Balanced Ant Colony Algorithm for Scheduling DAG to Grid Heterogeneous System

Mrs. Smitha Jha

Abstract- Ant Colony Optimization can be used for scheduling tasks on resources in Grid. In earlier work this technique has been applied for independent task scheduling. This paper applies the above technique for dependent task scheduling. Here a hybrid algorithm by Sakellariou can be applied, where tasks in DAG (Directed acyclic graph) are upward ranked and sorted decreasingly. Then the sorted tasks are grouped along the sorted sequences and in every group, tasks are independent. Then independent task groups can be scheduled to resources using algorithm specified by author Chang.

Index Terms – Grid System, Directed Acyclic graph(DAG),Independent, Dependent task scheduling, Ant Colony Optimization, Grouping, Ranking.

1 INTRODUCTION

Current scientific problems are very complex and need huge computing power and storage space. The past technologies such as distributed or parallel computing are unsuitable for current scientific problems with large amounts of data. Processing and storing massive volumes of data may take a very long time. Grid computing [3] is a new paradigm for solving those complex problems. In grids, we need to consider the conditions such as network status and resources status. If the network or resources are unstable, jobs would be failed or the total computation time would be very large. So we need an efficient job scheduling algorithm for these problems in the grid environment. Because the environment status may change frequently, traditional job scheduling algorithm such as ‘‘First Come First Serve’’ (FCFS), ‘‘Shortest Job First’’ (SJF), etc., may not be suitable for the dynamic environment in grids.

- *Mrs Smitha Jha is currently pursuing Ph.D degree program in Computer Science from BIRLA INSTITUTE OF TECHNOLOGY,EXT. CENTER NOIDA,UP,INDIA, PH-09953100878. E-mail: s,jha@bitmesra.ac.in*

In grids, users may face hundreds of thousands of computers to utilize. It is impossible for anyone to manually assign jobs to computing resources in grids. Therefore, grid job scheduling is a very important issue in grid computing.
Please refer to a survey [3], which also poses some open issues. A good schedule would adjust its scheduling strategy according to the changing status of the entire environment and the types of jobs. Therefore, a dynamic algorithm in job scheduling such as Ant Colony Optimization (ACO) [4,5] is appropriate for grids. ACO is a heuristic algorithm with efficient local search for combinatorial problems. ACO imitates the behavior of real ant colonies in nature to search for food and to connect to each other by pheromone laid on paths traveled. Many researches use ACO to solve NP-hard problems such as traveling salesman problem [6],graph coloring problem [7], vehicle routing problem [8], andso on. In [1] this technique has been applied for independent task scheduling. This paper apply the above technique for dependent task scheduling. Here a hybrid algorithm by Sakellariou[2] can be applied,where tasks in DAG(Directed acyclic graph) are upward ranked and sorted decreasingly. Then the sorted tasks are grouped along the sorted sequences and in every group,tasks are independent. We assume each job is an ant and the algorithm sends the ants to search for resources. We also modify the global and local pheromone update functions in ACO algorithm in order to balance the load for each grid resource.
The rest of the paper is organized as follows. Section 2 introduces the related work about many kinds of ACO algorithm and job scheduling in grids. Section 3 details the proposed ACO optimization algorithm, its pseudocode and

algorithm analysis in job scheduling for dependent task scheduling. Section 4 gives the comparison of proposed algorithm with other existing methods in terms of time complexity. Section 5 concludes paper and Section 6 lists references.

2.RELATED WORK:

2.1. Ant algorithms

There are many different kinds of ACO algorithm, i.e., AntColony System (ACS) [6], Max-Min Ant System (MMAS) [9],Rank-based Ant System (RAS) [10], Fast Ant System (FANT) [11] and Elitist Ant System (EAS) [12]. ACS uses the pseudo-randomproportional rule to replace state transition rule for decreasing computation time of selecting paths and update the pheromone on the optimal path only. It is proved that it helps ants search the optimal path.MMAS is based on the basic ACO algorithm but limiting the pheromone range to be greater than or equal to the low bound value (Min) and smaller than or equal to the upper bound value(Max). The low bound and upper bound are defined by the user.According to the low bound and upper bound values, MMAS could avoid ants to converge too soon in some ranges.

