Task Scheduling in Grid Environment with Ant Colony method for Reliability and Time

Ms. Garima Garg\(^1\), Mrs. Sheetal Kundra\(^2\)

\(^1\)M.Tech Student, Computer Science & Engineering Deptt., Rayat Institute of Engineering & Information Technology, Railmajra, Punjab, INDIA
\(^2\)Asstt. Prof., Computer Science & Engineering Deptt., Rayat Institute of Engineering & Information Technology, Railmajra, Punjab, INDIA

\(^1\)garimanitk@yahoo.co.in, \(^2\)sheetalkundra81@gmail.com

Abstract— Grid computing is a form of distributed computing that coordinates and shows computing power, applications, data storage and network resources across dynamic and geographically dispersed organizations. Resource Management in Grid computing system is a fundamental issue in achieving high performance due to the distributed and heterogeneous nature of the resources. The efficiency and effectiveness of grid resource management greatly depend on the scheduling algorithm. Usually the objective of scheduling algorithms is to map tasks onto processors and to order their executions so that task precedence requirements are satisfied and in the meanwhile, the minimum schedule length (makespan) can be achieved. In this paper, the problem of scheduling is represented by a weighted directed acyclic graph (DAG) where tasks are upward ranked and sorted decreasingly. Furthermore, based on directed acyclic graph (DAG) we propose a scheduling algorithm for precedence constrained tasks, which can achieve high quality of reliability for applications. Ant algorithm, which is one of the heuristic algorithm suits well for the grid scheduling environment. Ant Colony Optimization is used for scheduling tasks on resources in Grid which simultaneously pay attention to two objectives of makespan (schedule length) and the failure probability (reliability). These objectives are conflicting and it is not possible to minimize both objectives at the same time. With the help of concept of non-dominance, we are able to choose a trade-off between makespan minimization and reliability maximization.

Keywords—Task scheduling; DAG; makespan; reliability

I. INTRODUCTION

Grid computing systems have emerged as a new environment for coordinated resource sharing and problem solving in multi-institutional virtual organizations while providing dependable, consistent, pervasive access to global resources. The sharing ranges from simple file transfer to direct access to computers, software, data, and other network accessible resources. Grid resources are heterogeneous, dynamic, complex and self-autonomic in nature which makes the resource management a significantly challenging job. Job Scheduling and Resource Management are the critical issue in Grid Computing [1]. The multiprocessor task schedule problem is known to be NP-complete. In order to address this problem, many heuristics algorithms have been proposed. These heuristics are classified into different categories such
as list scheduling algorithms, clustering algorithms, duplication based algorithms [2].

Task scheduling is a challenging problem in grid computing environment. Many parallel applications consist of multiple computational components. While the execution of some of these components or tasks depends on the completion of other tasks, others can be executed at the same time, which increases parallelism of the problem. The task scheduling problem is the problem of assigning the tasks in the system in a manner that will optimize the overall performance of the application, while assuring the correctness of the result. The task scheduling problem can be modeled as a weighted directed acyclic graph (DAG). A vertex represents a task, and its weight the size of the task computation. An arc represents the communication among two tasks, and its weight represents the communication cost. The directed edge shows the dependency between two tasks. The primary goal of task scheduling is to schedule tasks on processors and minimize the makespan of the schedule, i.e., the completion time of the last task relative to the start time of the first task. The output of the problem is an assignment of tasks to processors.

Ant colony algorithm is one of the effective techniques used to solve NP-complete problems. ACO is a recently proposed metaheuristic approach for solving hard combinatorial optimization problems. The inspiring source of ACO is the pheromone trail laying and following behavior of real ants which use pheromones as a communication medium. Many efforts have been done to use ant colony to solve scheduling problems. ACO was used to solve many NP-hard problems such as traveling salesman problem [3], vehicle routing problem [4], and graph coloring problem [5] and so on. In [6] this technique has been applied for independent task scheduling. In this paper, we apply the above technique for dependent task scheduling. The input to the scheduling algorithm is a directed acyclic graph (DAG), in which the node weights represent task processing times and the edge weights represent data dependencies as well as the communication times between tasks. The multi-objective ACO algorithm uses non-dominance approach for tackling the two objectives (makespan and reliability).

