Dynamic Scheduling in Distributed System using Ant Colony Optimization

A Project Report for course CSC502
“Principles of OS & Distributed Systems”
Fall 2016

Under the guidance of

Dr.Ming-Hwa Wang

Submitted by :
Anu Mehndiratta
Dipali Suryawanshi
Sumitha Payyan Keloth

ACKNOWLEDGEMENT

We would like to express our heartfelt appreciation and sincere gratitude to our Professor, Dr.Ming-Hwa Wang for giving us this opportunity and encouraging our research. We would also like to thank International Technology University for providing us with resources necessary for the research.

Lastly, we would like to thank our family and friends for their continued support through the course of our research.

Table of Contents

List of Figures

List of Tables

1 Abstract

With the development in technologies and large resource intensive applications, a large scale distributed system has emerged as popular platforms. Distributed system involves integration of large number of resources which may be geographically dispersed. Hence, proper techniques are required for scheduling the job and balanced load distribution among recourses. Scheduling of tasks in distributed system involves deciding not only when to execute a process, but also where to execute it. A proper task scheduling will enhance the resource utilization, reduces execution time and increases system throughput. In this paper, a heuristic approach using an Ant colony optimization algorithm is proposed. In the natural environment, the ants have a tremendous ability to team up to find an optimal path to food resources. An ant algorithm simulates the behavior of ants. The overall objective of the proposed Ant Based Approach is to minimize Make Span while maximizing CPU utilization by distributing workload equally among the available resources. This research compares the proposed approach with the Random approach on the basis of Make span and the utilization of resources in the system.

2 Introduction

2.1 Objective

Task scheduling is a very crucial process in distributed computing environment. This problem is NP-complete and many algorithms have been proposed to achieve optimum performance in distributed systems. Ant Colony Algorithm is very promising in the field of task scheduling in distributed system. ACO is based on ant colony's foraging behaviors, ants can often find the best route between the food source and their nest. It employs simple agents called ants and these agents generate an artificial pheromone trail. We are implementing ant colony optimization algorithm that schedules the tasks from task pool using ant system so as to minimize makespan and maximize CPU utilization. The tasks will be selected according to its heuristic value and pheromone trail and then the selected task is assigned to the best processor.

As an enhancement to the existing algorithm, we are implementing the feature that addresses missed /failed tasks. Any task that is missed will be rescheduled on priority. Also, any failed task will be reallocated to the best processor. Also, the makespan and CPU utilization of scheduling using ant system will be compared to previous task scheduling algorithms such as First cum first serve, Round robin scheduling etc.

The aim of our project is also to balance the load of entire grid system so as to complete the entire assigned task as soon as possible and in a very feasible manner.

2.2 What is the Problem

The primary concern in distributed system for dynamic scheduling is deadline compliance for the set of tasks. Finding the optimal schedule for tasks has been shown to NP-complete. Not all systems can afford to solve such a complex problem. Instead, many systems implement a heuristic scheduler that, rather than guarantying full deadline compliance, either attempts to minimize the deadline-miss rate or introduces some amount of laxity for each deadline (or both).

Efficient resource management in Distributed systems is not easy because distributed computing environment consists of several personal computers or workstations that are combined through local networks in order to develop distributed applications. To make the distributed system more effective, number of tasks which is being allocated to the system should be scheduled efficiently so that it should execute maximum number of task in minimum time span. However, applications are difficult to be scheduled effectively in distributed computing because whenever load increase the performance of the system decreases gradually. Recent research has proved that EDF (Earliest Deadline First) and LST (Least Slack Time First) scheduling algorithms are not optimal in the situation where system becomes slightly overloaded.

Several characteristics make ACO a unique approach: it is constructive, population-based metaheuristic which can perform efficiently and effectively in terms of minimizing total tardiness time. Not only does it improve the overall performance of the system but it also adapts to the dynamic grid system.

2.3 Why is this project related this course?

In a distributed system multiple clients requesting for the available resources on the server or request to do some set of task. In order to make the effective utilization of all the resources or to execute number of task allocated in minimum time span in distributed system, it is very much required to schedule the task in distributed system. An effective distributed system is considered as efficient system when it performs optimal in loaded condition as well. Scheduling of jobs or tasks in distributed system enables the selection of best suitable resources for task execution.

Additionally, timely detection of imbalance of host loads, can amend SLA (Service Level Agreement) violations and prevent the loss of network bandwidth. Due to the complexities in the distributed systems, task scheduling is being researched by many researchers.

2.4 Why other approaches are not good

Most of the approaches for dynamic task scheduling in distributed system focus only on assigning the jobs based on the priority but it does not focus on finding the optimal path and execution time. These dynamic scheduling algorithms do not consider the scenario of how many tasks have missed the deadline in loaded conditions and hence result in poor performance and low efficiency in resource utilization.

2.5 Why you think your approach is better

Our approach involves in identification of tasks for which scheduler has missed the deadline and also it dynamically maps the set of tasks to best processor which can execute the task efficiently within a minimum time span using optimal path. Additionally, the approach which we have used to schedule the tasks dynamically, is a more ecologically inspired approach which we can simulate using the ant's natural behavior. Therefore, our approach is better and results in high resource utilization unlike other approaches.

2.6 Statement of the problem

Design a dynamic scheduling algorithm that results in high resource utilization efficiency in minimum time span while meeting the user's requirements. The current scheduling algorithms simply map tasks to processors without considering the load balancing of the host systems. To avoid the overhead involved since meeting user's requirements is of highest priority without violating the SLA (Service Level Agreement) than resource utilization. But inefficient resource utilization can incur significant wastage of power, network bandwidth.

Hence our solution is to dynamic scheduling algorithm that also considers the load of the hosts, avoiding overload so as to meet the user's requirements on time (adhering to SLA, optimal path, minimum time span), and detect jobs which had missed the deadline and reallocate it to the best performing processor to avoid unnecessary loss of performance in distributed system.

2.7 Area or scope of investigation

- Analyze the ants behavior.
- How many ants search the grid resource.
- How thoroughly do all the ants search the grid resource.
- How do the ants behave if the number of grid resource increases.
- The above behavior analysis will be used in allocating task to multiple available processors.
- Identify any missed/failed task and then reallocate it to the best processor.
- At each step check if the resource is over utilized, and take steps to avoid over loading.
- Simulate on GridSim.
- Compare with other existing task scheduling algorithms.

3 Theoretical basis and literature review

3.1 Definition of the problem

In distributed system, task scheduling is very crucial for optimal performance. This projects aims at reducing the makespan and improving the CPU utilization by efficient task scheduling using ant colony algorithm (ACO).

3.2 Theoretical background of the problem

ACO is a heuristic technique for optimization introduced in early 1990's. The inspiring source of ACO is the foraging behavior of real ant colonies. At the core of this behavior is the indirect communication between the ants by means of chemical pheromone trails, which enables them to find short paths between their nest and food sources. This characteristic of real ant colonies is exploited in ACO algorithms [5]. Several special cases of the ACO have been proposed in the literature, the three most successful ones: ant system (Dorigo 1992, Dorigo et al. 1991, 1996), ant colony system (ACS) (Dorigo& Gambardella 1997), and MAX-MIN ant system (MMAS) (Stützle & Hoos 2000).

In the natural world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food. Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads to all the ants' following a single path.

The flowchart of the ACO has been shown in figure (Figure 1).

