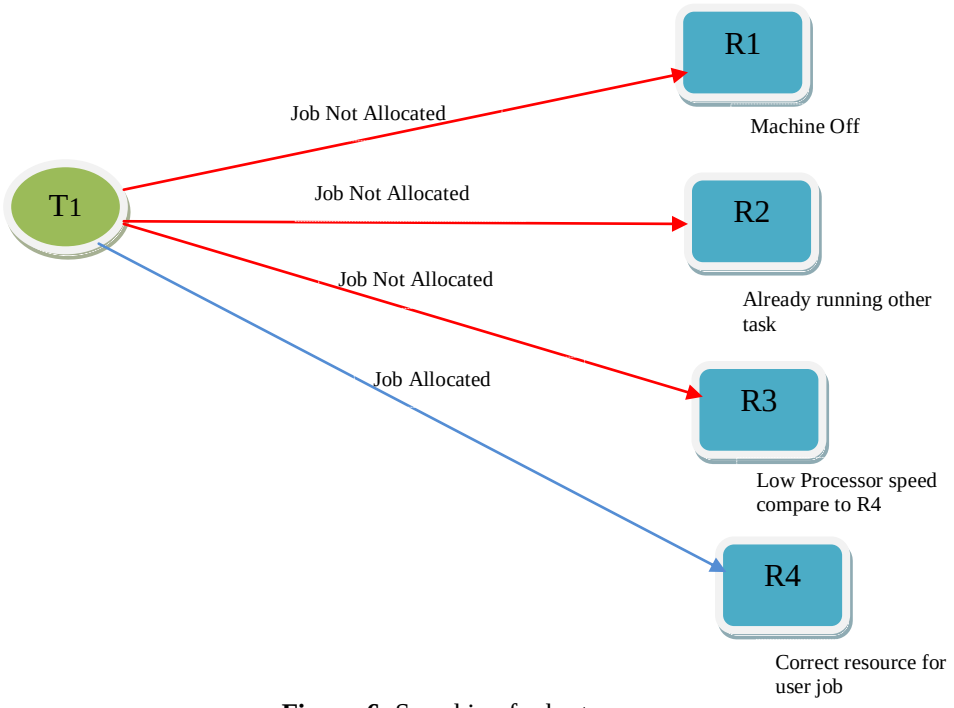


Figure-6: Searching for best resource

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 tool selects R4 since it has high processor speed than R3.

