Journal of
Information Technology & Software Engineering

Sharma and Kaur, J Inform Tech Softw Eng 2015, 5:2
DOI: 10.4172/2165-7866.1000151

Research Article Open Access

# An Efficient Task Scheduling of Multiprocessor Using Genetic Algorithm Based on Task Height

**Ashish Sharma* and Mandeep Kaur**

*Department of Computer Science and Engineering , Guru Nanak Dev University, Regional Campus Jalandhar, India*

**Abstract**

Static task scheduling in multiprocessor frameworks is one of the well-defined NP hard problem. Due to optimal utilization of processors and in addition investing less time, the scheduling of tasks in multiprocessor frameworks is of extraordinary significance. To solve NP hard problem using traditional strategies takes reasonable measures of time. Over the time, various heuristic procedures were presented for comprehending it. Therefore, heuristic methods such as genetic algorithms are appropriate methods for task scheduling in multiprocessor system. In this paper, a new GA for static task scheduling in multiprocessor systems has been presented whose priority of tasks' execution is based on the height of task in graph and other mentioned parameters and then scheduling is performed. This proposed method is simulated and then compared with basic genetic algorithm.

**Keywords:** Multi processor; Genetic algorithm; Schedule; Task graph; Distribute system; Task height; Make span; Schedule time

## Introduction

It is difficult to execute a single big problem on a single processor in a reasonable time. Because of this, it is divided into number of tasks and the length of each schedule must be so that it results in appropriate scheduling in a multiprocessor system. The task scheduling [1,2] problem in multiprocessors is to allocate the set of tasks to processors such that optimal or sub optimal utilization of processors and minimum scheduling time are achieved.

For mathematical modeling if task assignment problem, direct acyclic graph (DAG) is used as each task is represented by its adjacent node in the graph [3]. The edge between task ti to task tj represents that task ti is not finished, task tj cannot start execution. The task assignment objective is to schedule the n tasks to m processors, while the priority of tasks is being taken care and the utilization is maximized. The communication delay between processors and the data volume transmitted between two tasks is also observed. Scheduling of tasks in a multiprocessor system is an NP hard problem [4]. It is expensive and time consuming to schedule tasks using traditional and dynamic approaches. However using heuristic methods we can have the optimal solution or nearly optimal solution for these problems. There are many heuristic methods which can be used such as: min–min, max min, duplex, minimum completion time (MCT), minimum execution time (MET) [5], simulated annealing (SA) [6], tabu search [7]. One of the heuristic method which is best for scheduling in multiprocessors system is genetic algorithm (GA).

## Review of Genetic Algorithm Literature for Multiprocessors Task Scheduling

As scheduling is an NP-complete problem researchers try to apply heuristics or meta-heuristics to get optimum or near to optimum solution. People use a single heuristic or a combination of heuristics and meta-heuristics. It is called the hybrid meta-heuristics.

Ahmad et al. [8] have proposed performance effective genetic algorithm (PEGA). The PEGA efficiently finds the best solution from the search space; The reason why PEGA is performance effective is because of the effective utilization of genetic operators (crossover and mutation).

Agarwal and Colak [9], proposed a metaheuristic approach - called NeuroGenetic - which is a combination of an augmented neural network and a genetic algorithm. The results showed that the neuro genetic approach performs better than either the augmented neural network or the genetic algorithms alone.

Parvan et al. [10], proposed a hybrid scheduling algorithm for solving the independent task scheduling problem in grid which comprises of GA with Firefly algorithm. It was indicated from the results that as compared to the best processed method, the proposed algorithm can decrease Makespan of 10%.

Mehrabi et al. [11], solved the task assignment problem by considering load balancing with the use of a new method, based on the genetic algorithms (GAs). GA uses a repair function to ensure valid assignments during the process of algorithm.

Devi and Anju [12], in their study emphasize on development of a multi objective scheduling algorithm using Evolutionary techniques for scheduling a set of dependent tasks in a multiprocessor environment which minimizes the makespan and reliability cost. NSGA-II is Elitist Evolutionary algorithm that takes the initial parental solution without any changes, in all iteration to remove the problem of loss of some pareto-optimal solutions.