In the design of RAS, it sorts the ants by ant's tour length in ascending order after all ants completed their tours. It means that the first ant finds the shortest path to complete the tour and the last ant takes the longest tour. They give each ant a different density of pheromone to update their path by the ascending order: the higher the position of the ant, the more pheromone it could update; the lower the position of the ant, the less pheromone it has. By the idea of RAS, the shortest length gets more pheromone to attract more ants to follow and the system could get the optimal solution very soon, and it updates pheromone after each iteration. In order to avoid the sub-optimal solution, it applies a reset pheromone function.EAS update more pheromone on the best-so-far tour found inorder to attract more ants to follow the best-so-far tour.

2.2 Job scheduling in grids

Job scheduling is well studied within the computer operating systems [13]. Most of them can be applied to the grid environment with suitable modifications. In the following we introduce several methods for grids. The FPLTF (Fastest Processor to Largest Task First) [14] algorithm schedules tasks to resources according to the workload of tasks in the grid system. The algorithm needs two main parameters such as the CPU speed of resources and workload of tasks. The scheduler sorts the tasks and resources by their workload and CPU speed then assigns the largest task to the fastest available resource. If there are many tasks with heavy workload, its performance may be very bad. Dynamic FPLTF (DPLTF) [15] is based on the static FPLTF, it gives the highest priority to the largest task.DPLTF needs prediction information on processor speeds and taskworkload. The WQR (Work Queue with Replication) is based on the workqueue (WQ) algorithm [15]. The WQR sets a faster processor with more tasks than a slower processor and it applies FCFS and random transfer to assign resources. WQR replicates tasks in order to transfer to available resources. The amount of replications is defined by the user. When one of the replication tasks is finished,the scheduler will cancel the remaining replication tasks. The WQR's shortcoming is that it takes too much time to execute and transfer replication tasks to resource for execution.

Min-min [16] set the tasks which can be completed earliest with the highest priority. The main idea of Min-min is that it assigns tasks to resources which can execute tasks the fastest. Maxmin[16] set the tasks which has the maximum earliest completion time with the highest priority. The main idea of Max-min is that it overlaps the tasks with long running time with the tasks with short running time.

For instance, if there is only one long task, Min-min will execute short tasks in parallel and then execute long task. Max-min will execute short tasks and long task in parallel.The RR (Round Robin) algorithm focuses on the fairness problem. RR uses the ring as its queue to store jobs. Each job in queue has the same execution time and it will be executed in turn.If a job can't be completed during its turn, it will store back to the queue waiting for the next turn. The advantage of RR algorithm is that each job will be executed in turn and they don't have to wait for the previous one to complete. But if the load is heavy, RR will take long time to complete all jobs.

Priority scheduling algorithm gives each job a priority value and uses it to dispatch jobs. The priority value of each job depends on the job status such as the requirement of memory sizes, CPU time and so on. The main problem of this algorithm is that it may cause indefinite blocking or starvation if the requirement of a job is never being satisfied.The FCFS (First Come First Serve) algorithm is a simple job scheduling algorithm. A job which makes the first requirement will be executed first. The main problem of FCFS is its convoy effect [13].If all jobs are waiting for a big job to finish, the convoy effect occurs.The convoy effect may lead to longer average waiting time and lower resource utilization.

3 BALANCED ANT COLONY ALGORITHM FOR DEPENDENT TASK SCHEDULING

3.1 ARCHITECTURE OF THE SCHEDULING SYSTEM

Model Description:

Suppose Directed acyclic graph(DAG) given below represents a set of dependent tasks.