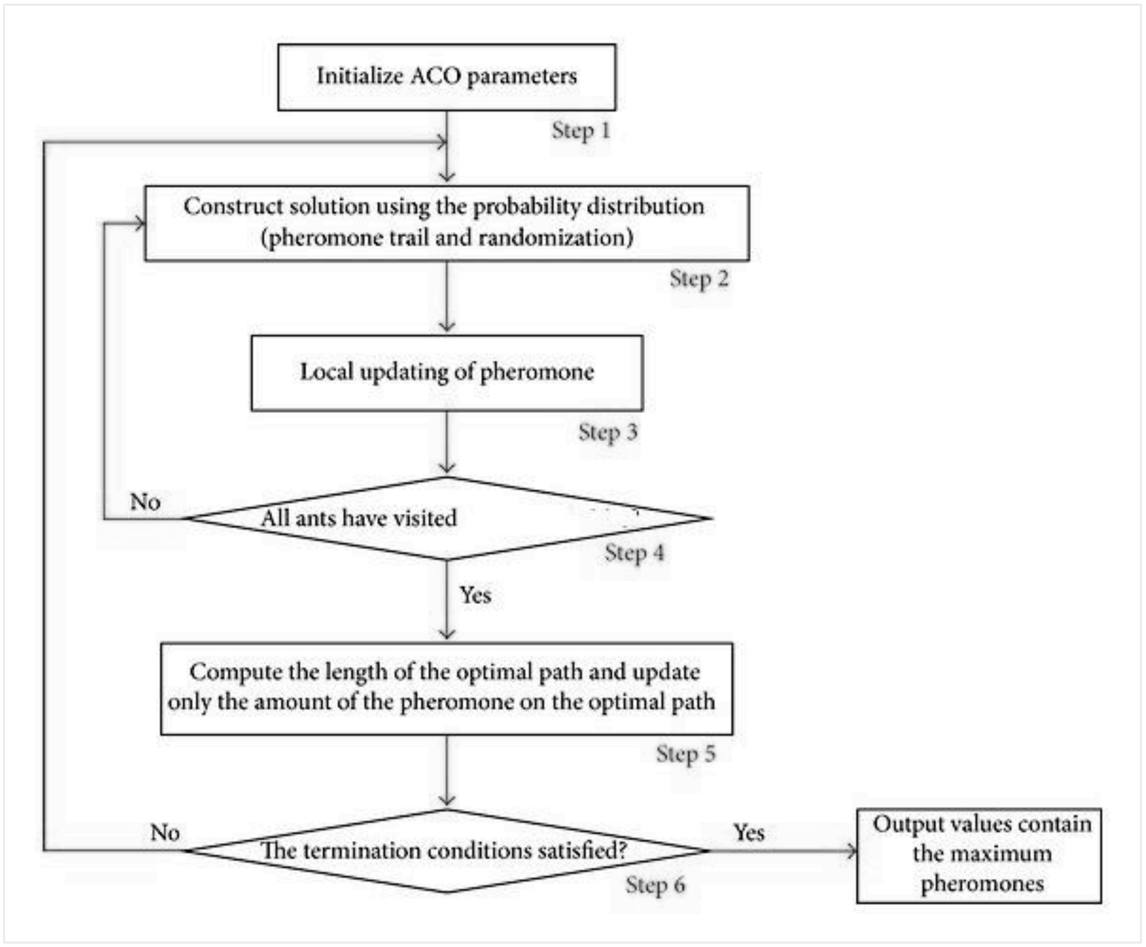

Initialize ACO parameters
Step 1
Construct solution using the probability distribution (pheromone trail and randomization)
Step 2
Local updating of pheromone
Step 3
No
All ants have visited
Step 4
Yes
Compute the length of the optimal path and update only the amount of the pheromone on the optimal path
Step 5
No
The termination conditions satisfied?
Step 6
Yes
Output values contain the maximum pheromones

Figure 1 : ACO Flowchart

Mathematical steps

Let τij be the artificial pheromone trail.

For the kth ant the next task to be scheduled is probabilistically selected according to heuristic value and pheromone trial information.

The selected task is allocated to its best processor (P best(ti).

The probability of task ti to be selected is obtained as follows:

$$probability(t_i) = \frac{[\tau(t_i, p_{best}(t_i))]^{\alpha}.[\eta(t_i)]^{\beta}}{\sum_{\forall unscheduled\ task\ t_k} [\tau(t_k, p_{best}(t_k))]^{\alpha}.[\eta(t_k)]^{\beta}}$$

α, β -> parameters that control relative imp. of trail vs visibility. Default value = 1

η(ti) -> Heuristic desirability

η(ti), Heuristic desirability

$$\eta(t_i) = (\alpha \times SRC(t_i, p_{best}(t_i))) \times (\beta \times DRC(t_i, p_{best}(t_i)))$$

The best suitable processor for a process t i is the processor
P j that maximizes SRC and DRC.
SRC (Static Relative Cost): -

$$SRC(t_i, p_j) = \frac{a_{ij}}{(\sum_{k=1}^{m} a_{ik}) / m}$$

DRC (Dynamic Relative Cost): -

$$DRC(t_i, p_j) = \frac{ct_{ij}}{(\sum_{k=1}^{m} ct_{ik}) / m}$$

The amount of pheromone trail τij represents the desirability of allocating task ti on processor P j in k-th iteration.

After the ants in the algorithm ended their tasks, the pheromone trail τij (ti , Pj) values of every task ti on processor P j are updated according to the following formula:

$$\tau_{k+1}(t_i, p_j) = \begin{cases} \rho \times \tau_k(t_i, p_j) + \Delta & \text{if } t_i \text{ is allocated on} \\ & \text{processor } p_j \text{ in } s^{ib} \\ \rho \times \tau_k(t_i, p_j) & \text{otherwise} \end{cases}$$

ρ – local pheromone decay parameter(0.1≤ρ≤0.99)
Δ – ratio of iteration best sol to global best sol.

Task scheduling in a distributed system can be stated as allocating tasks to processors of each computer such that the optimum performance is obtained. The aim of task scheduling is minimizing makespan while maximizing CPU utilization.

There are two types of scheduling namely static scheduling and dynamic scheduling in grid computing system. Static Scheduling and Dynamic Scheduling. For the static scheduling, jobs are assigned to suitable resources before their execution begin. Once started, they keep running on the same resources without interruption. In dynamic scheduling, jobs are assigned during run time and no knowledge of task is in hand until it arrives.

3.3 Related research to solve the problem

Some of the related research in the field of scheduling are:

"Dynamic Load Balancing Algorithm Based on FCFS", IEEE, (2009) [2] introduces a kind of load balancing algorithm that every node sends a corresponding request stream to remark its real-time load based on FCFS principle. FCFS (First Come First Served), used in parallel task processing, is the simplest task ordering strategy. It chooses and processes them according to the right order of jobs getting into system. Through this algorithm, each node can send assigning requests to the front-end scheduler with their parameters (static parameter and dynamic parameter), which makes the heavy-load node receive less work, the low-load more.
As stated in paper [2] algorithm has a less time complexity, and reduces the computing time of scheduler. The algorithm is feasible and better than the common static algorithm used. From the analysis of the whole algorithm, it is found that if communication among the load nodes and the scheduler is minimized, such as optimizing the length of the circular queue, it would be better to play the superiority of the algorithm. Disadvantage of this algorithm is time consuming and does not perform quite efficiently when there is a case of priority.