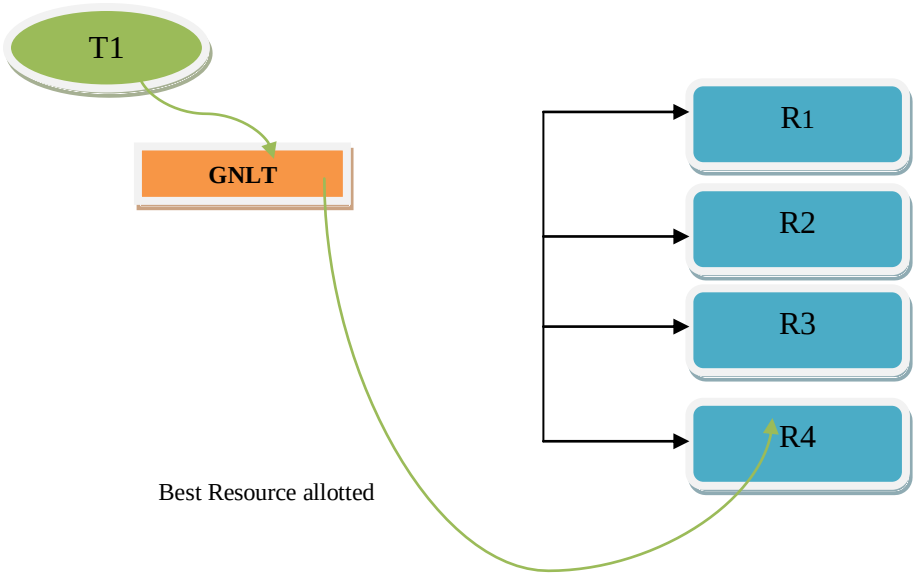


Figure -7: Best resource Allotment

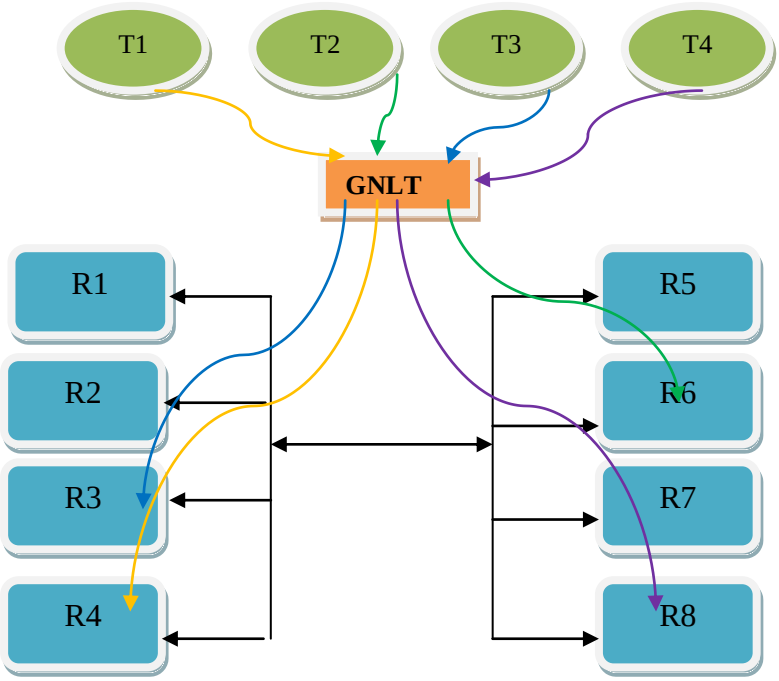


Figure-8: Four tasks and mapping with best resources

The above figures 7 & 8 show the four Tasks (T1, T2, T3 and T4) and mapping with the best resources. This tool finds best resource as well as mapping the good performance machines including multiple numbers of resources. The GNLT performs the action, T1 map with R4, T2 map with R5, T3 map with R3 and also T4 map with R8 depends upon the machines performance. The GNLT analysis and list the performance details of the resources before mapping with the machines.

3.1 Comparison Of Dynamic Eaco With Other Scheduling Algorithms

This paper compared and evaluated so far about ET_{ij} matrix environment scheduling of MACO, MAXMIN-ACO and RASA-ACO. The above scheduling algorithm's job assignment strategy will never starts before their execution. The proposed EACO method fetches the suitable resources based on speeds of processor, RAM and networks before assigned jobs to the resources. Another problem in ET_{ij}, while scheduling operation cannot identify the off line and low performance machines. This drawback is overcome by proposed dynamic EACO. In dynamic runtime environment job assignment is done automatically.

Dynamic tasks assignment assumes a continuous stochastic stream of incoming tasks. Very little parameters are known in advance for dynamic tasks assignment. Obviously, it is more complex than static tasks assignment for implementation, but achieves better throughput. Also it is the most desired because of the application demand. EACO algorithm also aims for reducing the average response time of tasks submitted to the grid, respecting the constraints of dependency between tasks and reducing communication costs.

This work, GNLT **s**oftware for grid computing will be able to adapt dynamically to changes in the underlying execution environments, in order to provide high performance, throughput, and quality of service to end users. EACO can be used to tackle the complexity of non-functional properties such as load balancing and fault tolerance.

Dynamic EACO algorithm execution covered all the need of grid job scheduling, which are various processor speeds, differences in the available memory and the usage of shared resources require the application to have a smart initial load distribution as well as dynamic load balancing. The differences in the communication characteristics for communication between processes located on potentially different sites require distinct programming techniques for hiding the wide area latency and dealing with the low bandwidth.