Diana et al. [13], proposed improved genetic algorithm (IGA) based approach for the single mode resource constrained project scheduling problem (RCPSP) with makespan minimization as objective. The suggested way uses binary string based representations and operators for chromosomes.

Awadall et al. [14] highlights two new approaches, modified list scheduling heuristic (MLSH) and enhanced genetic algorithm by constructing promising chromosomes. The result after comparison shows that the proposed approaches works to enhance processor efficiency and decrease task makespan.

***Corresponding author:** Ashish Sharma, Department of Computer Science and Engineering Guru Nanak Dev University, Regional Campus Jalandhar, India, Tel: +91-183-2258802; E-mail: Iamashish90@gmail.com

**Received** May 25, 2015; **Accepted** June 26, 2015; **Published** July 06, 2015

**Citation:** Sharma A, Kaur M (2015) An Efficient Task Scheduling of Multiprocessor Using Genetic Algorithm Based on Task Height. J Inform Tech Softw Eng 5: 151. doi:10.4172/2165-7866.1000151

Dhingra et al. [15], have proposed that Optimizing different parameters such as crossover, mutation, crossover probability, selection function etc. can lead to efficient and effective genetic algorithm.

## Proposed Algorithm

We propose a genetic algorithm for scheduling tasks in multiprocessor environment. We have used the randomized height based approach. Below we have listed the steps of our proposed GA for tasks scheduling.

- Find height for each task in DAG
- Encoding chromosome
- Initial population
- Generate population
- Repeat (adjust height)
- Fitness Function
- Apply GA operators
- Update population
- Until stopping condition

### Find height for each task in DAG

Height of each task can be calculated using DAG, if task is parent of other task then its height is 1- height of its child or height of child is 1+ height of its parent. Below is the equation that calculates the height of task.

Height [task]=0 (if task is root)

Otherwise

Height [task]=max (height (parents))+1

Calculation of height starts from entry node and ends at exit node (Figure 1 and Table 1).

### Encoding chromosomes

Chromosome shows the possible state of scheduling. Each chromosome is represented by the tuple (T,P), where T is set of tasks and P is set processors in the framework. pi ϵ P is shown a processor that allocated to task ti ϵ T. Each chromosome represents the execution of tasks in the processor and also the sequence of tasks to processor [16]. In below table there are three processors and 7 tasks that are shown Table 2.

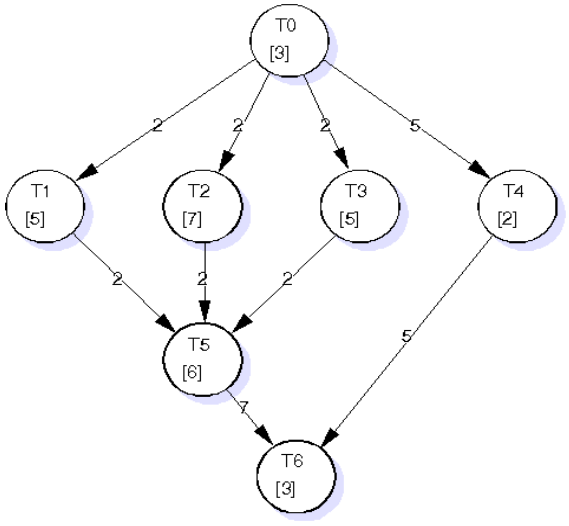


**Figure 1:** Directed acyclic graph.

### Generation of population

To generate population perform the steps shown below:

Repeat

Input number of tasks

Input number of processors

Check order of tasks

For i=0 to number_of_task

If (height [task[i]]=1)

Readytoexecute []=task[i]

end if

end for loop

Generate a random number x

Select readytoexecute[x]

Generate a random number y

Select machine [y]

adjust_height()

Repeat until there is no task with height 1

### Adjust height

Tasks are queued in priority queue on the basis of their priorities. As it is mentioned above, task must be ready to execute and must have the higher priority and stand on priority queue. Adjust height function arranges tasks in all the possible ways. Using this method we update the task height which are dependent on the selected task. This change is just limited to this population only as it is the local height concept. In further populations global height parameter is considered.T1 is root and if selected first then suppose its height is updated to 0 then height of all children must be updated accordingly (Table 3).

