Modified Genetic Algorithm Approach to Optimize Task Scheduling on Heterogeneous Multiprocessor Parallel System using Node duplication

Rachhpal Singh
Asst. Prof., P.G. Department of Computer Science & Applications, Khalsa College (KCA), City: Amritsar (State-Punjab-INDIA).

ABSTRACT

Task graph scheduling is the important factor which occurs in the multiprocessor system. There is problem solving technique called NP complete which is the optimal scheduling of parallel tasks with some precedence relationship onto parallel machines and can be solved only by using heuristic approach. The execution time requirements of the applications tasks are assumed to be stochastic. Genetic algorithms are the widely used technique for constrained optimization. Performance of genetic algorithm can be improved by using the modified genetic algorithm (MGA) having top level and bottom level approach. The complexity of the problem increases when task scheduling is to be done in a heterogeneous environment, where the processor is the network may not be identical and take different amounts of time to execute the same task. In this paper the concept of Modified Genetic Algorithm with Node duplication (NMGA) based on bottom level and top level approaches is used. It also exhibits the efficiency of Node duplication modified genetic based techniques by comparing against some deterministic scheduling technique like genetic algorithm, modified genetic algorithm, first come first serve (FCFS) approach, priority algorithm for minimizing inter processor traffic communication.

Keywords

Genetic algorithm, task scheduling parallel system, DAG (Directed Acyclic Graph), Node duplication Modified genetic Algorithm (NMGA).

1. INTRODUCTION

The last two decades have witnessed great breakthrough in the use of parallel processor system in a wide variety of applications consequently with improvements in device technology, computer architectures, and theory and software tools. There are many problems that need to be addressed which will keep the research community busy for years to come [1].

In the operation of parallel computing system scheduling is the major hurdle. Scheduling is to simply allocate a set of tasks/jobs in such a way so as to obtain the optimum performance. The multiprocessor scheduling problem is to allocate the tasks of a parallel program to processor in a way that minimizes its completion time. This problem has received considerable attention over the last few decades because the efficiency of execution of a parallel program critically depends on the schedule of these tasks. However it is known to be NP complete for the general case and even for many restricted cases [2]. For this reason many heuristic algorithms have been developed to obtain suboptimal solutions under the assumptions of various system models [3].

Here is a proposal of modified genetic algorithm with node duplication (NMGA) to solve the problem of multiprocessor task scheduling. Three unique features distinguish this NMGA from a Genetic Algorithm (GA) or a traditional GA are as follows:

i). First it uses a flexible representative style which allows the GA to evolve both the structure and the value of the solutions. The flexibility is expected to improve GA’s ability to identify and retain good building blocks.

ii). Second this GA uses a dynamically incremental fitness function which starts out rewarding for simpler goals, gradually increasing the difficulty of the desired fitness value or goals until a full solution is found. [4].

iii). Next during the scheduling with node duplication it might happen that some nodes are unnecessarily duplicated. But still schedule remains valid without any effect on its length. [5]. As a result our GA places no restrictions on the individuals that can be formed and does not require special operator or repair mechanisms to ensure validity. Rather, it attempts to give partial fitness for invalid individuals that contain some valid subsequences of tasks and encourages the successively longer valid subsequences.

In this paper, section 2 gives pertinent preliminaries i.e. problem methodology. Section 3 gives an overview of genetic algorithms with all the genetic operators related with this heuristic search, section 4 presents our proposed work i.e. implementation of modified genetic algorithm with node duplication. Section 5 discusses the experimental and performance analysis. Section 6 closes the paper with concluding remarks and ways for further research. Rest section is used for references.