"Load Balancing Scheduling with Shortest Load First", International Journal of Grid Distribution Computing Vol. 8, No.4, (2015) : Shortest Job First (SJF) scheduling is a priority and Non-Preemptive scheduling. Non-Preemptive means, when the allotted time a processor then the processor cannot be taken the other, until the process is completed in the execution. Basically Shortest Job First is a dynamic load-balancing algorithm which handles the process with priority basis. It determines the priority by checking the size of the process. [1] Shows that both average waiting time and average Turnaround Time are very less in case of SJF scheduling which helps to improve the performance of the system and maintain an efficient load balancing with very fast manner. But SJF faces some difficulties, such as the burst time of the process have to predict before CPU start the execution.

"Analysis of Different Variants in Round Robin Algorithms for Load Balancing in Cloud Computing" International Journal of Computer Applications (0975 – 8887) Volume 69– No.22, May 2013
Round robin uses the time slicing mechanism. The name of the algorithm suggests that it works in the round manner where each node is allotted with a time slice and has to wait for their turn. The time is divided and interval is allotted to each node. Each node is allotted with a time slice in which they have to perform their task.
In [3] four different scheduling algorithms are simulated along with different variants of round robin algorithm. Each algorithm is observed on their scheduling criteria like average response time. According to the experiment and analysis round robin algorithm

has the best integrate performance. [3] suggest application of some evolutionary algorithm like ACO instead of classical algorithms for better response time.

“Fair Scheduling Algorithm with Dynamic Load Balancing Using In Grid Computing” Research Invent: International Journal Of Engineering And Science Vol.2, Issue 10 (April 2013)
In [4] the tasks are allocated to multiple processors so that the task with unsatisfied demand gets equal shares of time. In this algorithm, tasks with a higher order are completed first which means that tasks are taken a higher priority than the others which leads to starvation that increases the completion time of tasks and load balance is not guaranteed. For this issue proposed Load Balance (LB) Algorithm gives uniform load to the resources so that all tasks are fairly allocated to processor based on balanced fair rates. [4] concludes that algorithm has proved the best results in terms of makespan and Execution Cost. In particular the algorithm allocates the task to the available processors so that all requesting task get equal amount of time that satisfied their demand. Proposed algorithm is definitely a promising tendency to solve high demanding applications and all kinds of problems but grid application performance remains a challenge in dynamic grid environment. Resources can be submitted to Grid and can be withdrawn from Grid at any moment

“A Comparative Study of Scheduling Algorithms for Real Time Task” International Journal of Advances in Science and Technology, Vol. 1, No. 4, 2010
In [8] the priority of each task is decided based on the value of its deadline. The task with nearest deadline is given highest priority and it is selected for execution. The comparison is based on scheduling periodic task on single processor environment and the tasks are pre-emptive. EDF algorithm does not perform well when the system is overloaded and comparison is based on scheduling periodic task on single processor environment and the tasks are pre-emptive. EDF algorithm does not perform well when the system is overloaded. [8] Suggests, in future a new algorithm should be developed to switch automatically between EDF algorithm and ACO based scheduling algorithm to deal overloaded and under loaded conditions

3.4 Advantages and Disadvantages of those research

The above approaches scan the entire solution space without consideration to techniques that can reduce the complexity of the optimization. The major disadvantage is that, they spend much more time scheduling and hence need exhaustive time. Moreover, there is no consideration for even workload distribution. Thus, an effective grid resource management with good task scheduling algorithm is needed to manage the distributed system which also handles failed/missed jobs.

3.5 Your solution to solve this problem

With the rapid growth of distributed systems, optimization of task execution procedures is one of the most important issues. Task scheduling in distributed systems finds role in improving efficiency in several applications such as communication, routing, production plans etc. The most important factor in good task scheduling is to minimize the makespan and improve resource utilization. However, the recent and previous effort usually focused on minimizing makespan.

Thus, our solution to this problem is to implement task-scheduling method in grid resources based on Ant Colony Optimization (ACO) algorithm with considerations to precedence and communication characteristics.
In the mentioned method in addition to optimization of task execution time, failed/missed jobs are also handled. In this method, by using of a new heuristic list, an algorithm based on ant colony is implemented. The results obtained are compared with the latest similar models of random search algorithms.

3.6 Where your solution different from others

Our task scheduling algorithm does local search based on Ant Colony Algorithm and gives optimal performance even when there is a large number of processors. Our solution is based on heuristic algorithm and assigns priority to tasks. This provides efficient scheduling results and incorporates real parallel processing. The task pool is also constantly searched for any failed/missed task. The task is then appropriately assigned to the best processor.
Thus, we are proposing a solution using the optimization of ant colony algorithm based on population search to reduce execution time and to improve the processor/resource utilization. The efficiency of the proposed solution is much improved than other algorithms as we are using a parameter called heuristic in probability expansion function.

3.7 Why your solution is better

Some of the existing task scheduling algorithms are:

- First-Come First-Serve Scheduling, FCFS.
- Shortest-Job-First Scheduling, SJF.
- Priority Scheduling.
- Round Robin Scheduling.
- Multilevel Queue Scheduling.
- Multilevel Feedback-Queue Scheduling.

In the above algorithms, starvation may be possible for the longer processes and the lowest priority processes. These scheduling methods are non-preemptive, that is, the process runs until it is finished. Also, there is no special priority to important tasks.
In our solution, by modifying the pheromone trail dynamically, we obtain the shortest path optimally and adaptively in scalable, dynamic and distributed environment. This enhancement process helps in optimal scheduling by completing the tasks with minimum execution time as well as utilizing the resources in a very efficient way.
Our solution is better because it not only helps to minimize deadlines, but also helps in minimizing makespan, maximizing processor utilization and load balancing.

4 Hypothesis (or goals)

4.1 Multiple hypothesis

Ants find the shortest path to food source from their nest. This kind of indirect communication via local environment is called stigmergy. The same principle if used in other applications such as Travelling Salesman Problem, Quadratic Assignment Problem, Networking modeling problem, vehicle routing, must provide the most optimum path to traverse through all the points.

Ants have very limited capabilities. They have rudimentary sight, limited visual and auditory communication, but are capable of producing very impressive group results, which implies that the same algorithm if applied in distributed environment, must be capable of producing very good results in terms of efficiency and makespan.

There is always a "Robert Frost Ant", that chooses the less chosen path and is crucial as exploration of different paths is extremely important in the search process. Such ant are comparatively very less in number and called "Robert Frost Ant" as he once quoted in his Poem - The Road Not Taken:
"Two roads diverged in a wood, and I—I took the one less traveled by, and that has made all the difference."

4.2 Positive/negative hypothesis

More the ants use a path, the more the pheromone trail grows stronger. Thus the path becomes more attractive for other ants. This is an example of a positive feedback.

ACO algorithms make use of simple agents called ants, which iteratively construct candidate solutions to a combinatorial optimization problem. The ants' solution construction is guided by (artificial) pheromone trails and problem-dependent heuristic information. The task is selected according to the heuristic value and pheromone trail and then the selected task is assigned to the best processor [6].

5 Methodology

5.1 How to generate/collect input data

- Create a finite number of Grid resource.
- A Grid resource may contain one or more Machines.
- Similarly, a Machine contains one or more PEs (Processing Elements or CPUs).
- Create a finite number of tasks with known processing time and are placed in the task pool.
- Best task is selected after optimal selection using Ant Colony Optimization and assigned to the best processing element or CPU.