II. TASK SCHEDULING PROBLEM

The basic elements of a scheduling system model are an application program, certain environment to run the application on and some strategy for the scheduling of application. There is used an ordered graph called Directed Acyclic Graph (DAG) for this purpose.

The application program can be represented by a directed acyclic graph G (V, E, C), where: a set V of nodes representing n number of the subtasks of the application and a set E of edges shows the dependencies among the subtasks. Each and every task is related by directed edge graph represents the communication directions between tasks and precedence constraints (i.e. data dependency) by e_{ij}. A directed edge e_{ij} \in E indicates the data dependency constraint exists between the tasks t_i and t_j. On the other hand, if in the graph the t_i precedence the t_j , t_j may not begin its work until t_i has not been completed. In this model, a task until conducting all parents of that node have not been completed, may not be sent to a resource for being processed. The value considered on the edge is the communication time of data transfer between tasks. If both parent and offspring tasks processed on a same resource, the value of communication time between them is considered about zero. And C is the set of communication times. Edge e_{ij} has a communication time c_{ij} in C.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>13</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>V2</td>
<td>15</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>V3</td>
<td>18</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>V4</td>
<td>16</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>V5</td>
<td>13</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>
In any DAG, there is always an input node, \( t_{\text{entry}} \) as a node with no parent and an output node, \( t_{\text{exit}} \) as a node with no offspring. When DAG has several entry or exit tasks, these tasks are connected to a pseudo entry-task or pseudo exit-task that has zero load weight and zero capacity edges.

Fig. 1 shows an example of a DAG \( G = (V, E) \), where \( V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\} \) comprising of 7 tasks to illustrate these definitions graphically, and the set of directed edges \( E \) consists of \{e_{12}, e_{13}, e_{15}, e_{24}, e_{25}, e_{36}, e_{47}, e_{57}, e_{67}\}. Table 1 indicates tasks computing cost matrix (W) existing in the program independently on different processors. This matrix called execution time of completion cost matrix (ETC) is entered to the system.

A schedule is a function \( S: V \rightarrow P \) assigns a task to the processor that executes it. Let \( V(j, s) = \{i \mid s(i) = j\} \) be the set of tasks assigned to processor \( j \).

The completion time of a processor \( j \) is calculated as

\[
C_j(s) = \sum_{i \in V(j, s)} st_{ij} + w_{ij}
\]

Where \( st_{ij} \) denotes the start time of the task \( i \) on processor \( j \). The start time of the entry task is assumed to be zero. Other tasks’ start time can be computed by considering the completion time of all immediate predecessors of the task. The communication time, \( st_{ij} \) is added to the start time, if the dependent tasks are allocated to different processors. The makespan of a schedule is the time where all tasks are completed.

\[
C_{\text{max}} = \max_j C_j(s)
\]

Table 1 ETC Matrix

<table>
<thead>
<tr>
<th></th>
<th>15</th>
<th>14</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>V6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V7</td>
<td>13</td>
<td>17</td>
<td>11</td>
</tr>
</tbody>
</table>

Every Processor has a constant failure rate, and let \( \lambda_j \) denote the failure rate of processor \( j \). The probability that a processor \( j \) executes all its tasks successfully is given by

\[
P_{\text{succ}}^j(S) = e^{-\lambda_j C_j(s)}
\]

It is assumed that faults are independent, therefore, the probability that schedule \( S \) has finished correctly is

\[
P_{\text{succ}} = e^{-\sum_j \lambda_j C_j(s)}
\]

The reliability index (rel) is defined by

\[
\text{rel}(s) = \sum_j \lambda_j C_j(s)
\]

Minimizing the objective function \( \text{rel} \) is equivalent to maximizing the probability of success of the schedule on a parallel heterogeneous system subject to failure. For solving the task scheduling problem the objectives \( C_{\text{max}} \) and \( \text{rel} \) are to be minimized simultaneously.