2. PROBLEM METHODOLOGY

On the basis of the multiprocessor system and the availability of information, Multiprocessor scheduling problem can be classified into many different classes based on characteristics of the program and tasks to be scheduled. H. El-Rewiini et. al. [6] gives a information about task scheduling in parallel and distributed system which is a big and general taxonomy of scheduling problems and discusses differences between classes. Our main focus is on a static scheduling problem in which there exist precedence relations among the tasks to be scheduled and in which task duplication is allowed. A static scheduling problem is one in which all information about the tasks and their relations to each other such as execution time and precedence relations are known to the scheduling algorithm in advance. In dynamic scheduling problem in which some information about tasks and their
relations may be undeterminable until runtime i.e. task execution time and precedence relations may be determined by data input. Here in this paper we will discuss the task scheduling problem in two different ways as deterministic and non-preemptive in a heterogeneous multiprocessor system to minimize the execution time with the waiting time or idle time. Let us take a multiprocessor system set consists of \( n \) heterogeneous processors and can be described as:

\[
P = \{ p_i, i=1,2,3,4, \ldots, n \}
\]

All the above processors are fully connected with each other via identical links and all the fully connected three parallel systems with identical links are as shown and well described in the Figure 1 as shown below. As there are in the heterogeneous environment, every processor works on different speeds and processing capabilities. We consider the speeding parameters of processor as \( P_1 \) is much faster than \( P_2 \) and \( P_3 \), i.e. \( P_1 < P_2 < P_3 \).

We represent a parallel program as task precedence graph model (or simply the DAG). A Directed Acyclic Graph (DAG) is represented as \( G=(V,E) \), where \( V \) is the set of nodes which represents the tasks and \( E \) is the set of directed edges which represent the execution dependencies as well as the amount of communication (Communication cost) is commonly used in static scheduling of a parallel program with tightly coupled tasks on multiprocessors. Hereafter we will use the terms node and task interchangeably. For example, in the task precedence graph shown in Figure 2, task \( n_8 \) cannot commence execution before task \( n_3, n_4 \) and \( n_5 \) finishes its execution and gather all the communication data from \( n_3, n_4 \) and \( n_5 \).

The scheduling objective is to minimize the program completion. Task scheduling is to be done in a heterogeneous environment, where the processor in the network may not be identical and take different amounts of time to execute the same task which is shown in Table 1. All processors are assumed to be fully connected. This is the model we use in this approach.

Let us suppose \( N \) connections of \( m \) non precedence tasks as

\[
N = \{ n_i, i=1,2,3,4,5, \ldots, m \}
\]

In this directed acyclic graph as all the edges are linked and a directed edge set \( E \) consists of \( e_i \) edges ranging from \( e_i=1,2,3, \ldots, k \). This set of edges represents the precedence relationship among the number of tasks in the multiprocessor environment. Suppose any two nodes or tasks \( n_1 \) and \( n_2 \) belongs to or the subset of \( N \) having directed edge \( e_i \), i.e. edge from \( n_1 \) to \( n_2 \) which mean that \( n_2 \) cannot schedule until \( n_1 \) has been completed, \( n_1 \) is precedence of \( n_2 \) or \( n_2 \) successor of \( n_1 \), under the relation of dependency on multiprocessor system as shown in the figure 2. The weight of the vertices is denoted by \( W \) which is the execution duration of the corresponding tasks and varies for the same task on different processor due to the multiprocessor heterogeneous environment and it is described as below:

\[
W = \{ W_{ij}, i=1,2,3, \ldots, p; j=1,2,3, \ldots, q \}
\]

The main objective is to minimize the total task completion time i.e. the combination of execution time and the waiting time or idle time in such type of multiprocessor system.