5.2 How to solve the problem

5.2.1 Algorithm design

Step 1: Create a set of grid resources, in which each grid resource may contain one or more Machines. Each machine may contain one or more processing elements.
Step 2: Create a finite set of tasks/jobs/gridlets with known communication characteristics and place them in a task pool.

Step 3: Initialize all the nodes with uniform pheromone level, ρ. Initialize the value of heuristic parameters α, β such as $0 < \alpha, \beta < 1$.

Step 4: Randomly place ants on the grid.

Step 5: Initialize the pheromone value, PV matrix which relates to the execution time of task, t_i by processor p_j.

Step 6: Ants progress forward tracing the path by probabilistically selecting the next node based on relative pheromone level.

Step 7: Eliminate loops in the path traced.

Step 8: Retrace steps.

Step 9: The largest entry in the trail matrix will be selected by proposed technique as the processor to process the selected task.

Step 10: Assign the gridlet g_i selected to the best grid resource, r_j.

Step 11: The execution results will be sent to the user.

Step 12: Global update of the trial is done by evaporating a portion of the pheromone trail according to the parameter ρ

Step 13: Repeat step 6 to Step 12

Step 14: Check if there is any failed task. If yes, assign to the best available processor. If no, go to step 15.

Step 15: Check if any task is missed. If yes, assign again to the best available processor. If no, go to step 16.

Step 16: Display execution results.

Step 17: Exit.

5.2.2 Language used

Java Programming Language will be used to implement the algorithm.

5.2.3 Tools used

The proposed algorithm will be implemented by sung GridSim-5.2 Toolkit [7] with the help of Eclipse Neon (4.6) IDE using Java Programming Language.

5.3 How to generate output

Java-based Grid simulation toolkit, called GridSim is used to generate output.The source code is written in Java using Eclipse IDE. The toolkit, built on a basic discrete event simulation system, called JavaSim, provides facilities for modeling and simulating Grid resources (both time and space-shared high performance computers) and network connectivity with different capabilities and configurations.
The Java-based GridSim discrete event simulation toolkit provides Java classes that represent entities essential for application, resource modeling, scheduling of jobs to resources, and their execution along with management[7].

5.4 How to test against hypothesis

The proposed approach is simulated on the GridSim-5.2 Toolkit[7] with the help of MyEclipse 8.5 IDE using Java Programming Language.

A set of heterogeneous grid resources is created. Also, a set of gridlets/tasks is created, which are allocated to the resources based on the scheduling policy. ACO based scheduling results are compared with that of FCFS(First Come First Serve.

The processing cost is generated for scheduling algorithms algorithms FCFS and Ant Colony Algorithm and relative comparison is done.
Processing Cost = actual CPU Time * cost per sec
The makespan is computed for the two scheduling algorithms and their results are compared.

6 Implementation

6.1 Code

The code consists of the source code to implement gridsim.
The following jar files were imported to run and test the algorithm on this tool.

1. gridsim.jar
2. simjava2.jar

The following files are added to implement our proposed algorithm:
1. AntColonyScheduler.java
2. GridletCreation.java
3. Ant.java

Pseudocode :

```
Initialize the gridsim package,
Set num of user =1
Input no of gridresources
Declare list of GridResource
For each gridresource,
        Input resource name, no of machines, MIPS rating
        Call creategridresource by passing above parameters

Declare jobs objects
Input no of jobs
Input jobs(gridlet) creation method(R:random,M:manual)
If(manual method is selected)
        For each job,
                Input job length;
                Input file size
                Input output file size
                Input job deadline
                Create job with above parameters
                Assign user id for each job
                Add job to joblist
If(Random method is selected)
        Create jobs with help of GridSimRandom & gridsimStandardPE class
        Set standard PE MIPS rating to 100
        For each job
                Call GridSimStandardPE methods to create job length, file size, outputfile size,
deadline by passing random number;
                Assign user id for each job
                Add job to joblist


Start simulation by invoking body method
While (true)
        Hold for 1 second
        Set resourceList = getGridResourceList
        If resourceList is equal to total no of resources then
```

```
                break out of loop
        else
                print message to wait
For (each grid resource)
        get recource characteristics by calling send method of gridsim
        set resoucename []= name of each resource from resource characteristics
        set recource cost [] = cost of each resource from resource characteristics
        set MIPS [] = MIPS rating each resource from resource characteristics
        set FreePE [] = free processing list of each resource from resource characteristics

Input scheduleType = 1 for FCFS, 2 Ant colony
Declare PVMatrix [][] = array[no of resources][no of jobs]
For (each grid resource)
  For (each job)
     PVMatrix= ((job length/100)+job_deadline/(resource_MIPS*no_of _free PE));

Initialize number of ants to be placed on the grid = numAntFactor*num of jobs
Create list of Ant objects
For each ant
        Declare & construct tour[];
        Declare & construct visited[];
        Declare & calculate tourlength using PVmatrix;

Initialize trails [][]= array[no of resources][no of jobs]
Set trails value to initial constant pheromone value
While (there are iterations)
        setupAnts();
        moveAnts();
        updateTrails();
        updateBest();
        increment iteration;

Set best_tour[]  = job_priority list
declare best resource[];
For (each job)
        Set best resource = job id of maxing trail value;
Submit sorted resources and jobs to gridsim for simulation
For (each job)
        Print job id, allocated resource id, Total processing cost

----------------------------------------------------------------------------
1. setupAnts:
For (each ant)
        update tour array by passing random job id
        initialize visited flag = false for that job id
        Update ant list = tour array

2. moveants:
        For (each ant)
                While (follow trails)
                        Create list of visited job id
                Update visited flag = true
3. updateTrails:
    For (each ant)
        Initialize contribution =  trail deposit coefficient /tour length;
        Trails matrix [][]= contribution
4. updateBest:
    For(each ant)
        If(tourlength<besttourlength)
            bestTourLength = a.tourLength();
        bestTour = a.tour.clone();
```

6.2 Design Document and flowchart

6.2.1 Design Flow

1. Initialize the gridsim package.
2. Get the number of Grid resources as input from the user.
3. The following resource characteristics are obtained from the user
    - Name of the grid resource
    - MIPS rating of the grid resource
    - Number of machines in each grid resource
4. For machine k, all the processing elements have same MIPS (Million Instructions per Second) rating and 4 Processing Elements.
5. The following parameters are fixed for the grid resources.
    - System architecture - "Sun Ultra"
    - OS - "Solaris"
    - time_zone = 9.0
    - Cost per sec = 3.0
    - Baud_rate = 100.0
6. Create Grid Resources based on the resource parameters.
7. Get the number of gridlets/jobs/tasks as input from the user.
8. The following gridlet characteristics are obtained from the user
    - Length of the gridlet(MI)
    - File Size of the gridlet
    - Output size of the gridlet
    - Deadline of the gridlet
9. Create set of gridlets manually based on the properties entered by user or randomly using GridSimRandom.
10. Start the Gridsim simulation.
11. Select the scheduling policy : FCFS/ ACO
12. FCFS : Allocate the gridlet to a resource on a first come first serve basis.
13. ACO : Gridlet is allocated to the grid resource based on Ant Colony Optimization.
    - Initialize the Ant Colony Algorithm parameters.
        - Original amount of pheromone trail, c = 1.0

        - Pheromone trail preference, alpha = 1
        - Greedy preference, beta = 5
        - Pheromone trail evaporation coefficient, evaporation = 0.5
        - New trail deposition coefficient, Q = 500
        - Number of ants to be placed on the grid = numAntFactor * numTasks
        - numAntFactor = 0.8
        - Number of iterations, maxIteration = 2000
    - Create heusristic desirability matrix based on the grid resource and gridlet characteristics and is given as input to ant colony optimization.
    - Ants begin their journey from one tasks to another and find the best route by the end of iterations.
    - Analyze the pheromone trail information and obtain gridlet priority list and best resource for each resource.
    - Allocate the gridlets to the best resource.