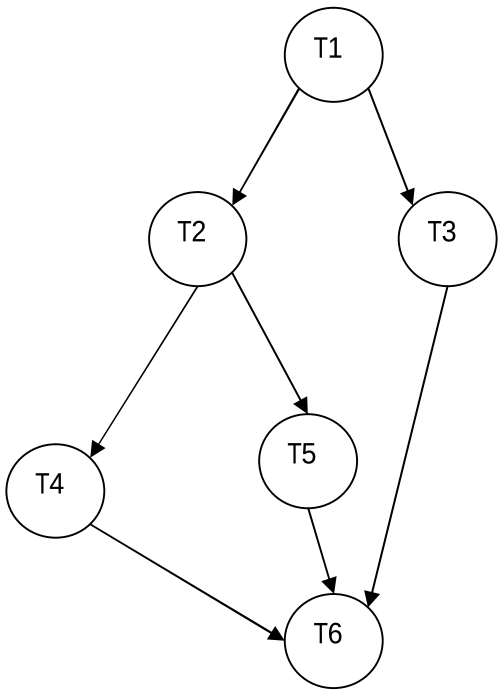
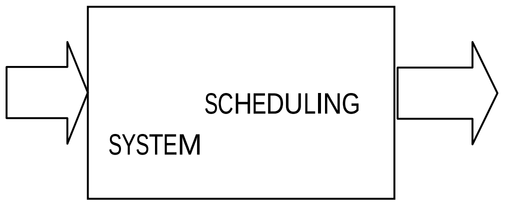


Fig-1(Given Directed Acyclic graph representing set of dependent tasks)

The DAG is divided into n no. of groups containing independent sets of tasks.

$G = G_0 \cup G_1 \cup G_2 ... \cup G_{n-1}$

With the condition that all the tasks in G_{i+1} will be executed after all the tasks in G_i has been executed for all i=0 to n-1.



Input- [$G = G_0 \cup G_1 \cup G_2 ... \cup G_{n-1}$]

Output- MACHINES(A cluster of Workstations)

Fig-2 Scheduling Architecture

This can be solved by taking a cluster of workstations connected by internet as shown in Fig-3

$G_0, G_1, G_2 ... G_{n-1}$ can be applied to these workstations with a condition that there should be synchronization among them,so that G_{i+1} group uses the result of tasks completed in G_i group

$\forall i = 0 \, to \, n-1$.Each group G_i tasks can be executed using Balanced Ant Colony Optimization.

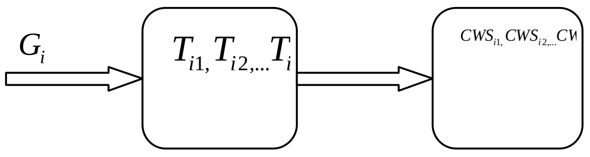


Fig-4 Assigning tasks to cluster of workstations.

Here each of the task is modelled as an ant and each of the machine in the cluster of workstation is the food . Each ant select food through shortest path i.e. where the pheromone density is more.

The balanced antcolony algorithm for scheduling tasks to resources is described in the next section.

3.1 Proposed algorithm:

In this section we are presenting the algorithm for the dependent task scheduling in Grid heterogeneous system using Ant Colony Optimization technique.

Step 1: Input the Directed acyclic graph(DAG) G depicting how different tasks are dependent to each other.[In the graph each node representing task node and the edges between them represents the dependency among them]

Step 2: Rank the nodes in the graph using Ranking() method.

Step 3:Using GroupCreation() method ,a set of groups of independent tasks are made.

Step 4:for Groupno=1;Groupno<n;Groupno++ [for n no. of groups of independent tasks]

Apply AntColony(Groupno.) method on each group.

Ranking() method algorithm

{

1. Assign a weight to each node as the average computation cost across all machines.
2. Assign a weight to each edge as the average communication cost across all machines.

3. Use upward ranking to compute the rank value for each node, The rank value Ri of a node I is recursively defined as follows

$R_i = W_i + \max(C_{ij} + R_j)$ for all j present in S_i

Where

Wi=weight of the node.

Si=the set of immediate successor of node i.

Cij=Weight of the edge connecting nodes i and j.

}

GroupCreation() method algorithm

```
{
1.Sort nodes in descending order of their rank values.
    G0={},i=0;
2.Scan nodes in descending order of the rank values
If current node has a dependence with a node Gi
        Then i++; and Gi={}; Add node to Gi
3.Repeat step 2 until there are no more nodes.
}
```

SchedulingAntColony(Group no.) algorithm

{

[An Ant in the Ant system is a job in the Grid system. Pheromone value on a path in the Ant system is a weight for a resource in the Grid system. A resource with a large weight value means that the resource has a better computing power. The scheduler collects data from information server and uses the data to calculate a weight value of the resource. The pheromone(weight) of each resource is stored in the scheduler and the scheduler uses it as the parameter for the Balanced Ant Colony Optimization algorithm. At last the selects the resource according to the algorithm .]