---

**Figure 1**: A fully connected parallel processor

**Figure 2**: Task precedence Graph (DAG- Directed Acyclic Graph with Task precedence)

**Table 1**: Execution cost matrix or Task execution matrix on different processor

<table>
<thead>
<tr>
<th>( n_1 )</th>
<th>( n_2 )</th>
<th>( n_3 )</th>
<th>( n_4 )</th>
<th>( n_5 )</th>
<th>( n_6 )</th>
<th>( n_7 )</th>
<th>( n_8 )</th>
<th>( n_9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>10</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>12</td>
<td>14</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>11</td>
<td>9</td>
<td>2</td>
<td>15</td>
<td>10</td>
<td>12</td>
<td>11</td>
</tr>
</tbody>
</table>
During experiment, consider another factor which represents the communication cost between two processors, 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:

$$C = \{ C_k, k=1,2,3,4,\ldots\ldots, m \}$$

3. GENETIC ALGORITHM:

In the parallel and distributed computing the most popular heuristic search algorithm is Genetic Algorithm (GA) and it is based on the principles of evolution and natural genetics. It combines the exploitation of past results with exploration of new areas of search space. By using survival of the fittest techniques and a structured yet randomized information exchange form a search algorithm. In other words genetic algorithm can mimic some of the innovation flair of human search. It is a randomized algorithm but not simple random walks. The goal of the Genetic algorithm to maintain a population of candidate solutions that evolves over time and ultimately converges. Individuals in the population are represented with chromosome. Each individual has numeric fitness value that measures how well this solution solves the problem. Below are given basic steps of a typical genetic algorithm. Actually genetic algorithm contains three operators to solve a problem and these are selection operator, crossover operator and mutation operator. The selection operator selects the fittest individuals of the current population to serve as parents of the next generation. The crossover operator chooses randomly a pairs of individuals and exchanges some part of information. The mutation operator takes an individual randomly and alters it. The probability of applying mutation is very low while that of crossover is usually high. To improve the fitness of its individuals, there be need of a population which evolves iteratively. At end genetic algorithm is executed until some termination condition is achieved, such as the number of iteration, execution time results stability etc. to find the better results. To optimize the solution of the given problem following are the steps used in the Genetic Algorithm are as:

Step I: First of all Initialize the population.

Step II: Repeat all the steps from (a) to (d) until termination condition not satisfied as given below:

(a) Evaluate the current population which is currently available.

(b) Select the parents for swap crossover and for mutation operation.

(c) Apply Genetic Operator to parents to create offspring from the given population.

(d) Set current population equal to the new offspring population.

Step III. Stop end terminate the operation.

Initial population:

Initial population is the first step occurs in solving any application of the popular Genetic algorithm. In the initial population creation, requirement of number of processor in multiprocessor environment, the number of tasks and size of population are the major parameters. Exactly one copy of each tasks is used by the each individual. It means the repetition of task on individual processor is not allowed and we can arrange them like length of all individual in an initial population is equal to number of tasks in the target DAG and during experiment each task is randomly assigned to the processor. Each individual of the initial population is generated through a minimum execution time along with b-level (bottom level) or t-level (top level) precedence resolution to avoid the problem of same execution time or completion time.

Fitness function computation:

To find the shortest possible schedule in the task scheduling fitness operator is very important as the fitness of the chromosome is directly proportional to the length of associated schedule. In other words it is to minimize the makespan i.e. the time when finishes the latest job. Various factors including throughput, turnaround time and processor utilization etc. contribute in the task scheduling but the fitness function used genetic algorithm targets the total finish time for the required schedule including execution time and communication delay time ultimately to reduce the execution time for a schedule. Fitness of task which equips us with the knowledge of all the tasks that are extended and scheduled. Here we consider some general schedule of tasks S1 and S2 on processor system for the uniprocessor and multiprocessor.

Consider in S1 schedule all tasks execute on P1 (in the case of uniprocessor system) and in S2 schedule tasks are randomly distributed on all processor P1, P2 and P3 (In the case of multiprocessor system) on all processors as execution time shown in Table 1.

$$S1 : n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow n_4 \rightarrow n_5 \rightarrow n_6 \rightarrow n_7 \rightarrow n_8 \rightarrow n_9 \rightarrow n_{10} \rightarrow n_{11}$$

In case of uniprocessor system total finish or execution time is 86 time unit and communication cost is 0.