The dynamic EACO algorithm is implemented in real time environment by JAVA platform. The grid network formations are in the same platform. The grid computing network contains a server and 10 machines. The duty of server allocates jobs by chosen best machines, it first list out the speed of processor, RAM and network rate**s**. User can know idle machines so they can avoid those machines from scheduling in the grid network. Java programs contain gridserver.java, gridjobs.java and gridclient.java. Users have to install gridclient.class file on all the client machines. Jobs may assign like sending files, sharing the video files, downloading etc. Therefore, the executions of GNLT are given below may assign like sending files, sharing the video files, downloading etc. Therefore, the executions of GNLT are given below.

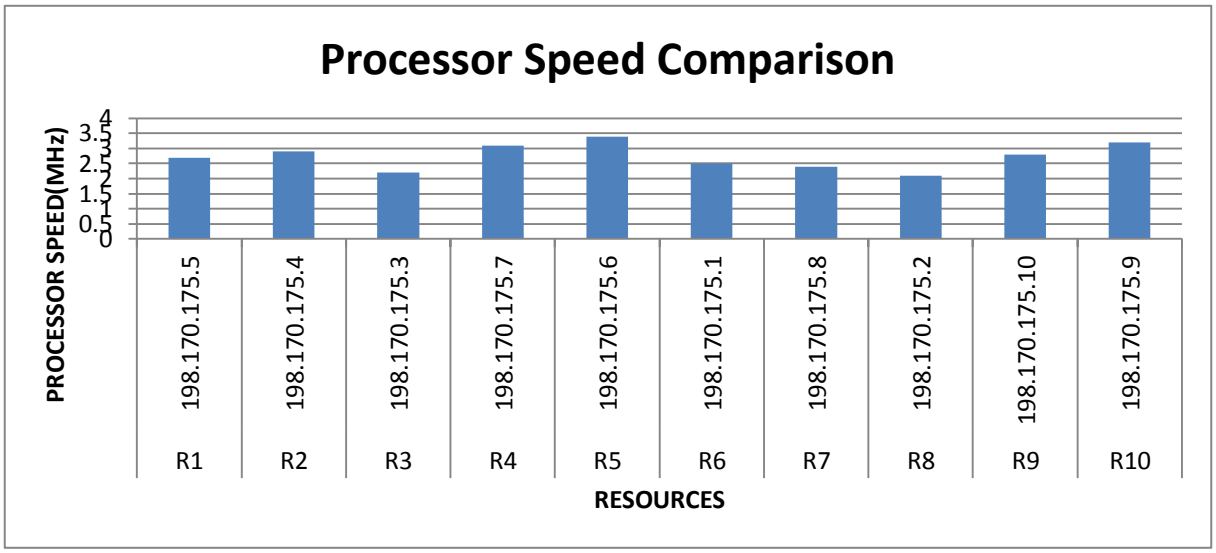
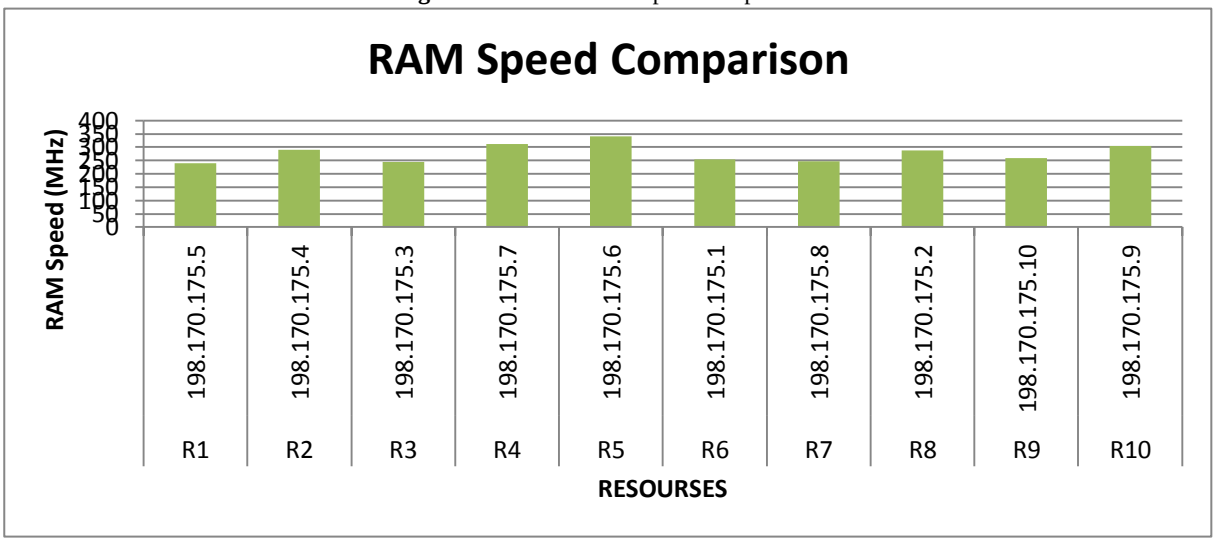


