Genetic Algorithm Solution for Scheduling Jobs in Multiprocessor Environment

Ravreet Kaur
Deptt of Computer Science & Engineering
Guru Nanak Dev University
Amritsar (Punjab), India
ravreet@yahoo.com

Dr. Gurvinder Singh
Deptt of Computer Science & Engineering
Guru Nanak Dev University
Amritsar (Punjab), India

***Abstract*-Multiprocessor task scheduling is considered to be the most important and very difficult issue in High Performance Computing. Task scheduling is performed to match the resource requirement of the job with the available resources resulting in effective utilization of multiprocessor systems. In this paper, a Genetic algorithm (GA) is proposed for static, non-preemptive scheduling problem in homogeneous fully connected multiprocessor systems with the objective of minimizing the job completion time. The proposed GA is used to determine suitable priorities that lead to a sub-optimal solution. To compare the performance of proposed algorithm, Static algorithms of BNP (Bounded Number of Processors) scheduling class i.e. HLFET (Highest Level with First Estimated Time) and MCP (Modified Critical Path) are implemented. HLFET, MCP and proposed GA are tested by mapping our tasks in a directed acyclic graph (DAG). Performance analysis of HLFET, MCP and proposed GA for a given job scheduling problem proves that GA results in better sub-optimal solutions.**

Keywords-multiprocessor task scheduling; Genetic Algorithm; optimal solution ; BNP scheduling class; HLFET (Highest Level with First Estimated Time); MCP (Modified Critical Path); DAG (Directed Acyclic Graph)

I. INTRODUCTION

The field of parallel task scheduling is one of the most advanced and rapidly evolving fields in computer sciences. High Performance Computing is expected to bring a break-through in the increase of computing speed. This calls for appropriate scheduling strategies controlling access to such resources as well as scheduling strategies controlling execution of the parallel application modules. The scheduling problem deals with the optimal assignments of a number of jobs onto parallel system and orders their execution to achieve certain objective function.

Optimal solution of scheduling is not feasible. A large number of algorithms have been proposed which attempt to bring a sub-optimal solution. Multiprocessor scheduling problem involves mapping a Directed Acyclic Graph (DAG) for a collection of computational tasks and their data precedence onto a parallel processing system [8]. In our case we consider only static scheduling problem. The objective of DAG scheduling is to minimize the overall program finish time by proper allocation of jobs to the processors and arrangement of execution sequencing of tasks.

In this paper, HLFET and MCP, BNP scheduling algorithms performance have been compared with proposed GA on basis of a performance metric; MakeSpan. In BNP scheduling algorithms, the processors are assumed to be fully connected and no attention is paid to link contention or routing strategies used for communication [1].

An example of a directed acyclic graph (DAG) consisting of 9 tasks is shown in "figure 2" and a fully connected multiprocessor systems consisting of two processors (m=2) is shown in "figure 1". The multiprocessor scheduling is to assign the set of tasks T onto the set of processors P in such a way that precedence constraints are maintained, and to determine the start and finish times of each task with the objective to minimize the completion time.

A. *Assumed Parallel Computation Model*

- A Multiprocessor System with m machines
- A task represented by a DAG
- The estimated execution duration of every subtask and the estimated data transmission duration between adjacent subtasks

In our study we choose Static BNP scheduling algorithms, HLFET and MCP for multiprocessor scheduling.

HLFET algorithm steps are as follows:

1. Calculate the Static Level of all the nodes in the DAG.
2. Insert all the nodes into a list according to descending order of Static Level of the nodes.
3. While not the end of the list do
   a. Remove node n_i from the list.
   b. Compute the earliest start execution time of n_i for all the processor present in the system.
   c. Map the node n_i to the processor that has the least earliest start execution time.

MCP algorithm steps are as follows:

1. Calculate the Latest Start Time (LST) of all the nodes in the DAG.
2. Insert all the nodes into a list and sort the list according to ascending order of Latest Start Time.
3. While not the end of the list do
   a. Remove the node from the list.
   b. Compute the earliest start execution time of n_i for all the processors present in the system.
   c. Map the node n_i to the processor that has the least earliest start execution time.