$$P_1 : n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow n_4 \rightarrow n_5 \rightarrow n_6 \rightarrow n_7 \rightarrow n_8 \rightarrow n_9 \rightarrow n_{10} \rightarrow n_{11}$$

$$S2 : P_1 : n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow n_4 \rightarrow n_5 \rightarrow n_6 \rightarrow n_7 \rightarrow n_8 \rightarrow n_9 \rightarrow n_{10} \rightarrow n_{11}$$

In the Figure 3 there are three processors P1, P2, P3 and horizontal axis shows time. In this figure WT represents the waiting time. CT represents the communication time of processor and bar with numeric data (below the line) shows the starting and finishing time of the task and name of the task (above the line).

Total finish time = execution time + communication cost = 75 time units.

Note that the schedule scheme in S1 represents total finish time of 86 time units and S2 shows 75 time units i.e. the proper calculation of fitness operator function reduces the total finish time.
Selection Operator:
The selection technique is based on the roulette wheel method \[8\]. The probability of a string going forward to the next generation is represented as a proportional sized slot on the roulette wheel with a range from 0 to 1 which generates random numbers from 0 to 1. The string which corresponds to the randomly selected slot is brought forward to the next generation. The probability of the fitter strings to be brought forward to the next generation is more as they have larger slots. This process continues until a sufficient number of strings are selected \[9\]. So in Genetic Algorithm, role of selection operator is for the selection of good or superior chromosome vector and remove or eliminate the inferior or bad chromosome from the list. Selection operator functions to select the good chromosome vector and filter out the bad chromosome from the available list of chromosome. Fitness function value plays a critical role in selecting good or bad vectors.

Crossover Operator or swap Crossover:
In case of chromosomes in the matting pool, swap crossover/crossover operator is useful. It works on two chromosomes to produce two offspring chromosome, each with genetic material from both parents. For each parent, swap crossover randomly chooses one substring to apply the crossover on it. On each chosen substring, swap crossover locates two crossover points \[10\] and generate a new list. One point crossover and two point crossover are the two major types of swap crossover process or crossover operator in the genetic scheduling algorithm.

Mutation Operator:
The probability that a task will be changed is depicted by mutation. As a result, the expected number of mutations per individual is equal to the mutation rate multiplied by the length of an individual chromosome. If a task is selected to be mutated, then either the task number or the processor number of that task will be randomly changed \[11\]. In this paper, we are using partial gene mutation which randomly selects a chromosome and changes a randomly selected gene \((N_i, P_i)\) to \((N_j, P_j)\) for which avail time \(P_j\) is minimum. It means time is minimum for all the processor for task \(N_i\). It is to be seen that partial genes mutation changes the processor which a task is assigned to, whereas crossover assigns a set of tasks, to probably, different processors. In this scenario, partial gene mutation was applied in conjunction with crossover operator \[12\]. So here the job of the mutation operator is designed to reduce the idle time or waiting time.

4. MODIFIED GENETIC ALGORITHM WITH NODE DUPLICATION (NMGA):
Genetic Algorithm operates through a simple cycle of stages creation of population strings, evaluation of each string, selection of the best strings and reproduction to create a new population. The number of genes and their value in each chromosome depends on the problem specification. Here the number of genes of each chromosome is equal to the number of nodes (Tasks) in the DAG and the gene value demonstrate the scheduling priority of the related task to the node where the higher priority means that task must be extended early. Modified genetic algorithm uses the concept of top level and bottom level on multiprocessor system.

**T-level \((T_i)\) or top level:**

It is defined to be the length of the longest path in the task from entry task toTi excluding the execution cost Ti \[7\].

\[
T-\text{level}(T_i) = \max_{Tj \in \text{pred}(T_i)} \{ \text{level}(T_j) + w_j + c_{ij} \}
\]

**B-level \((Bi)\) or bottom level:**

It is the length of the longest path from Bi to an exit task scheduling the execution cost of Bi. Here equations 1 and 2 are the equation definitions of t-level \((T_i)\) and b-level \((Bi)\) \[7\].

\[
B-\text{level}(T_i) = w_i + \max_{Tj \in \text{success}(T_i)} \{ c_{ij} = \text{b-level}(T_j) \}
\]

Top level provides the group search while bottom level provides the individual search.