Figure-9: GNLT-Processor speed comparison

Figure-10: GNLT-RAM speed comparison



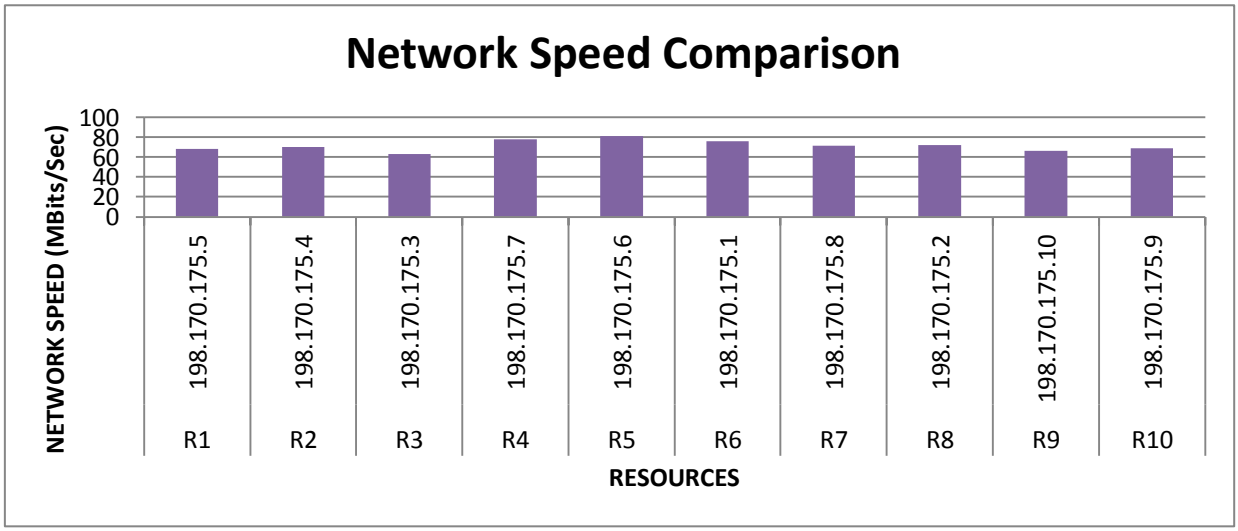


Figure-11: GNLT-Network speed comparison

If the ET_{ij} matrix scheduling MACO, MAXMIN-ACO, RASA-ACO algorithms are invoked for 4 tasks and 10 resources means it is getting 40 iterations (4x10) whereas dynamic EACO used only 4 iterations, compared to static scheduling it saved 36 iterations process times and workloads. EACO avoid unwanted ride for assigning jobs in grid network. Task 1 is scheduled to Resource 5 (198.170.175.6) because highest speed processor, RAM, Network line respectively 3.4 MHz, 340 MHz and 81 MB/SEC. Next best performance Task 2 is scheduled to Resource 10 (198.170.175.9) because highest speed processor, RAM speed, Network line respectively 3.2 MHz, 304 MHz and 69 MB/SEC. Task 3 and Task4 also scheduled same way to R4 and R2. These implementations are shown in figure-12 and figure-13.