A major factor in the efficient utilization of multiprocessor systems is the proper utilization of multiprocessor system by proper assignment and scheduling of computational tasks of a DAG among available processors. The multiprocessor scheduling problem is known to be NP-complete.

Hence, satisfactory sub-optimal solutions obtainable in a reasonable amount of computation time are generally sought [4]. GA helps in achieving this objective of sub-optimal solutions having greater efficiency than already existing scheduling algorithm. This paper proves the above stated result.

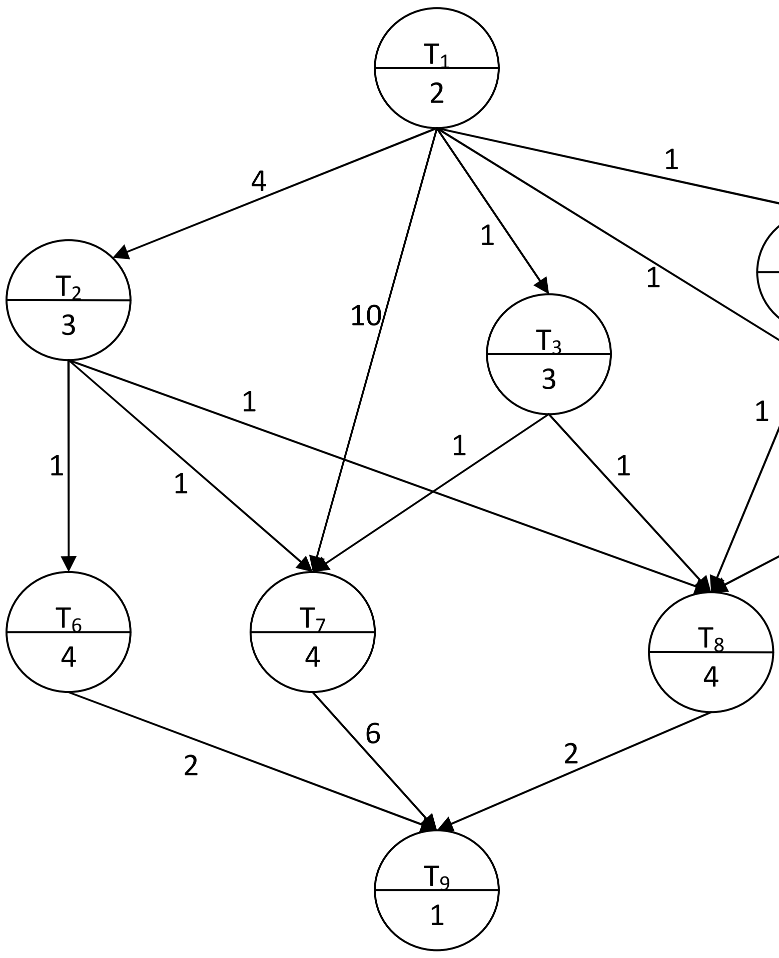


Figure 2: A Directed Acyclic Graph

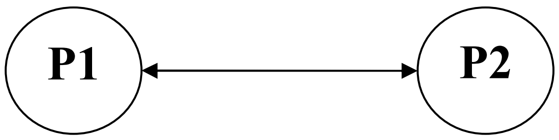


Figure 1: A Fully Connected Multiprocessor System

II. LITERATURE SURVEY

Kwok, Y.-K. and Ahmad, I.[1] have discussed the taxonomy of various scheduling algorithms in different categories. Further a set of benchmarks have been produced to compare different scheduling algorithms, so as to interpret which algorithms perform better than the others. **Hou, E.S.H., Ansari, N., and Hong Ren,[2]** developed an efficient method based on genetic algorithms to solve multiprocessor scheduling problems. Further proposed genetic algorithm is compared with list scheduling algorithm. **Ahmad, I.; Dhodhi, M.K[4]** have developed a problem – space genetic algorithm (PSGA) with the objective of minimizing the completion time. PSGA combines the genetic algorithms and list scheduling heuristic in order to reduce computation time and to increase the resource utilization. **Reakook Hwang, Mitsuo Gen and Hiroshi Katayama [7]** address the challenge of multiprocessor task scheduling parallel programs, represented as directed acyclic graph (DAG), for execution on multiprocessors with communication costs. Moreover, genetic algorithm has been proposed with new encoding mechanism – priority based multi-chromosome (PMC). The proposed priority – based GA has shown effective performance in various parallel environments for scheduling methods. **Fatma A. Omara and Mona M. Arafa, [9]** developed and implemented two genetic algorithms with some heuristic principles so as to improve performance. First genetic algorithm applies two fitness functions; one to minimize total execution time and the other one to provide load balance. Second genetic algorithm is based on task duplication technique to overcome communication overhead. **Gupta, S.; Agarwal, G.; Kumar, V. [10]** have developed a genetic algorithm based on the principles of evolution found in nature for finding an optimal solution. Further the proposed algorithm has been compared with another scheduling algorithm HEFT. **Dr.G.Padmavathi and Mrs.S.R.Vijayalakshmi[12]** have compared genetic algorithm with list scheduling heuristic. They have proved that genetic algorithm is best solution.

III. BASIC GENETIC ALGORITHM