The pheromone indicator(PI)(the initial pheromone value of each job to the resource)

$$PI_{ij} = \left[\frac{M_j}{BW_i} + \frac{T_j}{CS*(1-LD_i)} \right] \qquad (1)$$

Where

M_j =Size of a given job.

BW_i =Network Bandwidth between scheduler and the resource i.

T_j =CPU time needed for the job j.

LD_i =Current load at resource i.

CS=CPU speed of resource i.

The load,Bandwidth and CPU speed can be obtained by NWS(Network Weather Service).

1. Do steps 2 to 4 until all the tasks are assigned
2. The resource status(How many jobs are there in resource, i. e. how much is it busy,the program execution time and the size of jobs) is told by PI. The larger the value of PI_{ij} , the more efficient it is to assign job j to the resource i.
3. The PI matrix is as follows for m resources and n jobs.

$$\begin{bmatrix} P_{11} & P_{12} & \dots & PI_{1n} \\ P_{21} & P_{22} & \dots & P_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m1} & P_{m2} & \cdots & P_{mn} \end{bmatrix} \qquad (2)$$

The largest entry from the PI matrix is chosen say P_{ij} ,then the jth job assigned to the resource ith. After this equation (2) is updated using equation (1) for m resources and (n-1) jobs. This is called local pheromone update.

4. The global pheromone update is to recalculate the entire PI matrix,after the resource completes the job.

3.2 Pseudocode of the proposed algorithm

In this section we are presenting the pseudocode for the dependent task scheduling using Ant Colony Optimization technique.

```
Procedure SchedulingDependenttask_Antcolony(Graph G)
{
Procedure MakingIndependent taskgroup();
For(i=1;i<groupno.;i++){
Procedure SchedulingAntcolony(groupno.);
}
}
Procedure  Makingindependent taskgroup()
(
Procedure Ranking();
Procedure GroupCreation();
}
Procedure Ranking()
{
1.Assign a weight to each node as the average computation cost across all machines.
2.Assign a weight to each edge as the average communication cost across all machines.
3.Use upward ranking to compute the rank value for each node.}
Procedure GroupCreation()
{
```

```
1.Sort nodes in descending order of their rank values.
    G0={},i=0;
2.Scan nodes in descending order of the rank values
If current node has a dependence with a node Gi
        Then i++; and Gi={}; Add node to Gi
    3.Repeat step 2 until there are no more nodes.
}
```

The rank value Ri of a node I is recursively defined as follows

Ri= Wi + max(Cij +Rj) for all j present in Si

Where

Wi=weight of the node.

Si=the set of immediate successor of node i.

Cij=Weight of the edge connecting nodes i and j.

Procedure SchedulingAntColony(Group no.)

{

[An Ant in the Ant system is a job in the Grid system. Pheromone value on a path in the Ant system is a weight for a resource in the Grid system. A resource with a large weight value means that the resource has a better computing power. The scheduler collects data from information server and uses the data to calculate a weight value of the resource. The pheromone(weight) of each resource is stored in the scheduler and the scheduler uses it as the parameter for the Balanced Ant Colony Optimization algorithm. At last the selects the resource according to the algorithm .]

The pheromone indicator(PI)(the initial pheromone value of each job to the resource)

$$PI_{ij} = \left[\frac{M_j}{BW_i} + \frac{T_j}{CS * (1 - LD_i)} \right] \qquad (1)$$

Where

M_j =Size of a given job.

BW_i =Network Bandwidth between scheduler and the resource i.

T_j =CPU time needed for the job j.

LD_i =Current load at resource i.

CS=CPU speed of resource i.

The load,Bandwidth and CPU speed can be obtained by NWS(Network Weather Service).

No_of_tasks_assigned=0;

Do
{
1.The resource status(How many jobs are there in resource, i. e. how much is it busy,the program execution time and the size of jobs) is told by PI. The larger the value of PI_{ij}, the more efficient it is to assign job j to the resource i.
2.The PI matrix is as follows for m resources and n jobs.

$$\begin{bmatrix} P_{11} & P_{12} & \dots & PI_{1n} \\ P_{21} & P_{22} & \dots & P_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m1} & P_{m2} & \cdots & P_{mn} \end{bmatrix} \qquad (2)$$

The largest entry from the PI matrix is chosen say P_{ij} ,then the jth job assigned to the resource ith. After this equation (2) is updated using equation (1) for m resources and (n-1) jobs. This is called local pheromone update.
3.The global pheromone update is to recalculate the entire PI matrix,after the resource completes the job.
No_of_tasks_assigned++;

```
}while(no_of_ tasks _assigned<totalno_tasks)
```

3.3 Algorithm analysis

Let T represent the Time complexity of the proposed algorithm.