.

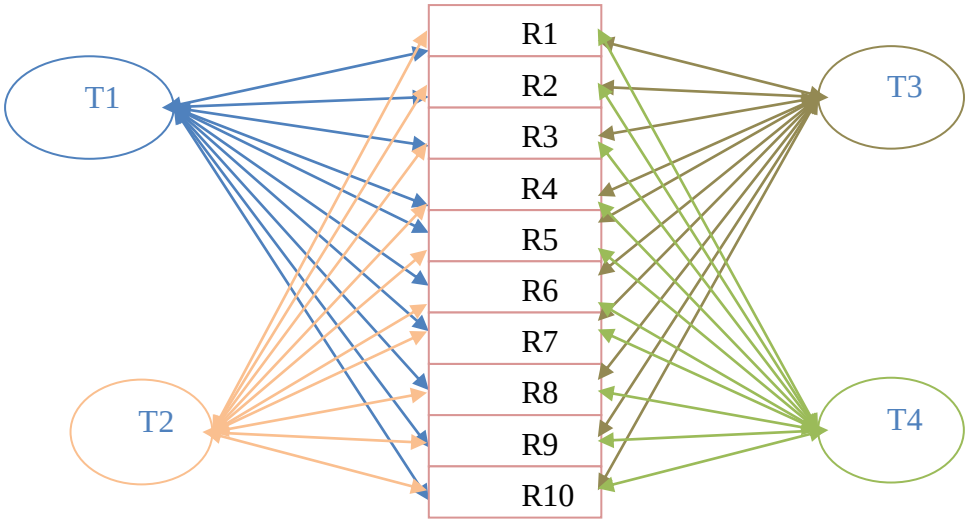


Figure-12: Static EACO scheduling (ET_{ij}) with 40 Iterations

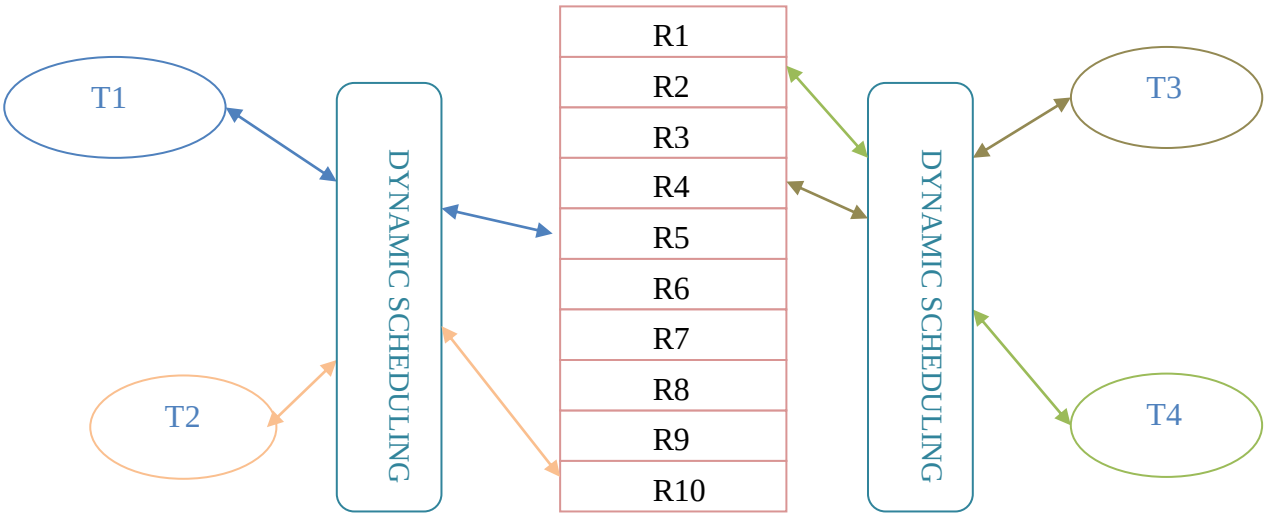


Figure-13: Dynamic EACO scheduling with 4 Iterations

TASK	RESOURCE	Dynamic Job Scheduling (EACO)		Job Scheduling (ET_{ij} Matrix)					
				MACO		MAXMIN-ACO		RASA-ACO	
		RT	CT	RT	CT	RT	CT	RT	CT
T1	R5	0.24	12.24	10.62	68.24	10.62	44.26	10.62	38.24
T2	R10	0.28	12.88	66.66	128.2	66.66	78.44	66.66	62.54
T3	R4	0.34	13.02	86.12	130.24	86.12	88.86	86.12	72.56
T4	R2	0.26	12.64	98.36	146.25	98.36	92.42	98.36	88.65

Table 1: EACO Grid Computing Scheduling compare with ET_{ij} Scheduling (MACO, MAXMIN-ACO and RASA-ACO)

* RT-Ready Time * CT –Completion Time

The proposed EACO scheduling algorithm achieves ready time within 0.34 MIPS to allocate 4 jobs within 10 resources. When using ET_{ij} matrix environment MACO, MAXMIN-ACO and RASA-ACO will have more time. For instance first job (task1) allocate in Resource 5 means, it should travel T1 with R1, R2, R3 and R4 (4 iterations) after that only can reach R5. So first allocation for dynamic ready time is 0.24 and static ready time is 10.62. Second iteration happens after 15 iterations (T2 to R10) so the ready time are 66.66 but for dynamic utilize 0.28 MIPS only. Due to this type of reasons the completion time also maximize. In ET_{ij} matrix environment maximum completion time for MACO is (146.25), MAXMIN-ACO (92.42) and RASA-ACO (88.65). The dynamic EACO maximum completion time is 13.02 MIPS. So, dynamic EACO complete all the necessary need in a very minimum completion time compared with MACO, MAXMIN-ACO and RASA-ACO. Figure-14 shows 4 iteration completion process. Figure-15 shows Comparative Completion Time of MACO,MAXMIN-ACO,RASA-ACO and dynamic EACO.