Genetic algorithm (GA) works on the principle of "Survival of the fittest". GA follows the process of biological evolution to achieve fittest solution for the problem considered. This algorithm generates an initial population by randomly selecting values for genes in its chromosomes. Fittest chromosome is selected based on fitness value of all chromosomes. Fitness function is based on some metric i.e. problem specific. To design a genetic algorithm for solving any optimization problem, we have to design its components. These components are as follows:

1. A string representation of a solution.
2. A fitness function.
3. A method to create initial population.
4. Genetic operators (reproduction, crossover, mutation), and
5. Values of the control parameters i.e. population size, crossover probability, and mutation probability.

IV. PROPOSED PARALLEL GENETIC SCHEDULING ALGORITHM

A. *Problem Definition*

The multiprocessor scheduling problem is to assign a number of jobs onto the set of available processors in such a way that precedence constraints are maintained with the objective to minimize the completion time.

Assumptions made are:-

- Communication system is contention free and it permits the overlap of communication with computation.
- Job execution is started only after all the data have been received from its predecessor's nodes.
- The communication links are full duplex.
- Duplication of same task is not allowed.
- Communication is zero when two tasks are assigned to the same processor; otherwise communication cost is equal to the edge weight.

B. *Outline of the Proposed Scheduling Scheme*

The proposed GA is started with an initial population of feasible solutions. The, by applying genetic operators, sub-optimal solution is obtained after a certain number of generations. Fitness function determines the selection of best solution. The proposed GA works on a chromosome structure; which is divided into two arrays; having size equal to number of tasks in the job considered for scheduling. First array includes the tasks in the order they are being scheduled to run on a given number of processors; second array stores the processor number on which the corresponding element of first array is scheduled to run. Chromosome structure representation example is shown in "figure 3". This example is built on basis of the scheduling algorithm HLFET; where tasks T_1, T_5, T_2, T_6, T_7, T_9 will be scheduled on processor P_1 and T_4, T_3, T_8 will be scheduled on processor P_2.

Initial Population

Initial population is constructed randomly. First array elements are chosen randomly from the range 1 to total number of tasks. Second array elements are chosen randomly from the range 1 to number of processors. This operation is repeated until all the tasks have been assigned to available processors.

Fitness Function

The objective of proposing GA for job scheduling problem is to obtain a schedule with minimum job completion time. Fitness function is generated by taking into account the objective of minimizing MakeSpan is as follows:

Fitness Function = 1/schedule_length

where schedule_length is the maximum finish time of a task in a particular chromosome. The fitness function will give maximum value for chromosome with lowest completion time.