After selecting each task for execution we update this table so that we can find the dependencies and arrange them in respective order. Scheduling is done based on tasks priority and in GA execution time, the sequence of tasks priority is not changed but the sequence of allocation of processors to the tasks is variant.

| **Task** | T0 | T1 | T2 | T3 | T4 | T5 | T6 |
|---|---|---|---|---|---|---|---|
| **Height** | 1 | 2 | 2 | 2 | 2 | 3 | 4 |

**Table 1:** Tasks Height according to DAG.

| **Task** | T0 | T1 | T2 | T3 | T4 | T5 | T6 |
|---|---|---|---|---|---|---|---|
| **Processor** | 1 | 3 | 2 | 3 | 1 | 2 | 1 |

**Table 2:** Sequence of tasks to processors.

| **Task** | T0 | T1 | T2 | T3 | T4 | T5 | T6 |
|---|---|---|---|---|---|---|---|
| **Height** | 0 | 1 | 1 | 1 | 1 | 2 | 3 |

**Table 3:** Adjusted task height.

### Fitness function

The aim of the proposed scheduling algorithm is to minimize the schedule time of the parallel program. It is the measure which decides the fitness of function. It is calculated in terms of minimum schedule time so that utilization and speed up can be achieved [17]. We consider the computation cost [18], communication delay between processors and data volume transmitted between two tasks as well. We have calculated the starting time and finish time of tasks.

We derive a formula to calculate starting time (ST) of a task which is show below.

ST(task)=max (max[task at processor [id] [level order[task]], max time on machine)

The finish time (FT) of that task is summation of starting time and its execution time(ET). Its completion time is the available time for a machine on which it is executed.

FT(task)=ST (task)+ET( on particular machine)

Fitness=max(FT(Pj)

for j=1,2,...,n

Where n is the number of processors and *FT(Pj)* is the finish time of the final task in the processor *Pj*.

### Reproduction

For reproduction, the pair of available solution is selected and then the reproduction operators like crossover and mutation are applied. All operators are discussed below.

**Selection:** The selection phase has two steps:

1) Applying a roulette wheel to select two chromosomes: After chromosomes are ascend on the basis of their fitness, a roulette wheel series is constructed based on their fitness [17]. Hence, the chromosomes with low fitness, occupy more slots in the roulette wheel. In this way the possibility of selecting chromosomes with best fitness is higher. Then two chromosomes will be selected.

2) Applying a roulette wheel for selecting a task: A roulette wheel is constructed for tasks based on their number of children. A task with more sub tasks has more probability to be selected compared to a task with fewer ones. The genetic operators like crossover, mutation will be applied on the current generation to produce the next generation.

**Crossover:** Crossover operator is used to differ the analogy of chromosomes from one generation to another. Single point crossover is used in proposed algorithm. Crossover is applied on the chromosomes so as to produce the best fit chromosome which has the best fitness than its parent. The crossover starts with two parent chromosomes to exchange subparts of them to create two new children chromosomes.

Following conditions must be satisfied for the reproduction of legal chromosome:

1) The height of the tasks next to the crossover points should be different.

2) The height of all the tasks immediately in front of the crossover points should be equal.

**Mutation:** The mutation selects a chromosome and then randomly exchanges the two tasks with the same height [17]. The mutation is applied with a certain mutation rate (Mr) which is used to prevent the search process from converging to the local optima prematurely.

## Scheduling Algorithm

The code of proposed scheduling algorithm is shown below.

1. Generate the population

2. While (finish time of each machine in chromosome are equal or generations number is no more than given number) do

For every chromosome in this population do

calculate fitness of every chromosome

End for

While next population is generated and complete do

choose chromosomes with best fitness values by roulette-selection

use single point crossover operator to create next generation children

use mutation operator

End while

End while

The exit condition in proposed algorithm determines the equal processing time for all processors and also the number of generations. By using it the optimal parallelism can be achieved, as all processors can have the equal or nearly equal processing time

## Implementation and Experimental Results of Proposed Algorithm

### Implementation environment

A set of simulation is performed using Java on Eclipse platform on a computer Pentium IV, having AMD processor 2.8 GHz, and 512 MB memory of RAM to evaluate our suggested algorithm. To load tasks and processors we used input file in which the number of tasks and number of machines are described. The communication delay between processors, computation cost(Execution time of tasks) and data volume between two tasks is also mentioned in input file. Create population method is used to create population of 1000 chromosomes. Height of Node (int) method is used to determine the height of each task and also adjusts the height. Fitness (int id) checks the fitness (makespan) [14] of each task. Schedule task class is used to define and manage tasks, processors, starting time and finish time of each task .procAvT defines the processor available time. The task group is J30 and all communication delays and costs are defined in this group.