14. Print the simulation results once the simulation is complete.

6.2.2 Flowchart

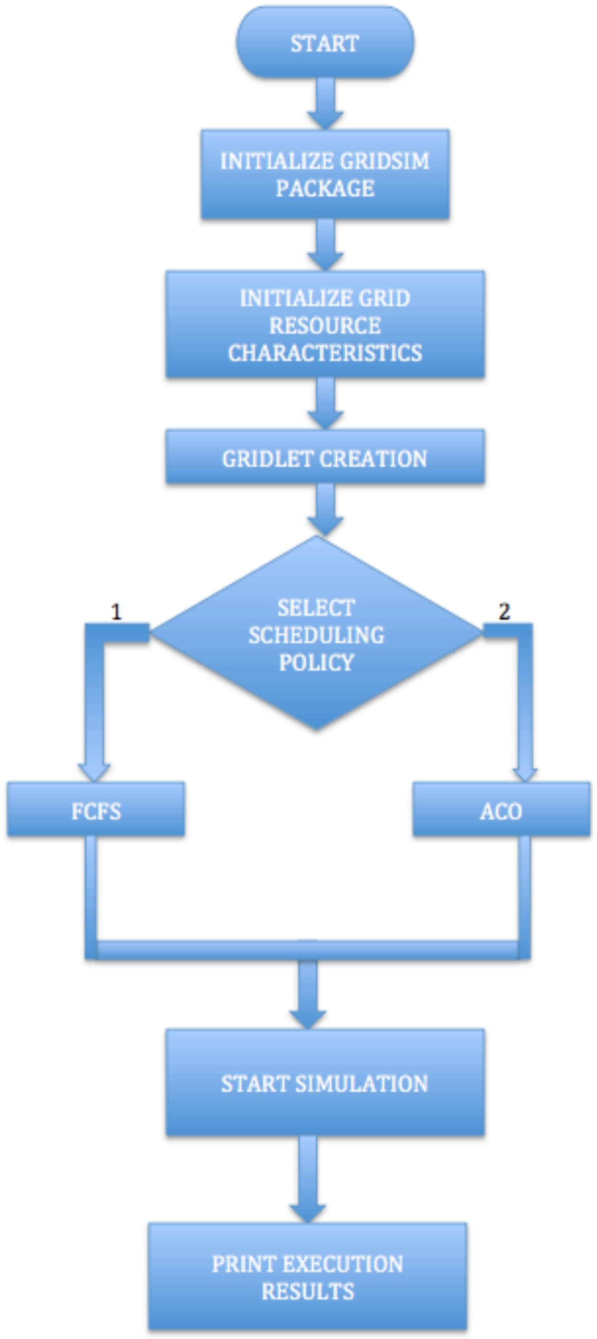


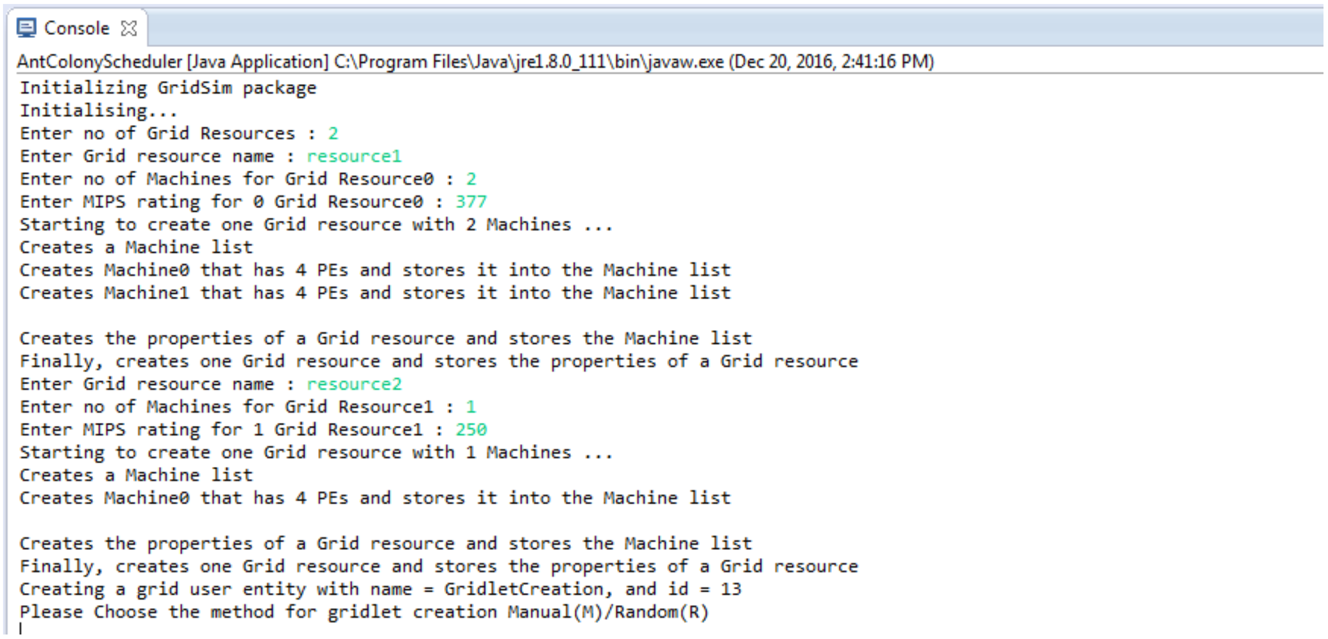
Figure 2 : Design Flowchart

7 Data analysis and discussion

7.1 Output generation

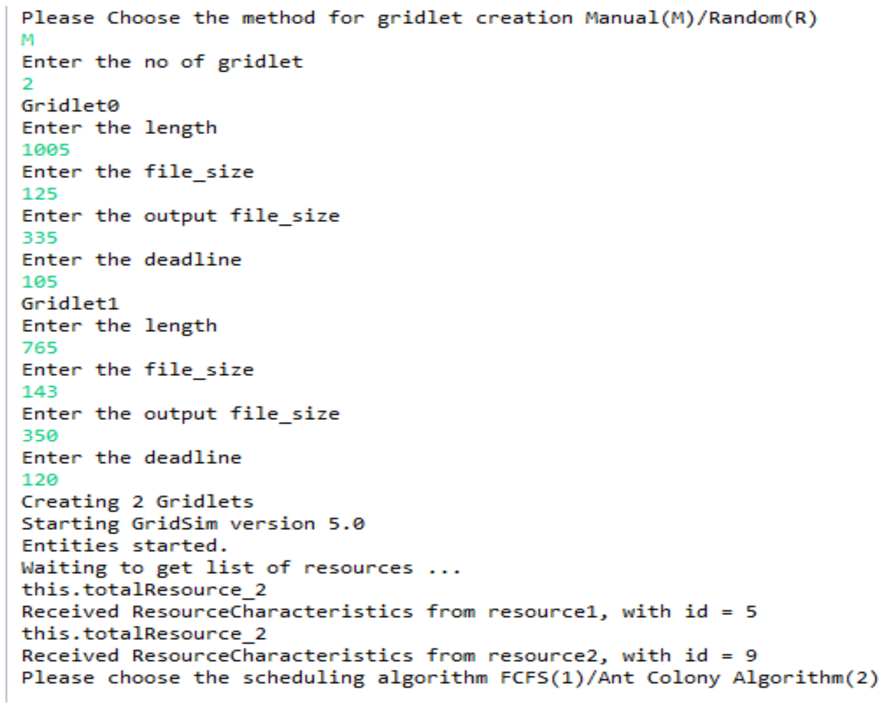
In order to generate the output we ran the implemented algorithm for FCFS and Ant colony.
We have calculated total cost by adding cost of each job. And compared total cost for both algorithm.
Below are the screen shots after running of FCFS and our ANT colony with the same no of jobs and resources.