Genetic operators

In order to apply crossover and mutation operators, the selection phase should be applied first. This selection phase is used to allocate reproductive trials to chromosomes according to their fitness [9]. In this paper, roulette wheel selection procedure is used, where each chromosome in population has a slot sized in proportion to its fitness value. Each time an offspring is required; roulette wheel is spun to obtain parent chromosomes.

Crossover

Each chromosome in population is subject to crossover with some probability. Crossover operator randomly selects two parent chromosomes by using roulette wheel selection procedure. In this paper, two crossover operators are used which are chosen randomly by proposed GA.

- Single-Point Crossover: This operator is applied to the 2nd array of chromosomes. A point is selected randomly between 1 to number of tasks. The portions of chromosomes lying to the right of crossover point are exchanged to produce two offsprings (shown in "figure 4").
- Proposed Crossover: This operator is applied to the 1st array of chromosomes. A random point is chosen between 1 to number of tasks. First, pass the part lying to the left of crossover point in chromosome 1 to the offspring; and then construct the right part of offspring by including the elements of chromosome 2 not present in left part of chromosome (shown in "figure 5").

Mutation

Mutation is used to change the value of genes in a chromosome. Mutation operator replaces the value of gene with another value from the domain defined for that gene. In our proposed GA, mutation operator changes the assignment of tasks from one processor to another. "Figure 6" illustrates the mutation operator on chromosome1. After mutation operator is applied, the assignment of task T_2 is changed from processor P_1 to P_2.

Pseudo code of Proposed GA

```
Step 1 Read directed acyclic graph and build a design database. Also read population size (p), cross rate ($X_r$), mutation rate ($M_r$), number of generations ($N_g$), no. of processors (m)
Step 2 Current_pop: = Gen_initial_pop($N_p$); //Randomly generates initial population
Step 3 For I = 1 to $N_g$ DO
    • Apply the decoding heuristic to generate solution for each chromosome
    • Calculate the fitness for each chromosome solution in Current_pop. Save the fittest current solution in the database.
    • Select a pair of chromosomes from the current population probabilistically based on their fitness as parents to contribute to the next generation.
    • Apply crossover and mutation to generate offspring to form a new population.
    • Replace the entire current population (Current_pop) with the new population.
Step 4 Report the best final solution by choosing the chromosome with highest fitness function value. [4]
```

Decoding Heuristic:

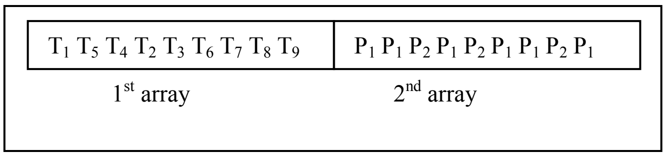
The scheduling problem can be thought of as consisting of two parts:

- Assignment of tasks to processors
- Jobs execution ordering within a processor

Heuristic is applied to generate a schedule from a given chromosome with the objective of minimizing the completion time.

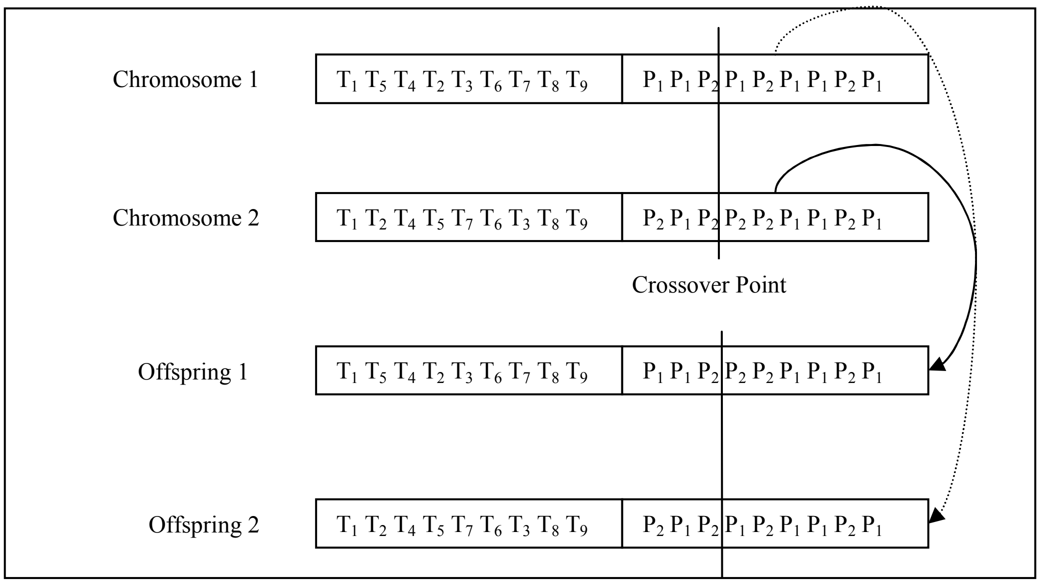
1. Build a task list from a chromosome and initialize a ready list with all the tasks.
2. Task from the ready list is selected and scheduled to the available processor on which the start time