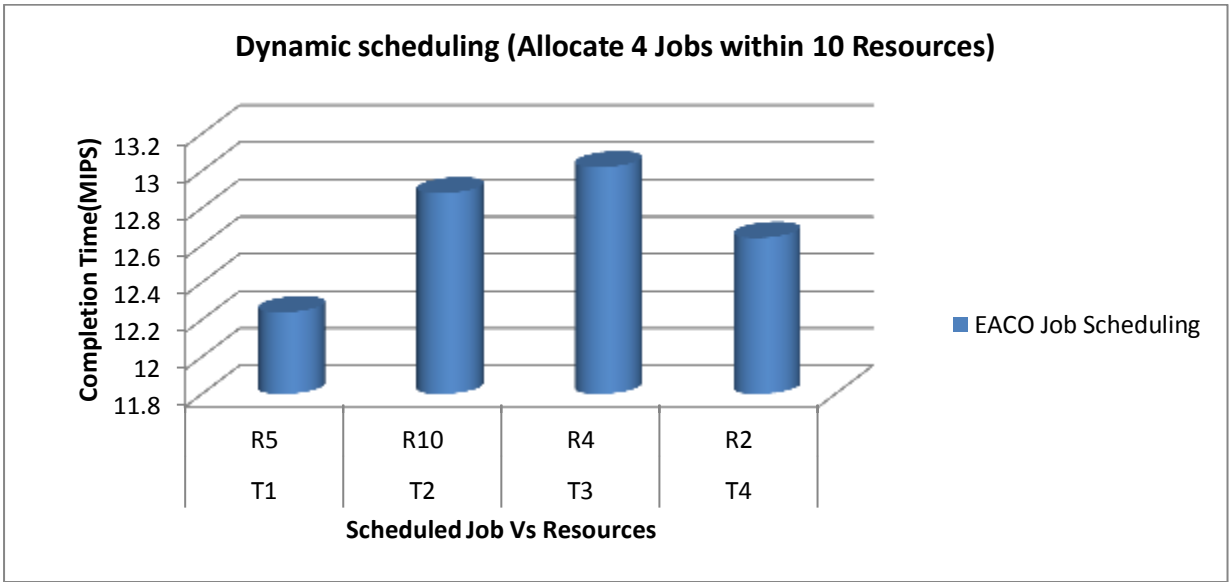


Figure-14: EACO Job Scheduling within 10 resources

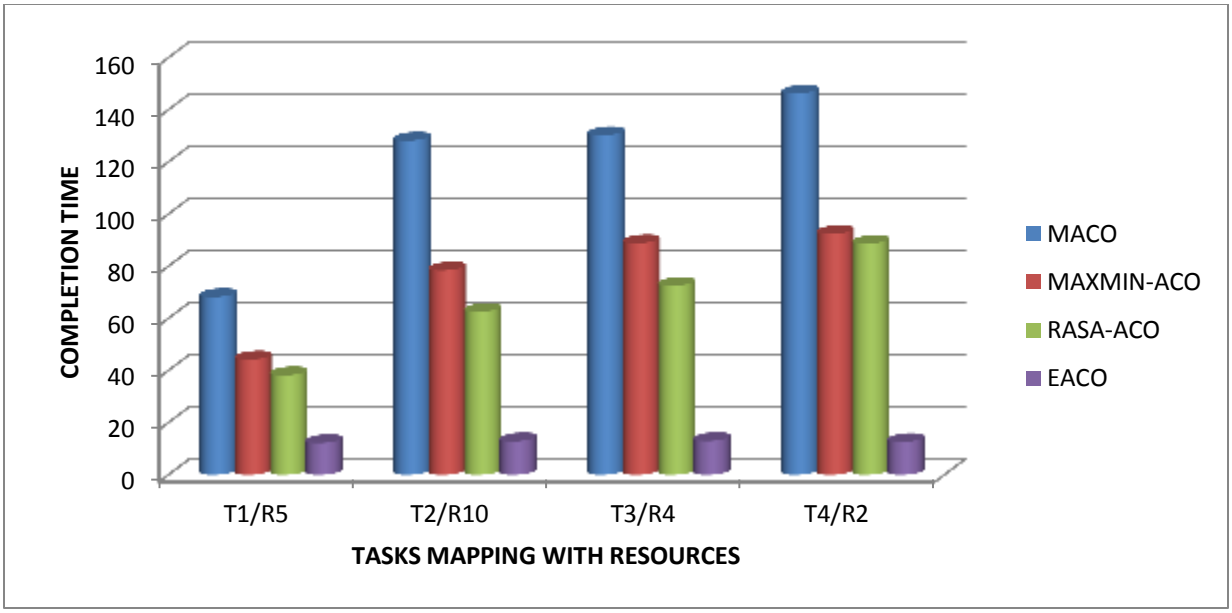


Figure-15: Comparison of Completion Time for MACO,MAXMIN-ACO,RASA-ACO and EACO.

4. Conclusion and Future Work

4.1. Conclusion

In this work, the proposed method EACO has achieved the minimum makespan and completion time. The drawbacks of existing methods (MACO, MAXMIN-ACO and RASA-ACO) were solved by considering some efficient factors in a task scheduling operation. Chosen jobs were allocated to the best selected resources in each iteration. This process is repeated until all jobs have been scheduled and a complete solution has been built. This proposed technique can find an optimal processor and network for each machine to allocate a job that minimizes the makespan time of a job when the job is scheduled. The proposed scheduling algorithm is designed to achieve high throughput for computing in a grid environment.