### Results and comparison

Following Figures shows the results achieved [19] (Figures 2-5). The speedup and utilization and schedule time using this technique are better than basic GA. Figure 3 shows the schedule time using our proposed algorithm and schedule time using basic genetic algorithm. It is clear that our algorithm is better than basic GA.

## Conclusion

We proposed a height base GA to solve task scheduling problem of dependent tasks in multiprocessor architecture. In this algorithm priority of tasks is based on their height, children number and execution time [20]. The experimental simulations applied on our algorithm using various task graphs and number of generations and by comparing it with Basic GA shows that proposed algorithm has less schedule time than basic GA. Under the same the schedule length, finish time are

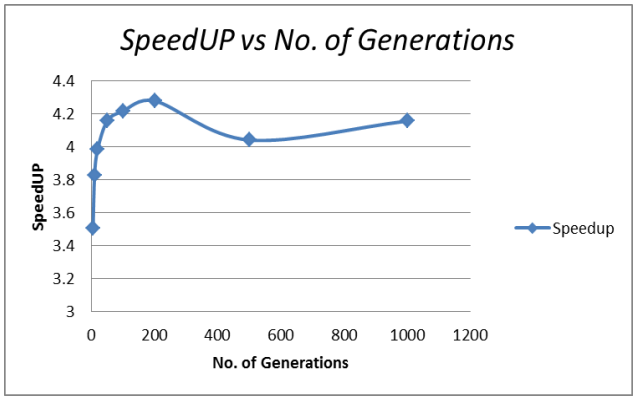


**Figure 2:** Speedup vs. Number of generations.

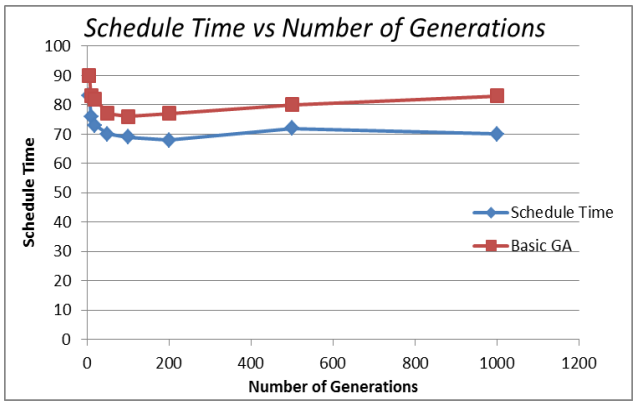


**Figure 3:** Schedule Time vs. Number of generations.

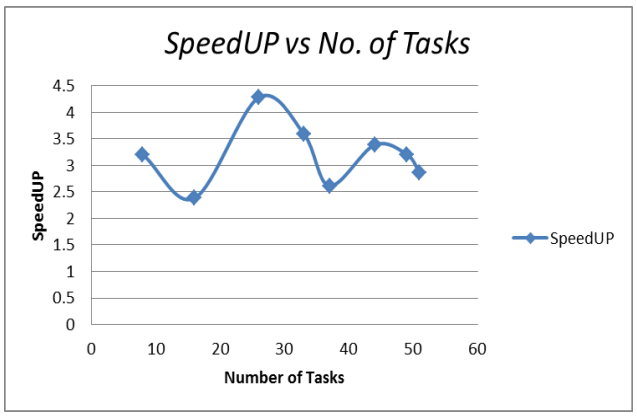


**Figure 4:** Speedup vs. Number of tasks.

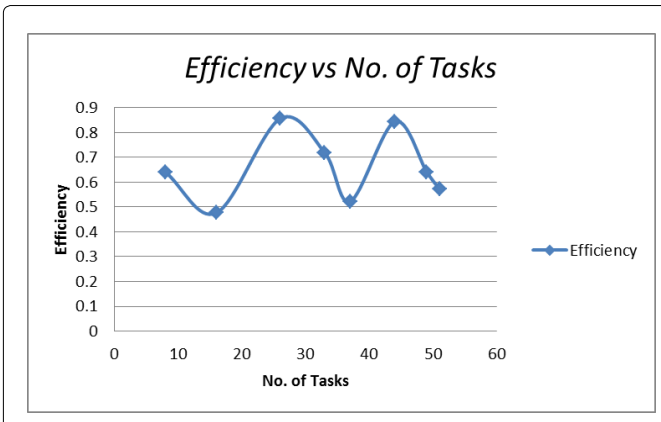


**Figure 5:** Efficiency vs. Number of tasks.

also less. Therefore proposed algorithm can show effective behaviour and can be used as an efficient and better scheduling system in many applications.

## References

1. Rewini HE, Lewis TG, Ali HH (1994) Task scheduling in parallel and distributed systems. Prentice Hall.
2. Kwok Y, Ahmad I (1999) Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Computing Surveys 31: 4406-4471.
3. Ahmad I, Kwok YK (1999) Benchmarking and comparison of the task graph scheduling algorithms. J Parallel Distributed Computing 95: 381-422.
4. Billionnet A, Costa MC, Sutter A (1992) An efficient algorithm for the task allocation problem. J ACM 39 : 502-518.
5. Braun TD, Siegel H J, Beck N, Boloni L, Maheswaran M, et al. (2001) A comparison of eleven static heuristic for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing 61: 810-837.
6. Rahmani AM, Resvani M (2007) A novel static task scheduling in distributed systems by genetic algorithm using simulated annealing. 12th International CSI Conference, Iran 83.
7. Silva ML, Porto SCS (1999) An Object-Oriented Approach to a Parallel Tabu Search Algorithm for the Task Scheduling Problem. Proceedings of the 19th International Conference of the Chilean Computer Science Society 105-111.
