Task Scheduling with Parallel Genetic Environment using Stepping Stone Method

Rachhpal Singh
Asstt. Professor, P.G. Department of Computer Science & Applications, Khalsa College (KCA), City: Amritsar (State-Punjab-INDIA).
E-Mail: rachhpal_kca@yahoo.co.in

ABSTRACT
In multiprocessor system Task scheduling is essential operation. The main objective of task scheduling is to shorten the length of schedule. The effectiveness by doing so is beneficial for large number of calculations having some constraints like time constraints etc. The proposed algorithm has the efficient execution of the schedule on parallel system that takes the structure of the application and the performance characteristics. Number of approximation and heuristics algorithms have been proposed to fulfill the task scheduling problem. It is well known NP-Hard problem. Here the study proposes a genetic based techniques to schedule parallel tasks on heterogeneous parallel system. In this paper the scheduling problem considered includes a new heuristic algorithm for task scheduling, based on evolutionary method which embeds a new fast technique named Stepping Stone into Genetic Algorithm (GA). By comparing the proposed algorithm with an existing GA based algorithm, it is found that the computation time of the new algorithm to find a sub-optimal schedule is decreased; however, the length of schedule or the finish time is decreased too.

Keywords
DAG (Directed acyclic graph), Task duplication, heuristics search, Stepping Stone Method, Parallel Genetic Algorithm.

1. INTRODUCTION
In multiprocessor environment the scheduling parallel tasks is a very complex and tedious job, but it can be make simple by allocating a set of tasks to processors such that the optimal usage of processors and accepted computation time for scheduling algorithm are obtained [1][2][3][4]. The allocation of different tasks into the different processor and the performance of the processors determine the total finish time of the tasks. In such type of environment, it depends upon three major components:
- Evaluation and Performance of the heterogeneous processor,
- Scheduling and Mapping of the tasks onto the processors,
- Sequence of the execution of the tasks on each processor.

These three components to solve the optimization problem [7][30] are highly dependent on each other and should not be optimized separately. Task scheduling in parallel environment concept is being proposed by using the Genetic Algorithm (GA) approach with stepping stone technique. A GA approach [5][8] starts with a generation of individual and produces more generations in iteration mode. The aim of this paper is to present a GA which uses a novel proposed method, named Stepping stone to decrease the computation time for finding a suboptimal schedule i.e to minimize the makespan. However, this new method is general and could be applied to any evolutionary method having GA tool. Reminder of this paper is organized in four sections as follows: An overview of the problem along with brief description of the solution methodology is given in section 2. Detailed proposed genetic algorithm with stepping stone technique explained in section 3. Experimental procedure with results and performance analysis of the methodology is given in the section 4. Conclusion of the research was done in the section 5 at end.

2. PROBLEM DEFINITION
A brief study of classification of parallel algorithms and methods based on the three basic factors
1). Characteristics of the tasks to be scheduled,
2). The parallel environment and
3). The availability of the information[6][9][11][12][13][14].

The complexity of the scheduling problem is very depended to the DAG (Directed Acyclic Graph) [25][28][29], the number of processors, the execution time of tasks and also the performance criteria which would to be optimized. Our main objective in such type of cases depends upon deterministic scheduling problem in which there exist precedence relations among the tasks to be scheduled. A deterministic scheduling problem [16] is one in which all information about the tasks and the relation to each other such as execution time and precedence relation are known to the scheduling algorithm in advance and the processor environment is heterogeneous[23][24][26][27]. The heterogeneity of processors means that the processors have different speeds or processing capabilities according to the given problem. So here, a study has been done regarding the task scheduling problem as a deterministic on the heterogeneous parallel multiprocessor environment. The main objective is to minimize the makespan (execution time + waiting time or idle time). The parallel multiprocessor computing environment or system consists of a set of n heterogeneous processor arranged in a sequence:

\[ P = \{ p_i, i = 1, 2, 3...n \} \]

A fully connected with each other via identical links of three parallel system \( P_1, P_2, P_3 \) is as shown in the Figure 1. DAG has the parallel application and can be represented as, \( G = (T, E, W, C) \), where the vertices set \( T \) consist of \( n \) tasks and are represented as:

\[ T = \{ t_{ij}, j = 1, 2, 3...n \} \]

A directed edge set \( E \) consist of \( k \) edges and all are denoted as:

\[ E = \{ c_{ik}, k = 1, 2, 3...m \} \]