### III. CONCEPT OF NON-DOMINANCE

A multi-objective problem does not have a single optimal solution but rather a set of optimum solutions called Pareto Optimal Set (POS) and point set defined by POS in the value space of objective functions is known as Pareto front. Pareto front consists of just non-dominated solutions. A solution dominates other if and only if it is at least as good as other in all the objectives and it is better in at least one of the objectives (Concept of Dominance).

**Non-dominated solutions:** The solutions used for updating the pheromone information are the non-dominated solutions. When there are more non-dominated solutions than size of non-domination set, we apply the truncation mechanism of Crowding
Distance to select only solutions for which pheromone updation is to be done.

IV. PROPOSED ALGORITHM

In this section we are presenting the algorithm for the dependent task scheduling in Grid heterogeneous system using Ant Colony Optimization technique, which aims at achieving high reliability and reducing the makespan. The algorithm consists of two mechanisms, a ranking mechanism [20], which is a modified version of the HEFT [18, 19] and a processor assignment mechanism.

The algorithm is described at:
Step 1: Input the processor graph and task graph DAG.
Step 2: Compute RRank for all tasks by traversing the graph from the exit task using Ranking () method.
Step 3: Sort the tasks in a scheduling list by descending order of RRank.
Step 4: For each task in scheduling list, apply ACO algorithm.
Step 5: For each local solution generated by all ants, find the makespan and reliability index.
Step 6: Apply the concept of non-domination on local solutions to find the globally best solution.

Ranking () Method

Our algorithm will use reliability rank (RRank) attribute to compute priorities of tasks. The RRank [20] is a rank of a task, from the exit task to itself, and equal to the sum of average communication costs of edges, average computation costs and reliability overhead of tasks over all processors. Communication costs between tasks scheduled on the same processor are assumed zero, and the execution constraints are preserved.

The RRank is recursively defined as:

\[
RRank(v_i) = \frac{1}{w(v_i)} + \max_{v_j \in \text{succ}(v_i)} \left\{ \frac{w(e_{i,j})}{RRank(v_j)} \right\} + RC_{vi}
\]

Where \( \text{succ}(v_i) \) is the set of immediate successors of task \( v_i \), \( w(v_i) \) is the average computation cost of task \( v_i \), and \( w(e_{i,j}) \) is the average communication cost of edge \( e_{i,j} \) [18, 19]. The \( RC_{vi} \) is the reliability overhead of task \( v_i \) and can be computed by

\[
RC_{vi} = (1 - \prod_{n=1}^{m} \exp(-\lambda_{in} \times \frac{w(v_i)}{w(p_n)})) \times w(v_i)
\]

The rank is computed recursively by traversing the task graph upward, starting from the exit task. For the exit task \( v_{exit} \), the rank value is equal to

\[
RRank(v_{exit}) = \frac{w(v_{exit})}{w(v_{exit})} + RC_{v_{exit}}
\]

Task Assignment Mechanism

In this mechanism, tasks are assigned to the processors in such a way that makespan is reduced and system reliability is improved. In order to achieve these goals, we find the best mapping of tasks and processors applying ACO. In ACO algorithm, ants put an amount of pheromone on the edges of the path to mark their solution (paths). The strength of pheromone is used to build the solution. The following ant will be attracted by the pheromone, so it will search the solution.