1. Create grid resources



```
AntColonyScheduler [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (Dec 20, 2016, 2:41:16 PM)
Initializing GridSim package
Initialising...
Enter no of Grid Resources : 2
Enter Grid resource name : resource1
Enter no of Machines for Grid Resource0 : 2
Enter MIPS rating for 0 Grid Resource0 : 377
Starting to create one Grid resource with 2 Machines ...
Creates a Machine list
Creates Machine0 that has 4 PEs and stores it into the Machine list
Creates Machine1 that has 4 PEs and stores it into the Machine list

Creates the properties of a Grid resource and stores the Machine list
Finally, creates one Grid resource and stores the properties of a Grid resource
Enter Grid resource name : resource2
Enter no of Machines for Grid Resource1 : 1
Enter MIPS rating for 1 Grid Resource1 : 250
Starting to create one Grid resource with 1 Machines ...
Creates a Machine list
Creates Machine0 that has 4 PEs and stores it into the Machine list

Creates the properties of a Grid resource and stores the Machine list
Finally, creates one Grid resource and stores the properties of a Grid resource
Creating a grid user entity with name = GridletCreation, and id = 13
Please Choose the method for gridlet creation Manual(M)/Random(R)
```

2. Create jobs


```
Please Choose the method for gridlet creation Manual(M)/Random(R)
M
Enter the no of gridlet
2
Gridlet0
Enter the length
1005
Enter the file_size
125
Enter the output file_size
335
Enter the deadline
105
Gridlet1
Enter the length
765
Enter the file_size
143
Enter the output file_size
350
Enter the deadline
120
Creating 2 Gridlets
Starting GridSim version 5.0
Entities started.
Waiting to get list of resources ...
this.totalResource_2
Received ResourceCharacteristics from resource1, with id = 5
this.totalResource_2
Received ResourceCharacteristics from resource2, with id = 9
Please choose the scheduling algorithm FCFS(1)/Ant Colony Algorithm(2)
```

3. FCFS

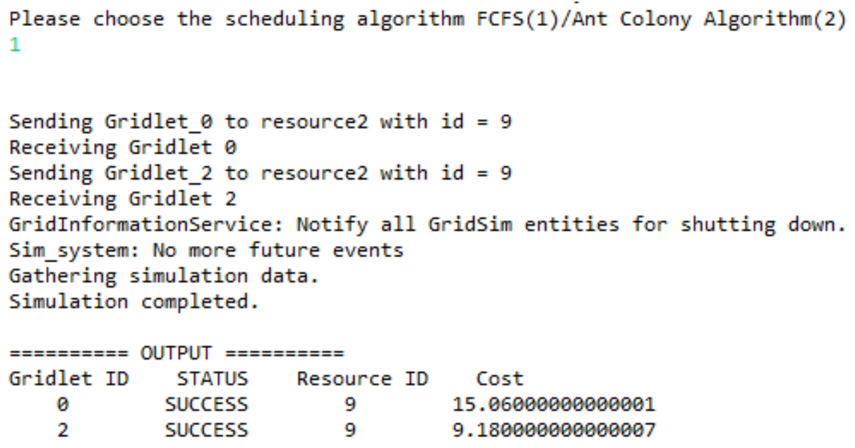
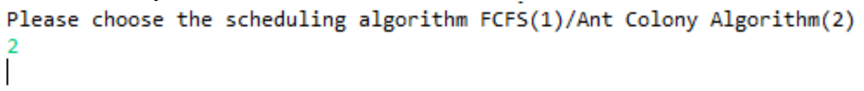
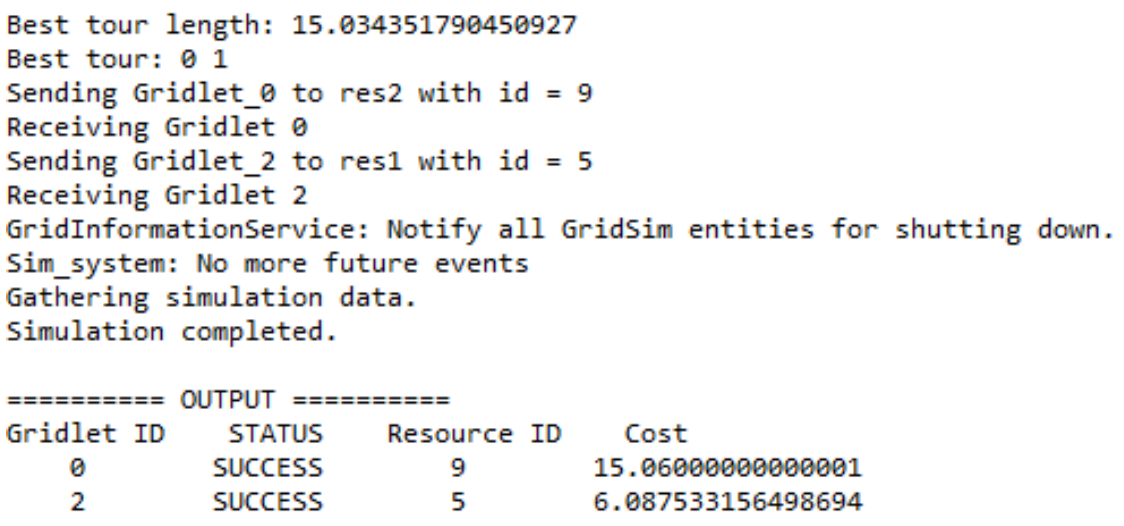

```
Please choose the scheduling algorithm FCFS(1)/Ant Colony Algorithm(2)
1


Sending Gridlet_0 to resource2 with id = 9
Receiving Gridlet 0
Sending Gridlet_2 to resource2 with id = 9
Receiving Gridlet 2
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.

========== OUTPUT ==========
Gridlet ID    STATUS    Resource ID    Cost
    0        SUCCESS        9        15.060000000000001
    2        SUCCESS        9        9.180000000000007
```

Figure 3 : FCFS Screenshot

4. Ant colony


```
Please choose the scheduling algorithm FCFS(1)/Ant Colony Algorithm(2)
2
```



```
Best tour length: 15.034351790450927
Best tour: 0 1
Sending Gridlet_0 to res2 with id = 9
Receiving Gridlet 0
Sending Gridlet_2 to res1 with id = 5
Receiving Gridlet 2
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.

========== OUTPUT ==========
Gridlet ID    STATUS    Resource ID    Cost
    0        SUCCESS        9         15.060000000000001
    2        SUCCESS        5         6.087533156498694
```

Figure 4 : ACO Screenshot

7.2 Output analysis

We have run the program to generate the gridlets either random or manual. When we are generating the gridlets manually below is result which we have found by comparing different algorithm results. (FCFS/ACO)