- Figure 1. A fully connected having three parallel processor
A directed edge set \( E \) consist of \( k \) edges ranging from \( k = 1, 2, \ldots, r \) this represent the precedence relationship among the no. of tasks. Suppose any two tasks \( t_1 \) and \( t_2 \in T \) having a directed edge \( e \) i.e. edge from \( t_1 \) to \( t_2 \) which mean that \( t_2 \) cannot scheduled until \( t_1 \) has been completed, \( t_1 \) is predecessor of \( t_2 \), \( t_2 \) successor of \( t_1 \), under the relation of dependency on multiprocessor system. The weights of vertices are denoted by \( W \) as:

\[ W = \{ \text{wij} : i = 1,2,3 \ldots n ; j=1,2,3 \ldots m \} \]

This represent the execution duration of the corresponding tasks and they are varies for same task on different processor due to heterogeneous environment. During experiment we have consider one another factor which represent the communication cost between two processor, if they are scheduled on different processor and notice if both tasks are scheduled on same processor, then this factor is null, denoted by \( C \) as:

\[ C = \{ \text{ck} : k = 1,2,3,\ldots, r \} \]

Data communication between the two tasks can be represented by using this technique i.e. if they are scheduled to different processors, but in case of if both tasks are scheduled to the same processor, then the weight associated to the edge becomes null. There is an example of a complete DAG as shown in the Figure 2. DAG consist of a set of tasks \( T \): \( \{ t_i : i = 1, 2,3 \} \) and Table 1 show a matrix of execution time of each task on different, because of the heterogeneous environment every processor works on different speeds and processing capabilities. It is assumed that processor \( p_1 \) is much faster than \( p_3 \); \( p_1 \) and so on. Processor \( p_2 \) is faster than \( p_3 \), \( p_4 \), and so on. (i.e., the order of speed and processing capabilities can be expressed as \( p_1 > p_2 > p_3 > p_4 \)). As given in Table 1 task \( t_1 \) takes 4 time units to complete their execution on processor \( p_0 \) and takes 9 time units and 10 time units to complete their execution on processor \( p_2 \) and \( p_0 \) respectively. On the basis of the size of the tasks processed on different processors, the execution time has been calculated[12][22].

3. PROPOSED GENETIC ALGORITHM

In the evolutionary system, it is noted that every GAs passes through a cycle of phases and these phases occurs in a sequence as:

1. from the string population creation of strings,
2. string evaluation,
3. best string or strings selection and
4. reproduction to create a new population.

A genetic algorithm (GA) is based on principle of survival of the fittest. Here the concept of chromosome i.e. the individuals are encoded in the population string known as chromosomes. It is possible to evaluate the performance or fitness of individuals in a population after chromosome has been coded. A good coding scheme [17][18] will benefit operators and make the object function easy to calculate. Each individual is assigned a fitness value given by the objective function during the selection operation and choose the fittest individual of the current population to serve as parent of the next generation. Crossover and mutation are the two prime operator of the reproduction. With the advent of crossover and mutation, we can get a sequence of strings in the multiprocessor system in the heterogeneous environment to fulfill the required events procedure and find the final list.

Individual’s fitness in a population generates a good coding scheme [17][18] which will benefit operators. It make the object function easy to calculate. During selection, each individual is assigned a fitness value given by the objective function and choose the fittest individual of the current population to serve as parent of the next generation. Reproduction involves two types of operators namely crossover and mutation. The crossover operator chooses randomly a pair of individuals among those selected previously and exchange some part of the information. The mutation operator takes an individual randomly and works with task duplication heuristics, so that, the total execution time of the schedule should be minimum[10].

A. Initial population:

Following are the list of steps to create and display a list of strings for initial population as:

Step 1: The length of the longest path from the task to an exit task i.e. the task has no child, is the bottom level of a task. There are two conditions occur in the case of bottom level to get a sequence of tasks:

1. The bottom-level is equal to its execution time in case of task has no child.
2. The bottom-level is equal to the maximum bottom-level of its children in case of task has at least one child.

Step 2: Following are the two sub steps to get the sorted list as:

a). Sort the tasks according to their heights in ascending order in a sequence.
b). Sort the tasks with the same height according to their bottom-level in descending order.

Step 3: Repeat step 4 and step 5 until finish of all the tasks and get the sequence of tasks in a particular order.

Step 4: Generate a permutation of processors and sequence of tasks accordingly.

Step 5: Assign tasks to processors in order or in a sequence to get the required output.

The above five steps are repeated for the number of population size. The initial population is initialized with randomly generated individuals. The length of all individuals in an initial population is equal to the number of tasks in the DAG. Each task is randomly assigned to a processor[19][31].

B. Fitness function evaluation:

The objective of the fitness operator in task scheduling is to find shortest possible schedule, the fitness of chromosome is directly related to length of associated schedule. Proposed Genetic Algorithm uses the fitness function which is based on the total completion time for the schedule. It includes execution time and communication delay time. The fitness function separates the evaluation into two parts:

- Task fitness and
- Processor fitness.