of the task is the earliest taking into account the communication of data among processors.
3. Task is deleted from the ready list.
4. This process is repeated on the ready list till either no idle processor is available or there is no task in the ready list.



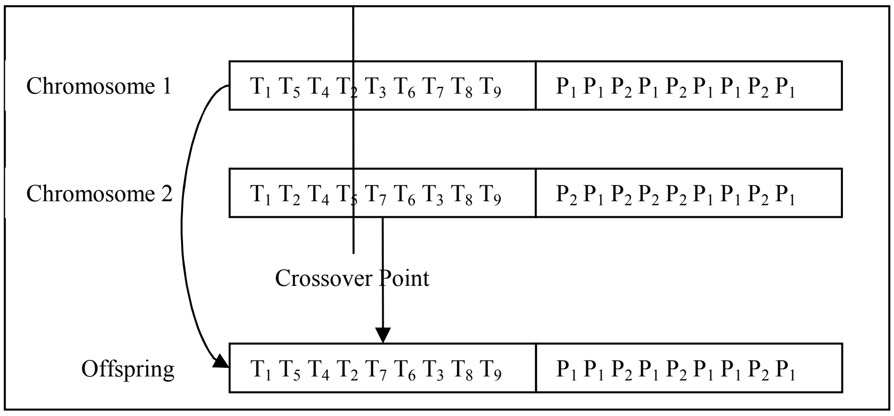
T_1 T_5 T_4 T_2 T_3 T_6 T_7 T_8 T_9
P_1 P_1 P_2 P_1 P_2 P_1 P_1 P_2 P_1
1st array
2nd array

Figure 3: Chromosome Structure Representation



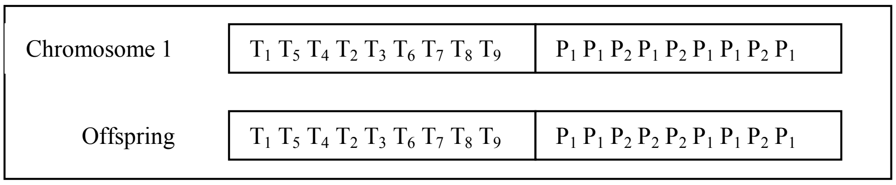
Chromosome 1
T_1 T_5 T_4 T_2 T_3 T_6 T_7 T_8 T_9
P_1 P_1 P_2 P_1 P_2 P_1 P_1 P_2 P_1
Chromosome 2
T_1 T_2 T_4 T_5 T_7 T_6 T_3 T_8 T_9
P_2 P_1 P_2 P_2 P_2 P_1 P_1 P_2 P_1
Crossover Point
Offspring 1
T_1 T_5 T_4 T_2 T_3 T_6 T_7 T_8 T_9
P_1 P_1 P_2 P_2 P_2 P_1 P_1 P_2 P_1
Offspring 2
T_1 T_2 T_4 T_5 T_7 T_6 T_3 T_8 T_9
P_2 P_1 P_2 P_1 P_2 P_1 P_1 P_2 P_1