1. **No of Resources = 2**

No of Resources	R1	R2
MIPS Rating	298	358
No. of Machine	2	1
No of PE's	8	12

No of Gridlets (Task to be executed)	G1	G2
Length	1000	750
File_Size	225	300
O/P file Size	350	420
Deadline	105	65

Output of FCFS

Gridlet ID	STATUS	ResourceID	Cost
0	SUCCESS	5	13.067
2	SUCCESS	5	7.55
Total			20.122

Table 1 : FCFS Output(2 Grid Resources)

Output ACO

Gridlet ID	STATUS	ResourceID	Cost
2	SUCCESS	9	6.2849
0	SUCCESS	5	10.06711
Total			16.35201

Table 2 : ACO Output (2 Grid Resources)

2. No of Resources =3

No of Resources	R1	R2	R3
MIPS Rating	253	377	233
No. of Machine	2	1	2
No of PE's	8	4	8

No of Gridlets (Task to be executed)	G1	G2	G3
Length	1110	300	255
File_Size	135	132	145
O/P file Size	330	345	315
Deadline	105	75	145

Output FCFS

Gridlet ID	STATUS	ResourceID	Cost
0	SUCCESS	13	17.29
2	SUCCESS	13	6.86
4	SUCCESS	9	3.0
Total			27.15

Table 3 : FCFS Output (3 Grid Resources)

Output ACO

Gridlet ID	STATUS	ResourceID	Cost
2	SUCCESS	9	3.0
4	SUCCESS	5	3.023
0	SUCCESS	13	17.29
Total			23.313

Table 4 : ACO Output (3 Grid Resources)

3. No Of Resources= 4

No of Resources	R1	R2	R3	R4
MIPS Rating	240	377	353	212
No. of Machine	1	2	1	2
No of PE's	**4**	**8**	**4**	**8**

No of Gridlets (Task to be executed)	G1	G2	G3	G4
Length	1005	404	655	755
File_Size	125	133	144	138
O/P file Size	350	313	389	304
Deadline	85	65	108	213

Output FCFS

Gridlet ID	STATUS	ResourceID	Cost
0	SUCCESS	5	12.56
2	SUCCESS	17	8.716
4	SUCCESS	5	8.187
6	SUCCESS	13	6.416
Total			35.8719

Table 5 : FCFS Outrput(4 Grid Resources)

Output ACO

Gridlet ID	STATUS	ResourceID	Cost
6	SUCCESS	9	6.0079
2	SUCCESS	5	5.05
4	SUCCESS	13	5.56
0	SUCCESS	17	17.22
Total			33.0162

Table 6 : ACO Output (4 Grid Resources)

4. No Of Resources= 5

No of Resources	R1	R2	R3	R4	R5
MIPS Rating	253	300	251	200	315
No. of Machine	2	1	3	4	2
No of PE's	**8**	**4**	**12**	**16**	**8**

No of Gridlets (Task to be executed)	G1	G2	G3	G4	G5
Length	1100	1200	1300	1000	900
File_Size	135	125	145	150	150
O/P file Size	300	200	255	300	375
Deadline	100	150	120	140	125

Output FCFS

Gridlet ID	STATUS	ResourceID	Cost
0	SUCCESS	17	16.5
2	SUCCESS	13	17.34263
4	SUCCESS	17	19.5
6	SUCCESS	13	14.95219
8	SUCCESS	21	8.571429
Total			76.866

Table 7 : FCFS Output (5 Grid Resources)

Output ACO

Gridlet ID	STATUS	ResourceID	Cost
0	SUCCESS	5	16.04348
8	SUCCESS	9	9
4	SUCCESS	17	19.5
6	SUCCESS	13	14.95219
2	SUCCESS	21	14.42857
Total			73.924

Table 8 : ACO Output (5 Grid Resources)

7.3 Compare output against hypothesis

From the graph we can see the Processing Cost using ACO algorithm is lesser than FCFS.
Thus, Makespan is reduced with ACO as Processing Cost = Cost per sec * CPU Execution Time.

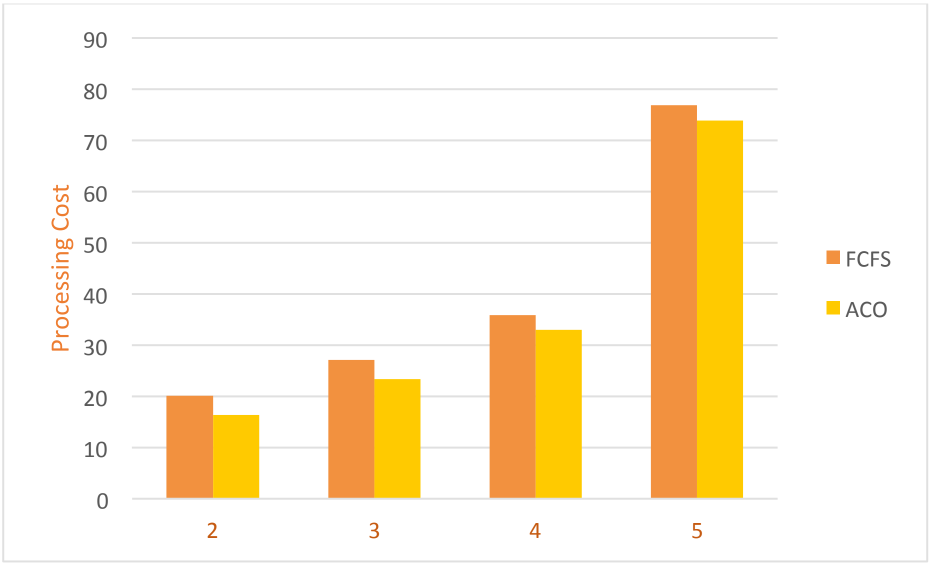


Figure 5 : Plot of No. of Grid Resources vs Processing Cost

7.4 Abnormal case explanation

In few cases where number of resources and no of task to be executed are less we are not able to see significant difference in the performance and also in some situation where parameters values are not consistent in such cases this algorithm may not give correct results.

7.5 Discussion

The main aim of our project was to minimize makespan and maximize CPU utilization. Makespan is the total length of the schedule (that is, when all the jobs have gone through their processing). For given set of jobs with identical or different processing times and set of resources, assigning jobs to resources in such a way that total completion time of task is minimized or in other words makespan is minimized.

We have compared FCFS and Ant colony algorithm for equal no of jobs and resources, results shows that tasks are almost equally divided in no of resources so that each resource is utilized properly where as in FCFS same resources are used for different task results in some of resources may left as unused.

In Ant colony algorithm, optimal tour of resources is selected by multiple times iterating over given resources. Also best tour length is calculated. Optimal tour of resources corresponds to which resources should be allocated to complete given job and best tour length corresponds to time required to finish job. As Ant colony algorithm finds its solution by considering all possibilities, end result is most optimized one.

8 Conclusions and recommendations

8.1 Summary and conclusions

Scheduling in distributed systems has a significant role in overall system performance and throughput. The Ant Colony Optimization has proved to produce optimized results for many problems in science and engineering area.
Simulation results demonstrate that Ant Colony Algorithm based scheduling algorithm outperforms FCFS (First Come First Serve) based scheduling. ACO helps in minimizing makespan and maximizing CPU utilization.
The traditional methods try to reduce the overall response time by giving an optimized schedule. But they fail to produce a load balanced schedule. Moreover they do not share the load among the available resources. The proposed method also helps in load balancing as the grid resource with minimum load is attracted by most of the ants. Thus, it is convincingly proved that task scheduling is best solved by heuristic approach.