The task fitness focuses on ensuring that all tasks are performed and scheduled in valid order. A valid order means that a precedence relations are satisfied i.e. successor task cannot scheduled until predecessor has been completed. The processor fitness component attempts to minimize the processing time. Consider the following schedule S1 and S2 for single processor and multiprocessor parallel system tasks schedules with task size equal to 11 tasks respectively (here, consider the case when fitness function assigned all tasks to a single processor and randomly generated tasks to heterogeneous parallel system.) The processor chosen for scheduler S1 is \( p_1 \), \( p_2 \), \( p_3 \) and the execution time for all
A genetic algorithm (GA) is a search algorithm that can be used to solve problems with the fitness of solutions, and it is a non-random method for solving optimization problems. GA is often used in the task assignment, scheduling, and load balancing in distributed and parallel computing environment. In this section, we present the genetic algorithm technique for the task scheduling problem in the distributed system. We define the fitness function as follows:

\[ f(x) = \sum_{i=1}^{n} w_i \times C_i \]

where \( f(x) \) is the fitness function, \( C_i \) is the cost of the task, and \( w_i \) is the weight of the task.

Table 1: Shows a tasks execution matrix on different processors with task size = 11 as shown in DAG.

<table>
<thead>
<tr>
<th>Task number</th>
<th>Execution time</th>
<th>Height</th>
<th>Bottom-level</th>
<th>Order of execution according to height</th>
<th>Order of execution according to bottom-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>0</td>
<td>72</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>41</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
<td>50</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>0</td>
<td>60</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>1</td>
<td>21</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>1</td>
<td>40</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>2</td>
<td>20</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>2</td>
<td>19</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
<td>2</td>
<td>20</td>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 2: A DAG with 11 different tasks.

The total finish time of scheduler S1 (bottom level) and S2 (high level or maximum level) is:

\[ S1: P_1: t_1(0-50), t_2(50-70), t_3(70-90), t_4(90-110) \]
\[ P_2: t_5(0-5), t_6(50-70), t_7(70-85), t_8(85-115) \]
\[ P_3: t_9(0-15), t_10(50-80), t_11(81-115) \]

\[ S2: P_1: t_1(0-50), t_2(50-60), t_3(60-70), t_4(70-120) \]
\[ P_2: t_5(0-5), t_6(50-70), t_7(70-85), t_8(86-90) \]
\[ P_3: t_9(0-10), t_10(50-55), t_11(56-70) \]

From the above two schedules on three different processors we can see that by changing the policy (priority) of task assignment, the total finish time based on the priority of the tasks’ bottom-level is 115 while that, the total finish time based on the priority of the tasks on the basis of height is 120, which is significantly longer. Now, using the tasks’ bottom-level is considerably better than using the tasks’ height where the number of tasks increases. Therefore, proper fitness function reduces the total finish time very well. Therefore, the fitness values (task and processor) have been evaluated for all chromosomes and the probability of higher fitness is to be selected for reproduction from current generation to the next generation[22][31].

C. Selection operator:

Selection operation is the basic design of fitness function, so how to design the fitness function will directly affect the performance of genetic algorithm. To select the superior and eliminate the inferior, GA uses the selection operator. According to their fitness value individual are selected. Once fitness values have been evaluated for all chromosomes, we can select good chromosomes through rotating roulette wheel strategy. This operator generate next generationby selecting best chromosomes from parents and offspring[20].

D. Crossover operator:

Crossover operator randomly selects two parent chromosomes (chromosomes with higher values have more chance to be selected) and randomly chooses their crossover points, and mates them to produce two child (offspring) chromosomes. We examine one and two point crossover operators. In one point crossover, the segments to the right of the crossover points are exchanged to form two offspring as shown in Figure 3(a) and in two point crossover [19][20], the middle portions of the crossover points are exchanged to form two offspring as shown in Figure 3(b).

E. Mutation operator:

To reduce the idle time of a processor waiting for the data from other processors a mutation operation is designed. It works by randomly selecting two tasks and swapping them accordingly to get the result. Firstly, it randomly selects a processor, and then randomly selects a task [21] on that processor. This task is the first task of the pair to be swapped. Secondly, it randomly selects a second processor (it may be the same as the first), and randomly selects a task. If the two selected tasks are the same task the search continues on. If the two tasks are different then they are swapped over and put these into sequence (provided that the precedence relations must satisfy)[15].
F. The Stepping Stone Technique:

Note that a fixed step with a few stones has a low convergence rate i.e. a high computation time, but it is best to find a sub-optimal schedule. Whereas, the fixed step with a lot of stones has fewer solutions in the search nodes and it is difficult to find a sub-optimal schedule, but it converges fast. So, a new method stepping stone is introduced to solve such type of complex problem. The goal of this technique is to decrease the computation time of the algorithm to find an acceptable sub-optimal schedule. Here in this technique, the number of steps in the first generation is two and after that, by increasing the number of generations, the number of steps increases, until they reach to the population size. At end the convergence is happened and a sub-optimal schedule is founded.