T =T(Ranking)+T(Grouping)+T(ACO)

Where ACO is Ant Colony Optimization algorithm.

T(Ranking)=O(logn)
T(Grouping)=O(nlogn)[Tasks are sorted using heap sort)
T(ACO)= $o(n^2P)$

4 COMPARISON WITH OTHER METHODS:

There exist other clustering heuristics for dependent task scheduling[3].Dominant Sequence Clustering(DSC) algorithm proposed by Yang et al[4] has time complexity for a DAG as $O((e+n)\log n)$ which is equal to $O(n^2 \log n)$ for a dense graph.. Similarly CLASS-II,a cluster algorithm proposed by Liou et al [5] has time complexity as $O(e+n\log n)$ which is equal to $O(n^2+n\log n)$ for a dense graph. The fig-5 depicts a comparison of these methods with respect to no. of tasks to be assigned.

5 CONCLUSION

From above algorithm analysis it is concluded that ACO gives better performance with respect to DSC and CLASS II clustering heuristic methods for a fixed no. of processors. In this paper pseudocode of the proposed algorithm is written which has to be coded in C in next paper.

6 REFERENCES

[1] An ant algorithm for balanced job scheduling in grids Ruay-Shiung Chang_, Jih-Sheng Chang, Po-Sheng Lin Department of Computer Science and Information Engineering, National Dong Hwa University, Shoufeng Hualien, 974 Taiwan, ROC

[2].A Hybrid heuristic for DAG Scheduling on Heteorogeneous Systems. Rizos Sakellario and Henan Zhao ,Department of Computer Science,University of Manchester,Oxford Road,Manchester M13 9PL,U.K.

[3]. Fangpeng Dong and Selim G. Akl "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems" Technical Report No. 2006-504

[4]. M. Dorigo, C. Blum, Ant colony optimization theory: A survey, Theoretical Computer Science 344 (2–3) (2005) 243–278.

[5]. M. Dorigo, Ant colony optimization, http://www.aco-metaheuristic.org.

[6]. M. Dorigo, L.M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.

[7]. E. Salari, K. Eshghi, An ACO algorithm for graph coloring problem, in: Congress on Computational Intelligence Methods and Applications, 15–17 Dec. 2005, p. 5.

[8]. Xiaoxia Zhang, Lixin Tang, CT-ACO—hybridizing ant colony optimization with cycle transfer search for the vehicle routing problem, in: Congress on Computational Intelligence Methods and Applications, 15–17 Dec. 2005, p. 6.

[9]. T. Stutzle, MAX-MIN Ant System for Quadratic Assignment Problems Technical Report AIDA-97-04, Intellectics Group, Department of Compute Science, Darmstadt University of Technology, Germany, July 1997.

[10]. B. Bullnheimer, R.F. Hartl, C. Strauss, A new rank-based version of the ant system: A computational study, Central European Journal for Operations Research and Economics 7 (1) (1999) 25–38.

[11]. E.D. Taillard, L.M. Gambardella, Adaptive memories for the quadratic assignment
problem, Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland,
1997.
[12]. M. Dorigo, V. Maniezzo, A. Colorni, The ant system: Optimization by a colony of
cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part
B 26 (1) (1996) 29–41.

[13]. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Operating System
Concepts, 7th edition, John Wiley & Sons, 2005.

[14]. D. Saha, D. Menasce, S. Porto, Static and dynamic processor scheduling
disciplines in heterogeneous parallel architectures, Journal of Parallel and
Distributed Computing 28 (1) (1995) 1–18.

[15]. D. Paranhos, W. Cirne, F. Brasileiro, Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids, in: International Conference on Parallel and Distributed Computing (Euro-
Par), in: Lecture Notes in Computer Science, vol. 2790, 2003, pp. 169–180.