Figure 4: Single Point Crossover



Chromosome 1
T_1 T_5 T_4 T_2 T_3 T_6 T_7 T_8 T_9
P_1 P_1 P_2 P_1 P_2 P_1 P_1 P_2 P_1
Chromosome 2
T_1 T_2 T_4 T_5 T_7 T_6 T_3 T_8 T_9
P_2 P_1 P_2 P_2 P_2 P_1 P_1 P_2 P_1
Crossover Point
Offspring
T_1 T_5 T_4 T_2 T_7 T_6 T_3 T_8 T_9
P_1 P_1 P_2 P_1 P_2 P_1 P_1 P_2 P_1

Figure 5: Proposed Crossover



Chromosome 1
T_1 T_5 T_4 T_2 T_3 T_6 T_7 T_8 T_9
P_1 P_1 P_2 P_1 P_2 P_1 P_1 P_2 P_1
Offspring
T_1 T_5 T_4 T_2 T_3 T_6 T_7 T_8 T_9
P_1 P_1 P_2 P_2 P_2 P_1 P_1 P_2 P_1

Figure 6: Mutation Operator

V. PERFORMANCE EVALUATION AND COMPARISON

The final best schedule obtained by applying the proposed GA to the DAG of figure 1 onto the parallel multiprocessor system in figure 2, is shown in figure 11. The completion time obtained by this method is 21.0 time units. Also the comparison of results with HLFET and MCP scheduling method on parallel systems is shown in figure 10 and execution of the schedule is shown in figure 9.

HLFET Scheduling Policy assigns the jobs to the processors P1 and P2 as:
P_1: T_1, T_5, T_2, T_6, T_7, T_9
P_2: T_4, T_3, T_8
The total finish time of the HLFET scheduler is 27.0 time units as shown in “figure 7”.

MCP Scheduling Policy assigns the jobs to the processors P1 and P2 as:
P_1: T_1, T_2, T_5, T_7, T_8, T_9
P_2: T_3, T_4, T_6

The total finish time of the MCP scheduler is 24.0 time units as shown in "figure 8".

After applying the proposed GA, the best schedule we found is:

P_1: T_1, T_2, T_4, T_7, T_8
P_2: T_3, T_5, T_6, T_9
The total finish time of proposed GA scheduler is 21.0 time units as shown in "figure 9".