**Node duplication:**
As there is major problem in scheduling to avoid inter processor communication, a node often has to wait for entering in the communication. As shown in the figure 2 node \(N_2\) and \(N_3\) are delayed due to communication from node \(N_1\) and both processor \(P_2\) and \(P_1\) run idle during that time. Now after node duplication communication from node \(N_2\) is local to each processor as shown in the figure 4 of the target system and node \(N_i\), \(N_j\), \(N_k\) and \(N_t\) can start immediately after node \(N_i\) finish. Node duplication is employed in a large variety of scheduling approaches by Ahmed and Kwok \[17\]. In the basic genetic algorithm the initial population is generated randomly which can cause to generate more bad results. Here we introduce the node duplication concept with the modified genetic algorithm based on heuristic. To avoid the generation of non-optimal results, heuristic approach can be applied to generate the initial population that gives better results in terms of quality of solutions. During scheduling with node duplication it might happen that some nodes are unnecessarily duplicated. But still schedule remains valid without any effect on its length \[13\] \[14\]. The steps of the modified and improved parallel genetic algorithm are:

**Step 1:** Read the DAG:

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 the parameters:

Let us take 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 \(M\) value is also \(0\).

**Step 3:** Initialize:

Generate a list of chromosome having its \(p\)-size after selecting the chromosome randomly.

**Step 4:** Compute Fitness function:

Calculate the fitness function or value of each chromosome. Let it be \(f_x\). Also compute the fitness function or fitness value of each node or task. Let it be \(f_y\) 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 \( f_p \) and called processor fitness.

Step 5: Crossover swapping or Crossover Operation:

Perform the crossover swapping or operation on the chromosome either by using one point crossover or two point crossover from the available list of chromosome with probability \( C_p \).

Step 6: Mutation process:

Perform the mutation operation or swap mutation process on chromosome selected with probability \( M_p \).

Step 7: Final selection:

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: Testing and Stop:

If \( g = M_{gen} \), 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.

5. PERFORMANCE ANALYSIS:

The task priority list on the basis of DAG as shown in the figure 2 having heterogeneous parallel processor with node duplication is as shown in the figure 4.

![Figure 4: Priority list of DAG in case of heterogeneous multiprocessor system](image)

Our main objective is to reduce the total finish time because here we have obtained a 72 time unit from 86 time units in case of uniprocessor system. During FCFS policy tasks are assigned to three different processor as follows:

**S3:**

- \( P_1: n_1 \rightarrow n_3 \rightarrow n_8 \rightarrow n_{10} \)
- \( P_2: n_2 \rightarrow n_4 \rightarrow n_6 \rightarrow n_7 \)
- \( P_3: n_5 \rightarrow n_9 \rightarrow n_{11} \)

After the analysis of execution time of FCFS of schedule S3 for processor \( P_2 \), we get the total finish time 42 time units. Note that during priority scheduling without any preemption, we need some more data as priorities of node as shown in the figure 1. In scheduling pipe, Priority scheduling tasks are assigned priority value and after assigning the values according to their priority value, tasks occupy their space in different processor as shown below:

- \( P_1: n_1 \rightarrow n_3 \rightarrow n_8 \rightarrow n_{10} \)
- \( P_2: n_1 \rightarrow n_3 \rightarrow n_4 \rightarrow n_8 \)
- \( P_3: n_5 \rightarrow n_9 \rightarrow n_{11} \)

After discussing the above execution schedule, we get the total finish time 72 time units.