By performing some simulation of a GA for the different DAGs and by studying diagrams of the average finish time of schedules in different generations, it is seen that the stepping stone has suitable behaviour according to the nature of task scheduling problem. Figure 4 shows the average finish time of a GA for a set of 11 random DAGs consist of 20 to 99 tasks for scheduling in 3 to 5 processors where, each simulation has 150 generations. The average finish time for the first generation is 875 time unit, for the 100th generation is 740 time unit and for 200th generation is 725 time unit. If the number of generation is divided to three parts then, the first one-third has the most effectiveness to improve the solutions. The second part improves the solution slowly and in the last one-third, the average finish time usually converges to one value[31].

It is obvious that by using the stepping stone technique, there are more areas in the search nodes at first because the first solutions have more ability of improvement. Then, as the number of generations increases, the ratio of improvement decreases, so this technique causes fewer areas exist in the search nodes and all solutions converge to a sub-optimal schedule [31]. This schedule is as shown in Table 2.

The phases of the proposed Genetic Algorithm (GA) are as follows:

Step 1. Generate a DAG and read all the node values (i.e. to create a task execution matrix). Here n is the number of task and m is the number of processor. Also C is the communication cost and WT is the waiting time.

Step 2. Set some parameters or variables after reading the complete DAG values. The different parameters are like population size says p-size, crossover probability as Cp, mutation probability as Mp and maximum generation value to be computed during the process is Mgen. Let us take generation g=0 before computation and maximum generation Mvalue is also 0.

Step 3. Generate a list of chromosome having its p-size after selecting the chromosome randomly.

Step 4. Calculate the fitness function or value of each chromosome. Let it be fE. Also compute the fitness function or fitness value of each node or task. Let it be fD and it is called task function or node function. At end compute the fitness value or fitness function of each processor from the list of chromosome. Let it be fP and called processor fitness.

Step 5. Perform crossover swapping operation on the chromosome either by using one point crossover or two point crossover from the available list of chromosome with probability Cp.

Step 6. Perform the mutation operation or swap mutation process on chromosome selected with probability Mp.

Step 7. At end apply the last operation of genetic approach called selection process, i.e. select the size of population chromosomes as p-size from the parents and offspring for the next generation.

Step 8. If g=Mgen, then the computed output has the best solution and stop the processing. Otherwise increment the generation i.e. g=g+1 and return to step 4 to find next best solution.

4. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

A set of simulation is performed under one set of common assumptions by using the 7 version of MATLAB tool for comparison of the proposed algorithm and the previous GA. For this purpose a set of 15 graphs consists of 20 to 90 tasks with random execution time are generated. These tasks would be
scheduled on a multiprocessor system with 3 to 6 processors as shown in Table 2[31].

<table>
<thead>
<tr>
<th>Graph number</th>
<th>Number of tasks</th>
<th>Number of processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>45</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>55</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>60</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>65</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>70</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>75</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>80</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>85</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>90</td>
<td>6</td>
</tr>
</tbody>
</table>

After generating the data, the parameters of the both algorithms need to be determined. The number of generation is set to 150, so to stop the process the condition is 150 iterations. In order to achieve a proper search nodes and whereas the number of generation is fixed then, the population size is chosen proportionate to the number of tasks. Each graph in Table 2 is scheduled for both algorithms three times. The average computation time of two algorithms for finding a sub-optimal schedule and their average finish time (fitness values) are calculated for each graph and then, the total average time are obtained for 15 graphs. The average computation time of generation 150 (a sub-optimal schedule) is 18s for GA and is 10s for the proposed algorithm. Therefore, the speed up of the new algorithm is almost 1.8 times of the GA[31].

5. CONCLUSION

Here Genetic Algorithm (GA) has been proposed for task scheduling in heterogeneous parallel multiprocessor system to minimize the finish time including execution time and waiting or idle time and increase the throughput of the system. It is found a better solution for assigning the tasks to the heterogeneous parallel multiprocessor system. After the discussion, experimental results and Genetic Algorithm are compared with previous scheduling methods. Finally we presented a genetic algorithm which uses a new method, named stepping stone technique for the task scheduling problem in multiprocessor systems, with the objective to reduce the schedule length within an acceptable computation time. In order to show the effectiveness of the proposed algorithm, we performed experimental simulations by applying the algorithm to various kinds of task graphs.

6. REFERENCES


[14] David E Culler, “Parallel Computer Architecture”, Published by Morgan Kaufmann & Elsevier India.