[16]. M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R. Freund, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing
system, Journal of Parallel and Distributed Computing 59 (1999) 107–131

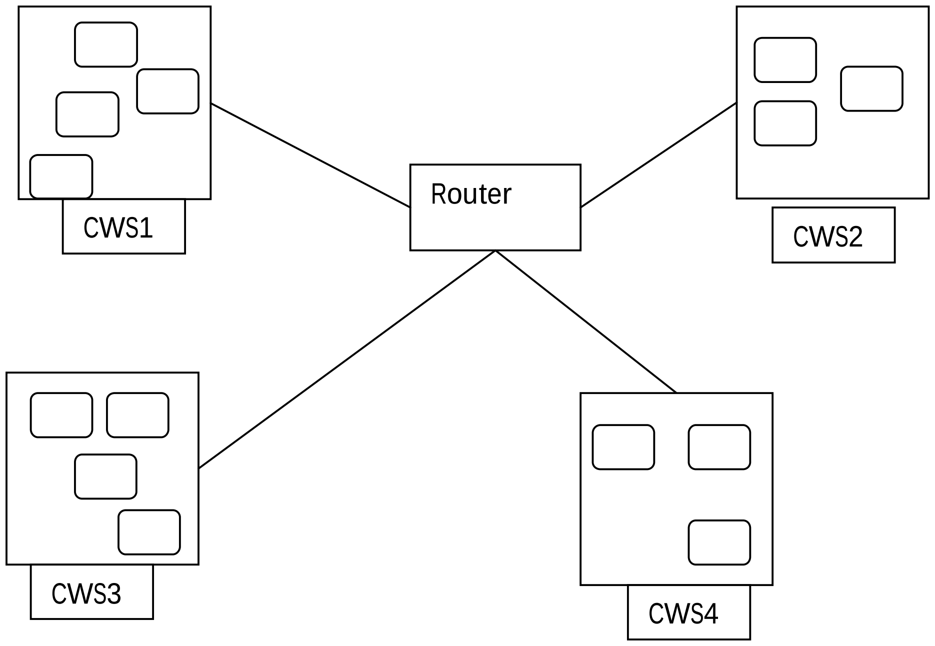
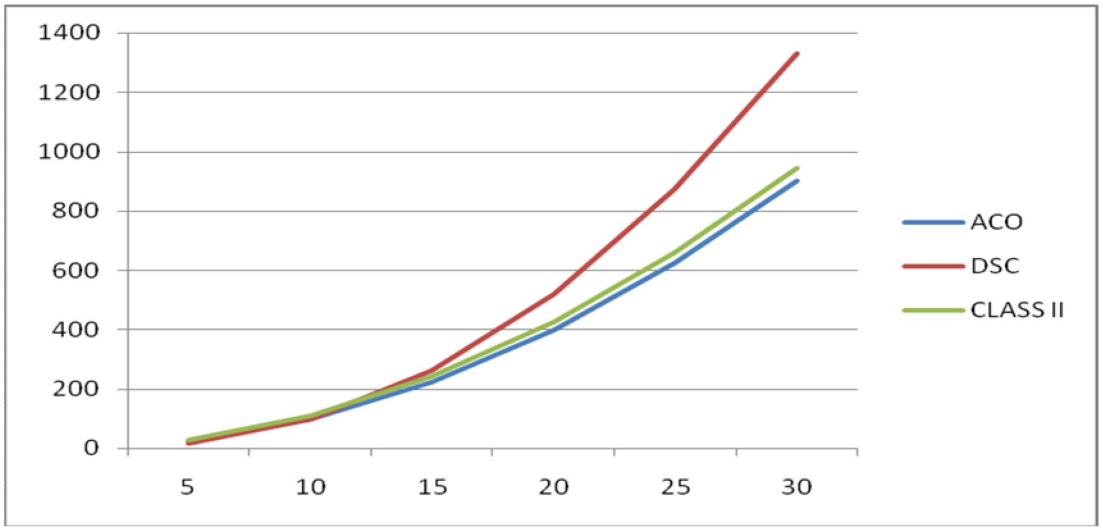


Figure-3



X AXIS-NO. OF TASK

Y AXIS-TIMECOMPLEXITY

Fig-5