The ant is placed at first task in the generated order and that task is mapped on one of the available resources required by that task. When each of the tasks of the system is mapped to any specific resource then solution construction for that ant is completed and a complete and feasible solution is created. Each task is mapped to a specific resource, as pre-emption is not allowed, that resource is selected based on probability Rule that depends on both pheromone on edges between tasks and resources and heuristic information that depends on objective functions.

\[
P_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{i} \tau_{ij}^{\alpha} \eta_{il}^{\beta}}
\]

Where \( \tau \) is the value of pheromone on edge between task \( i \) and resource \( j \) and \( \eta \) is the value of heuristic information, \( \alpha \) and \( \beta \) are two constants that represents the relative importance of pheromone trail values and
heuristic information values that depends on problem considered.

When each ant constructs its solution, local updating is done to increase the diversity of the algorithm i.e. to avoid different ants to traverse the same path that is traversed by the previous ant. After each iteration of the algorithm i.e. after all the ants completed their solution construction, Global updation is performed to increase the value of pheromone on the edges of the solution that is found to be global best in case of single objective and Non-dominated solutions in case of multi-objective problem, so that the probability of edges of the best solution to be traversed by ants in next iteration of algorithm increases as probability depends on pheromone value also.

V. RESULTS

In my experiments, Graphs are generated for all combinations of the above parameters with the number of tasks range between 5 and 20 and communication to computation cost ratio, (CCR). Every possible edge (DAGs are acyclic) is created with the same probability, which is calculated based on the average number of edges per task node. By using a range of CCR values; different types of applications can be accommodated. According to the simulation results, the performance of the proposed algorithms varies with respect to the No. of tasks and the CCR.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>No. of Tasks</th>
<th>Makespan</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=5,CCR=0.5</td>
<td>5</td>
<td>16090</td>
<td>32.985</td>
</tr>
<tr>
<td>N=10,CCR=0.5</td>
<td>10</td>
<td>31052</td>
<td>63.657</td>
</tr>
<tr>
<td>N=15,CCR=0.5</td>
<td>15</td>
<td>50423</td>
<td>103.367</td>
</tr>
<tr>
<td>N=20,CCR=0.5</td>
<td>20</td>
<td>64347</td>
<td>131.911</td>
</tr>
</tbody>
</table>

Table 2 Table showing different values of No. of tasks and a fixed value of CCR used for calculating makespan and reliability

![Fig. 2 Makespan of GACO algorithm for CCR=0.5](image1)

![Fig. 3 Reliability of GACO algorithm for CCR=0.5](image2)
Now we have compared the performance of Grid schedules obtained from ACO (GACO) using nondominance approach with that heterogeneous earliest-finish time (HEFT) algorithm.

Table 3 Table showing different values of CCR and a fixed value of No. of tasks used for calculating makespan and reliability

<table>
<thead>
<tr>
<th>Criteria</th>
<th>CCR</th>
<th>Makespan</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=10,CCR=0.1</td>
<td>0.1</td>
<td>24799</td>
<td>50.438</td>
</tr>
<tr>
<td>N=10,CCR=0.5</td>
<td>0.5</td>
<td>31052</td>
<td>63.657</td>
</tr>
<tr>
<td>N=10,CCR=1</td>
<td>1</td>
<td>31785</td>
<td>65.159</td>
</tr>
<tr>
<td>N=10,CCR=5</td>
<td>5</td>
<td>39521</td>
<td>81.018</td>
</tr>
</tbody>
</table>

Fig. 4 Makespan of GACO algorithm for No. of tasks=10

Fig. 5 Reliability of GACO algorithm for No. of tasks=10
We observe from Fig. 6 that GACO outperforms HEFT algorithm in terms of the makespan. The results of the experiment in this work showed that the suggested method is an efficient solution for solving the problem of grid scheduling.

VI. CONCLUSION AND FUTURE WORK

The task scheduling problem considered in this paper focus on two objectives of makespan and reliability index. The scheduling algorithm is designed for dependent tasks in Grid environment. The best values for makespan and reliability index are obtained using concept of non-dominance. The promising results of the algorithm invite us to compare the results of the proposed algorithm with others multi-objective heuristics and meta-heuristics approaches.

VII. REFERENCES

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