8. Ahmad SG, Munir EU, Nisar W (2012) PEGA: A performance effective genetic algorithm for task scheduling in heterogeneous systems. IEEE 14th International Conference on High Performance Computing and Communications 1082-1087.
9. Agarwal A, Colak S (2014) The Task Scheduling Problem: A NeuroGenetic Approach. Journal of Business and Economics Research - Fourth Quarter 12.
10. Parvan H, Nejad EB, Alavi SE (2014) New hybrid algorithms for task scheduling in computational grids to decrease makespan. International journal of Computer Science and Network Solutions 2.
11. Mehrabi A, Mehrabi S, Mehrabi AD (2009) An adaptive genetic algorithm for multiprocessor task assignment problem with limited memory. Proceedings of the World Congress on Engineering and Computer Science 2.
12. Devi MR, Anju A (2014) Multiprocessor scheduling of dependent tasks to minimize makespan and reliability cost using NSGA-II. International Journal of Computer Science and Telecommunications 4.
13. Diana S, Ganapathy L, Pundir AK (2013) An improved genetic algorithm for resource constrained project scheduling problem. International Journal of Computer Applications 0975-8887 78.
14. Awadall M, Ahmad A, Al-Busaidi S (2013) Min–min GA Based Task Scheduling In Multiprocessor Systems International Journal of Engineering and Advanced Technology (IJEAT) ISSN 3: 2249-8958.
15. Dhingra S, Gupta SB, Biswas R (2014) Genetic algorithm parameters optimization for bi-criteria multiprocessor task scheduling using design of experiments. World Academy of Science, Engineering and Technology International Journal of Computer, Information, Systems and Control Engineering 8.
16. Sivanandam SN, Deepa SN (2008) Introduction to genetic algorithms. Springer Publishing Company, Incorporated.
17. Mitchell M (1998) An Introduction to Genetic algorithms. The MIT Press.

18. Bonyadi MR, Moghaddam ME (2009) A bipartite genetic algorithm for multi-processor task scheduling. International Journal of Parallel Programming 37: 462-487.

19. Chitra P, Venkatesh P, Rajaram R (2011) Comparison of evolutionary computation algorithms for solving bi-objective task scheduling problem on heterogeneous distributed computing systems. Sadhana 36: 167-180.

20. Hartmann S (1997) A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics 45: 733-750.