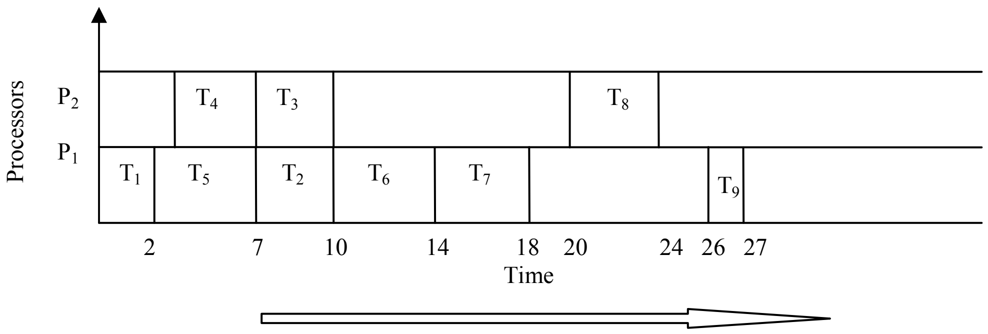


Figure 7: Gantt Chart of HLFET Scheduler

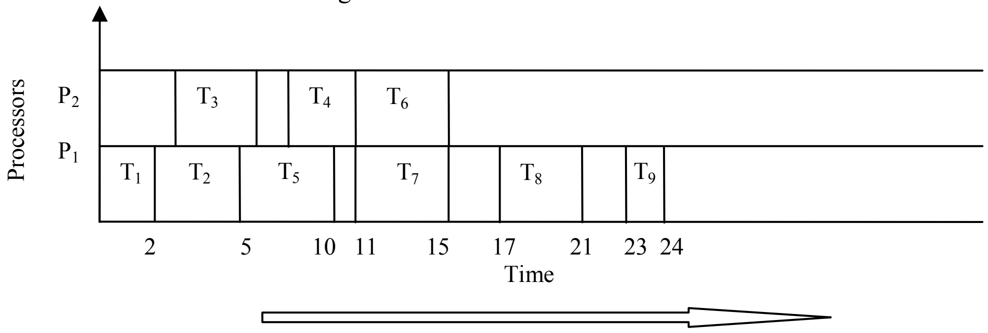


Figure 8: Gantt Chart of MCP Scheduler

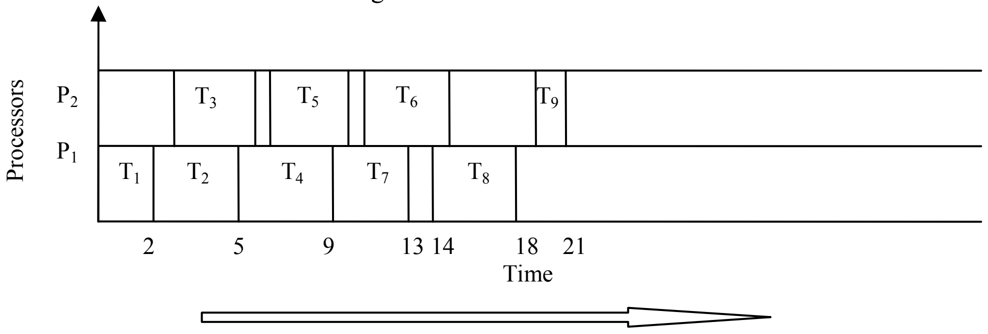


Figure 9: Gantt Chart of Proposed GA

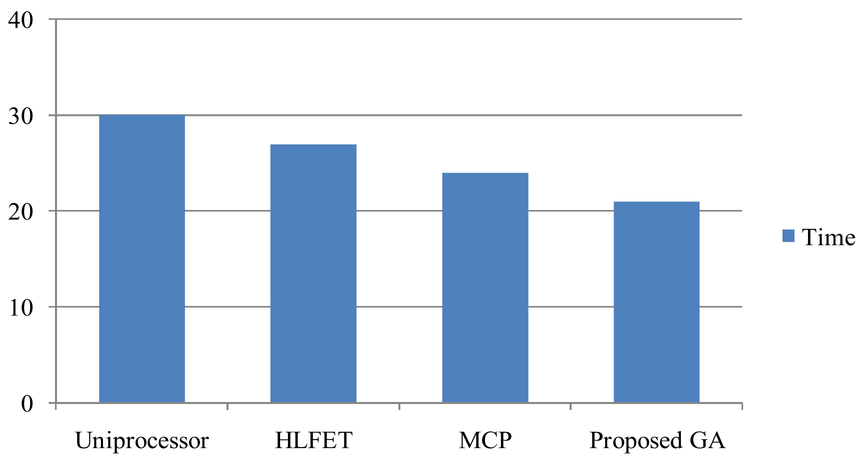


Figure 10: Time Analysis of Uniprocessor, HLFET, MCP and Proposed GA

A. *Performance Analysis*

Efficiency (E) = (SpeedUp *100)/m

Proposed Advanced GA
SpeedUp = 30/21 = 1.429
= 71.429%

HLFET Scheduler
SpeedUp = 30/27= 1.112
= 55.556%

MCP Scheduler
SpeedUp = 30/24= 1.25
= 62.5%

Therefore, performance analysis of the proposed GA, HLFET and MCP scheduling method in "figure 11" shows that Proposed GA is more efficient than HLFET Scheduler and MCP Scheduler i.e. Static Scheduling algorithms of BNP class .