In this paper, an EACO was proposed to allocate the dynamic jobs to the best resources in grid environment. Thus, the proposed method has achieved high performance in allocating the arrival of dynamic tasks to the accurate resources as well as attained a high efficiency. The performance of the proposed task scheduling technique was analyzed with three mixture techniques namely, MACO, MAXMIN with ACO and RASA with ACO. The experimental results prove that the proposed task scheduling method has attained high precision and efficiency than the three hybrid methods through MACO, MAXMIN-ACO and RASA-ACO. In ACO optimization algorithm, a pheromone trail value is utilized to find out the optimal resource. In MAXMIN algorithm, a maximum completion time value is used for mapping with the best resource. In RASA algorithm, maximum earliest completion time as well as minimum earliest completion time values are utilized for avoiding starvation of the tasks in the available optimal resources in grid environment. Hence the proposed Enhanced ACO task scheduling algorithm is capable of finding the optimal jobs to the optimal resources as well as achieving the minimum makespan and completion time.

4.2. Future work

Future work may cover the grid reservation problem while performing the scheduling operations. For instance, the grid resources can be reserved in advance for a designated set of jobs. Such reservations operate much like an appointed time system used to reserve seminar rooms for meetings. When the appointment time starts, resource is automatically reserved. This resource will be busy during the scheduling period. Due to this reason job will not allocate properly. Since reservation has to deal complete utilization in the grid environment. Further enhancements of our proposed algorithm consider the above factors and will be solved very precisely in dynamic scheduling.

References

[1] Deneubourg JL, Aron S, Goss S,et al.Theself "organizing exploratory pattern of the Argentine ant". J Insect Behav 1990;3(2):159–168.
[2] Marco Dorigo, Mauro Birattari, and Thomas St¨ utzle Universit´ eLibre de ruxelles, BELGIUM, "Ant Colony Optimization Artificial Ants as a Computational Intelligence Technique".
[3] Goss S, Aron S, Deneubourg JL,et al.Self "organized shortcuts in the Argentine ant". Naturwissenschaften 1989;76:579–581.
[4] E. Tsiakkouri et al., "Scheduling Workflows with Budget Constraints", In the CoreGRID Workshop on Integrated research in Grid Computing, S. Gorlatch and M. Danelutto (Eds.), Technical Report TR-05-22, University of Pisa, Dipartimento Di Informatica, Pisa, Italy, Nov. 28-30, 2005, pages347-357.
[5] K. Kousalya and P. Balasubramanie " To Improve Ant Algorithm's Grid Scheduling Using Local Search" International Journal of Computational Cognition (http://www.ijcc.us), vol. 7, no. 4, December 2009.
[6]. R. F. Freund, M. Gherrity, S. Ambrosius, et al., "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet", in 7th IEEE Heterogeneous Computing Workshop (HCW '98), pp. 184_199, 1998.
[7] Stutzle T., Hoos H. H.: "MAX–MIN Ant System". Future Generation Computer Systems 16(8):889–914 (2000).
[8]. Saeed Parsa and Reza Entezari-Maleki." RASA: A New Task Scheduling Algorithm in Grid Environment" Parallel Processing and Concurrent Systems Laboratory, Department of Computer Engineering, Iran University of Science and Technology (IUST), Tehran, Iran
[9] D.Maruthanayagam and Dr.R.Umarani, "(IJARCET) A Novel Approach to Scheduling in Grid Environment: Enhanced Ant Colony Optimizer" on March in the "International Journal of Advanced Research in Engineering & Computer Technology" Volume 2, Issue 3, March 2013, pp 904-915. (ISSN: 2278 – 1323).

Authors Profile

Dr.D.Maruthanayagam received his Ph.D Degree from Manonmanium Sundaranar University, Tirunelveli in the year 2014. He has 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 is working as a Head, Department of Computer Science, Siri PSG Arts & Science College for Women, Sankari, Salem, Tamilnadu, India. He has above 12 years of experience in academic field. He has published 1 book, 8 International Journal papers and 12 papers in National and International Conferences. His areas of interest include Grid Computing, Cloud Computing and Mobile Computing