8.2 Recommendations for future studies

The algorithm proposed here assumes that there is no constraint of resources and any task is able to be scheduled at any time once it arrives. The algorithm can be extended considering the constraints of resources which is required to be considered during real-world applications.
This algorithm can be extend to select the type of scheduling automatically based on the no of task to be executed and available resources to give the better results. The ant colony approach could also be included in the list of future reliable and useful optimization tools in various fields such as chemical engineering, quadratic assignment problems, network routing problems etc.

9 Bibliography

[1] Ranjan Kumar Mondal , Enakshmi Nandi and Debabrata Sarddar , "Load Balancing Scheduling with Shortest Load First" International Journal of Grid Distribution Computing Vol. 8, No.4, (2015), pp. 171-178

[2] W. Li and H. Shi, "Dynamic Load Balancing Algorithm Based on FCFS" IEEE, (2009), pp. 1528-1531..

[3] Subasish Mohapatra,Subhadarshini Mohanty and K.Smruti Rekha, "Analysis of Different Variants in Round Robin Algorithms for Load Balancing in Cloud Computing" International Journal of Computer Applications (0975 – 8887) Volume 69– No.22, May 2013

[4] Mr.V. P. Narkhede, and Prof. S. T. Khandare, "Fair Scheduling Algorithm with Dynamic Load Balancing Using In Grid Computing" Research Inventy: International Journal Of Engineering And Science Vol.2, Issue 10 (April 2013), Pp 53-57 Issn(e): 2278-4721, Issn(p):2319-6483, Www.Researchinventy.C

[5] Rohit Saxena , Ankur Kumar, Anuj Kumar and Shailesh Saxena, "AHSWDG:An Ant Based Heuristic Approach toScheduling & Workload Distribution in Computational Grids" 2015 IEEE International Conference on Computational Intelligence & Communication Technology

[6] M. H. Kashani , R. Sarvizadeh "A Novel Method for Task Scheduling in Distributed Systems Using Max-Min Ant Colony Optimization" from the proceedings of 20I I 3rd International Conference on Advanced Computer Control (ICACC 201 I).

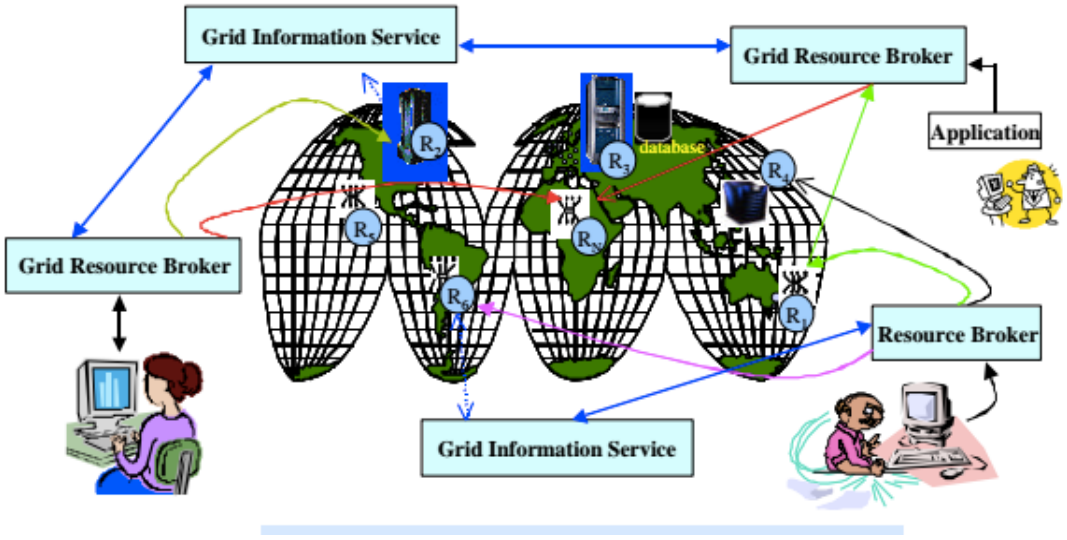
[7] Rajkumar Buyya and Manzur Murshed "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing"

[8] M.Kaladevi, and Dr.S.Sathiyabama "A Comparative Study of Scheduling Algorithms for Real Time Task" International Journal of Advances in Science and Technology, Vol. 1, No. 4, 2010

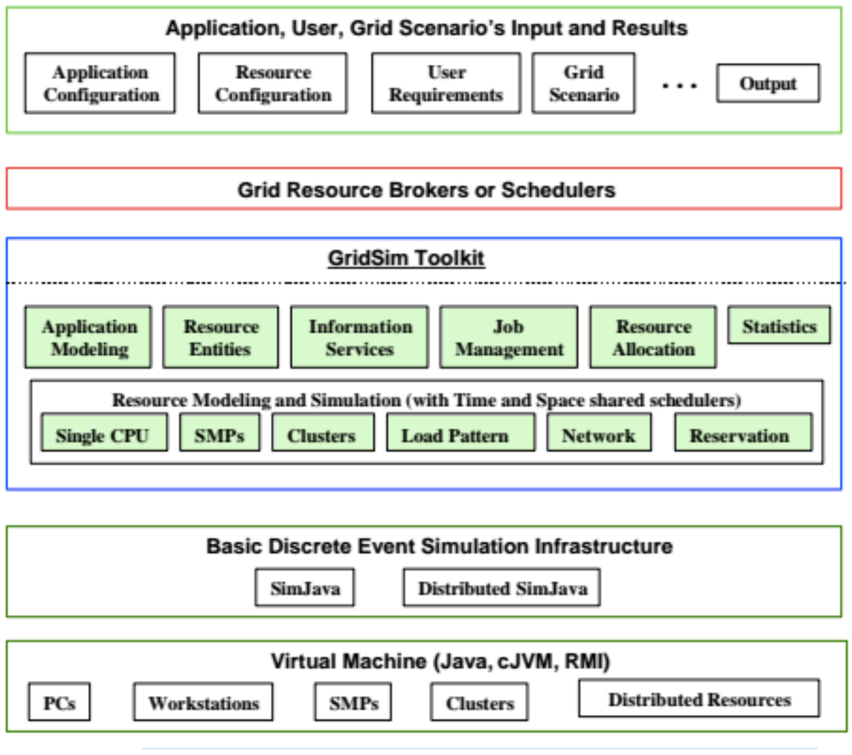
[9] Saeed Molaiy, Mehdi Effatparvar "Scheduling in Grid Systems using Ant Colony Algorithm", I.J. Computer Network and Information Security, 2014, 2, 16-22.

10 Appendix

A. A generic view of World-Wide Grid computing environment.



B. A modular architecture for GridSim platform and components.


Application, User, Grid Scenario's Input and Results
Application Configuration
Resource Configuration
User Requirements
Grid Scenario
. . .
Output
Grid Resource Brokers or Schedulers
GridSim Toolkit
Application Modeling
Resource Entities
Information Services
Job Management
Resource Allocation
Statistics
Resource Modeling and Simulation (with Time and Space shared schedulers)
Single CPU
SMPs
Clusters
Load Pattern
Network
Reservation
Basic Discrete Event Simulation Infrastructure
SimJava
Distributed SimJava
Virtual Machine (Java, cJVM, RMI)
PCs
Workstations
SMPs
Clusters
Distributed Resources