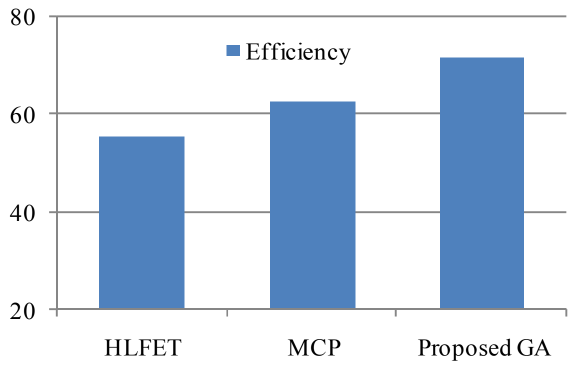


Figure 11: Performance Analysis of HLFET, MCP and Proposed GA

VI. CONCLUSION

In this paper, a new GA has been proposed for Job scheduling in parallel multiprocessor system including the communication delays to reduce the completion time and to increase the throughput of the system. The proposed GA is different from the Basic GA which requires a uniform order based crossover operator and a special mutation operator, while proposed GA can use any crossover and mutation operator. The proposed GA uses a different chromosome structure to search a large solution space in an intelligent way in order to find the best possible solution within an acceptable CPU time. The proposed method found a sub-optimal solution for assigning the tasks to the homogeneous parallel multiprocessor system. Performance analysis of the proposed GA, HLFET and MCP scheduling algorithm shows that proposed GA gives better result for job scheduling in multiprocessor environment.

REFERENCES

[1] Kwok, Y.-K. and Ahmad, I., "Benchmarking the Task Grapg Scheduling Algorithms" in Parallel Processing Symposium, 1998. IPPS/SPDP 1998. Proceedings of the First Merged International ... and Symposium on Parallel and Distributed Processing 1998, pp. 531 – 537

[2] Hou, E.S.H., Ansari, N., and Hong Ren, "A genetic algorithm for multiprocessor scheduling" in Parallel and Distributed Systems, IEEE Transactions, Volume: 5 Issue:2, pp. 113 – 120, 1994

[3] Ahmad, I. Yu-Kwong Kwok, "A New Approach to Scheduling Parallel Programs Using Task Duplication", in Parallel Processing, ICPP 1994, International Conference, 1994, pp. 47-51

[4] Ahmad, I.; Dhodhi, M.K.; "Multiprocessor scheduling using a problem-space genetic algorithms", in Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995, GALESIA, First International Conference on (Conf. Publ. No. 414), pp. 152 – 157

[5] Wu Ling, Li Bin, "Job-shop scheduling using Genetic Algorithm" in Systems, Man, and Cybernetics, 1996, IEEE International Conference, pp. 1994 – 1999 vol. 3

[6] Shiyuan Jin, Guy Schiavone and Damla Turgut, "A performance study of multiprocessor task scheduling algorithms", in The Journal of Supercomputing, Volume 43, Number 1, pp. 77-97, 2007

[7] Reakook Hwang, Mitsuo Gen and Hiroshi Katayama, "A comparison of multiprocessor task scheduling algorithms with communication costs", in Journal Computers and Operations Research, Volume 35 Issue 3, March, 2008

[8] David E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Published by Pearson Education, 2004, Page No. 60-83

[9] Fatma A. Omara and Mona M. Arafa, "Genetic algorithms for task scheduling problem", in Journal of Parallel and Distributed Computing, Volume 70, Issue 1, January 2010, pp. 13–22

[10] Gupta, S.; Agarwal, G.; Kumar, V., "Task Scheduling in Multiprocessor System Using Genetic Algorithm", in Machine Learning and Computing (ICMLC), 2010 Second International Conference, feb 2010, pp. 267 - 271

[11] Dr.G.Padmavathi and Mrs.S.R.Vijayalakshmi, "Multiprocessor Scheduling for Tasks with Priority using GA", in International Journal of Computer Science and Information Security, IJCSIS, Vol. 6, No. 3, pp. 093-100, December 2009

[12] Dr.G.Padmavathi and Mrs.S.R.Vijayalakshmi, "A Performance Study of GA and LSH in Multiprocessor Job Scheduling", in International Journal of Computer Science Issues, IJCSI, Vol. 7, Issue 1, No. 1, January 2010