Later on after the implementation of modified GA (MGA) has the best schedule at \( t \)-level is as follows:

- \( P_1: n_1 \rightarrow n_2 \rightarrow n_6 \rightarrow n_{10} \)
- \( P_2: n_3 \rightarrow n_5 \rightarrow n_8 \)
- \( P_3: n_9 \rightarrow n_{11} \)

The total finish time is 37 time units.

Later on after the implementation of modified GA (MGA) has the best schedule at \( b \)-level is as follows:

- \( P_1: n_1 \rightarrow n_2 \rightarrow n_7 \)
- \( P_2: n_3 \rightarrow n_5 \rightarrow n_{10} \)
- \( P_3: n_9 \rightarrow n_{11} \)

The total finish time is 35 time units.

During the implementation of modified genetic algorithm with concept of node duplication having \( b \)-level and \( t \)-level collection of tasks as shown in DAG considered above has the total finish time as:

- The optimal schedule for the processors in case NMGA (at \( t \)-level):
  - \( P_1: n_1 \rightarrow n_2 \rightarrow n_6 \rightarrow n_{10} \)
  - \( P_2: n_3 \rightarrow n_5 \rightarrow n_{10} \)
  - \( P_3: n_9 \rightarrow n_{11} \)

The total finish time is 33 time units.

The optimal schedule for the processors in case NMGA (at \( b \)-level):

- \( P_1: n_1 \rightarrow n_2 \rightarrow n_7 \)
- \( P_2: n_3 \rightarrow n_5 \rightarrow n_{10} \)
- \( P_3: n_9 \rightarrow n_{11} \)

The total finish time is 30 time units.

To compute speed up and efficiency (Performance Analysis):

**Speed up (N_sp):** Speedup is defined as “It is the ratio of completion time on uniprocessor system to (or divided by) completion time of multiprocessor system”. Here the speedup of uniprocessor system is 86.

**Efficiency (N_e):** Efficiency of node is defined as “It is the multiplication of speed up with 100 and the resultant value is divided by the number of processor in the multiprocessor system”. In other word it is as:

\[ N_e = \frac{(N_{sp} \times 100)}{n} \]

where \( n \) is the number of processor in the multiprocessor system.
The comparison chart of the various algorithms having speedup and efficiency is computed as below:

<table>
<thead>
<tr>
<th>S#</th>
<th>Types of Algorithm</th>
<th>Speed up N_p</th>
<th>Efficiency N_e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FCFS</td>
<td>86/42= 2.047</td>
<td>68.3%</td>
</tr>
<tr>
<td>2</td>
<td>Priority scheduling</td>
<td>86/72= 1.194</td>
<td>39.8%</td>
</tr>
<tr>
<td>3</td>
<td>MGA at t-level</td>
<td>86/37= 2.324</td>
<td>77.4%</td>
</tr>
<tr>
<td>4</td>
<td>MGA at b-level</td>
<td>86/35= 2.457</td>
<td>81.9%</td>
</tr>
<tr>
<td>5</td>
<td>NMGA at t-level</td>
<td>86/33= 2.606</td>
<td>86.7%</td>
</tr>
<tr>
<td>6</td>
<td>NMGA at b-level</td>
<td>86/33= 2.866</td>
<td>95.6%</td>
</tr>
</tbody>
</table>

6. CONCLUSION:

After analysis and study, in this paper the proposed Modified Genetic Algorithm for the task scheduling in heterogeneous parallel multiprocessor system using Node duplication which is used to minimize the completion and by improving the throughput (work done) of the system by maximizing it. After comparing all the system like uniprocessor system, FCFS, Modified Genetic algorithm at b-level and t-level, we find that NMGA at b-level and t-level is the best. The main factor here is which task should be duplicated to reduce the overall time. In future, we can attempt to implement this proposed NMGA method also for the problem of non-deterministic homogenous multiprocessor used for real time.

REFERENCES:

10. Mohammad Daoud, Nawwaf Kharma, “GATS 1.0: A novel GA-based scheduling algorithm for task scheduling on heterogeneous processor nets”.