TASK SCHEDULING WITH GENETIC APPROACH AND TASK DUPLICATION TECHNIQUE

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

ABSTRACT
The basic and prime job in parallel multiprocessor environment is accurately mapping of tasks and set the scheduling of different tasks on different processors to minimize the makespan. Total runtime is time taken time for all jobs with the individual runtime of tasks and their communication cost among tasks. In parallel multiprocessor environment, an optimal scheduling of parallel tasks having precedence relationship on such system is same as NP-complete problem. In this paper, the scheduling hindrance is bringing out the optimal mapping of tasks and their efficiently possible execution stream on multiprocessor system configuration. Numbers of alternative solutions and heuristics techniques are proposed to solve such type of problem. By applying the GA with task duplication technique enhances the efficiency of the task scheduling in parallel multiprocessor environment. Here also by comparing this new approach with some existing deterministic scheduling techniques for minimizing inter processor traffic communication will speed up the scheduling.
.

Keywords:
Task scheduling, Parallel Computing, Genetic Algorithm (GA), Task duplication Genetic algorithm (TGA).

1. INTRODUCTION
The task scheduling has utmost significance in homogeneous and heterogeneous multiprocessor parallel computing environment. The problem of task scheduling on such type of systems is very crucial and it verify the sequence of processor, when and on which processor a given task executes. Our major goal is to minimize the mean task response time (total finish time or makespan) and meet the deadline at the same time. At last our final intention in this scheduling is to get the final time and cost of the execution and that would be minimized and also all other fundamental constraints are satisfied [1][2][3]. Here in the problem currently originated, the process of scheduling is static because the finite number of processor, mapped tasks according to the processor sequence, their dependencies on tasks and processor, and inter processor communication cost used for these systems are known in advanced before going to generate a task schedule[4][5]. Here it is important to determine the set of finite tasks T and finite number of processors NOP so that the mapping of every task to each processor from a set of processors P. In this defined or created task set every task may communicate with zero or more other tasks in set with some communication cost value having positive to negative set of values. So in experiment the main focus is on four variables during the execution of scheduling. These are

- Performance factor of homogeneous and non homogenous processors,
- Mapping of different scheduling tasks on working processors,
- Sequence and ordering of processors and
- At end variable set the sequence of execution of tasks under various scheduling criteria like FCFS-first come first serve, PS-priority scheduling, LS-list scheduling, GA-genetic algorithm and TA-Task duplicated concept with genetic approach on different processors.

Here a powerful tool DAG (Directed Acyclic Graph) can be taken which represents the execution and communication costs in the system [1][5]. In this system the task scheduling problem is taken as NP-complete problem or NP-hard problem and there are number of heuristic search techniques considered and only genetic algorithms have attracted much attention as class of arbitrarily search algorithm for task scheduling in homogeneous and heterogeneous processor environment [5].

This paper is organized in six sections and rest five are as follows: Section 2. gives an overview of the problem along with brief description of the solution methodology. In the section 3 detailed proposed genetic algorithm explained. Section 4 describe the concept of task duplication. Section 5 provides the experimental results and performance analysis of the study. At end in the section 6 conclusion was done.

2. OVERVIEW
A brief review of parallel classification algorithms and methods based on the characteristics of the tasks to be scheduled, the parallel environment and the availability of the information is presented in this section[6][9][11][12][13][14]. Such type of big task is efficiently partitioned into set of small tasks having appropriate grain size with or without fine grain based on the technique behind the execution of the tasks on parallel computing environment and create an abstract model of the partitioned tasks that can be represented in the form of a set graph known as DAG (Directed Acyclic Graph) [20]. The major objective is to analyize the deterministic scheduling problem in which there exist a set of precedence relations among the tasks to be scheduled in a sequence. 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 either homogeneous or non homogeneous[18][19][20]. The homogeneity of processor generate a set of same type of processor with same processing speed and heterogenity of processors means that the processors have different speeds or processing capabilities. So here, a study has been done regarding the task scheduling problem as a

deterministic on the homogeneous and heterogeneous parallel multiprocessor environment. The main goal is to minimize the makespan i.e. total task finish time. Here makespan means it is execution time plus the waiting time or idle time. Let us suppose a parallel multiprocessor computing environment consists of a set of m either homogeneous or hetrogeneous processor in a sequence and can be defined in the equation as:

$$P = \{p_{i:}\ i = 1, 2, 3 \ldots m\}$$

These all are fully connected with each other via unique and identical links showing three parallel system in the Figure 1 below. DAG has the parallel application and can be represented as by the notation DAG = (v,e,w,c), where the vertices set v consist of n tasks and are represented as:

$$v = \{v_{j:}\ j = 1, 2, 3 \ldots n\}$$

A directed edge set e consist of k edges in directional form and all are denoted in the equation form as below:

$$e = \{e_{k:}\ k = 1, 2, 3 \ldots r\}$$

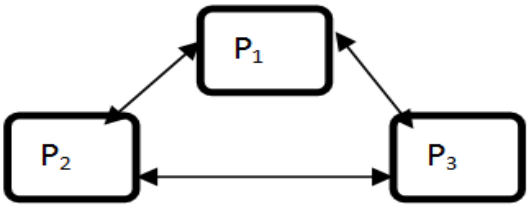


Figure 1. A fully connected having three parallel processor P_1,P_2,P_3

Precedence relationships among tasks is represented in the above figure according to the above said equation. For any two tasks t_i, t_{i+1} belongs to task set T with having directed edge e_k (edge from task t_i to t_{i+1}) means that task t_{i+1} cannot be scheduled until t_i has been completed, t_i is a predecessor of t_{i+1} and t_{i+1} is a successor of t_i. So it is common that t_i sends a message whose contents are required by t_{i+1} to start execution. The weight of the vertices has a set w weight elements and can be represented in the equation as below:

$$w = \{w_{i,j:}\ i = 1, 2, 3 \ldots m,\ j: 1, 2, 3, \ldots n\}$$

It represents the execution duration of the corresponding task and are varies from processor to processor because of either homogenous or heterogeneous processor environment. The elements set c is the communication cost according to the weights of the edges and can be represented in the equation as below:

$$c = \{c_{k:}\ k = 1, 2, 3 \ldots r\}$$

Communication of data between the two tasks can be rpresented by using this strategy 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 zero or null. There is an example of a complete DAG as shown in the Figure 2 and it consist of a set of tasks as T: $\{t_{j:}\ j = 1, 2 \ldots t\}$. Similarly Figure 1 indicates set of processors $P = \{p_i : i = 1, 2, 3\}$ for three processors and Table 1 show a matrix of execution time of each task on different processor in case of of heterogeneous environment every processor works on different speeds and processing capabilities. It is assumed that processor p_1 is much faster than p_2, p_3 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 $p_1 > p_2 > p_3 > p_4 > p_5 \ldots\ldots$). As given in Table 1 task t_1 takes 4 time units to complete their execution on processor p_1 and takes 9 time units and 10 time units to complete their execution on processor p_9 and p_{10} respectively. On the basis of the size of the tasks processed on different processors, the execution time has been calculated[10][17].

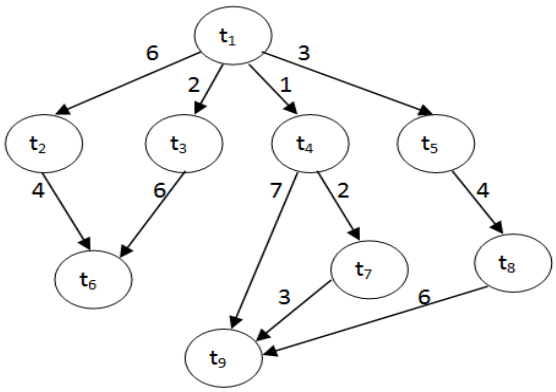


Figure 2: A DAG with task and edges

Table 1: Task execution matrix on three processors

W_{ij}	t1	t2	t3	t4	t5	t6	t7	t8	t9
P1	6	7	7	6	5	9	10	8	7
P2	7	9	9	7	6	11	13	10	9
P3	8	11	10	8	7	12	14	11	10

3. PROPOSED GENETIC ALGORITHM

A genetic algorithm is evolutionary heuristics technique starts with an initial population that evolves through generations and to reproduce generations depends on its fitness [3, 4]. Here the fitness of an individual is defined as the difference between its final execution time and the one of the individuals having the maximum final time in the population. The best individual corresponds to the one having the minimum makespan and the maximum fitness. In the next step, the three genetic operators that compose a genetic algorithm are studied. The selection operator allows the algorithm to generate a set of good individuals after changing the generations. By doing so a set of good individuals are replicated from the set of given generation, while some of the bad individuals are deleted. After this a result of good individual population will be generated. By considering the three processor and starting from a population of processor P_1, this transformation is implemented iteratively by generating a new population of processor P_2 of the same size as in the processor P_1. Genetic algorithms are based on the principles that crossing two individuals can result an offsprings that are better than both parents and slight mutation of an individual can also generate a better individual. The crossover takes two individuals of a population as input and generates two new individuals, by crossing the parents' characteristics. The offsprings keep some of the characteristics of the parents. The mutation randomly

transforms an individual that was also randomly chosen. It is important to notice that the size of the different populations is same. The structure of the algorithm is a loop composed of a selection followed by a sequence of crossovers and a sequence of mutations. After the crossovers, each individual of the new population is mutated with some (low) probability. This probability is fixed at the beginning of the execution and remains constant. The termination condition may be the number of iterations, execution time, results stability, etc **[3][7][8]**.

3.1 Initial population

Basic part of the GA is 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 task 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 two levels that is b-level (bottom level) or t-level (top level) precedence resolution to avoid the problem of same execution time or completion time[1][5].

3.2 Fitness operator function

Fitness operator has the major goal in the task scheduling in multiprocessor system and it compute the shortest possible schedule. In other words, the fitness of chromosome is directly related to length of associated schedule and its sequence. In the task scheduling environment, various factors like throughput, turnaround time, and processor utilization etc. can be considered. The fitness function or fitness operator used for genetic algorithm for computing the minimum task span or the shortest possible schedule. For this focus on total finish time for the required schedule that include execution time and communication delay time. Fitness function computation depends upon the two factors. In the first factor is task fitness and fitness of task which equipped us with the knowledge of all the tasks which are executed and scheduled in the legal order. The legal order is the scheduling of a pair of task on single processor if the pair is independent to each other. The second factor is processor fitness i.e. fitness of processor attempt to minimize processing time. With the help of these two factors of fitness operator possible shorter schedule can be drawn. In the table 2, a legal order is designed and according to the legal order the processor P1 start execution from task t1, whereas in case of illegal order nor a single processor start their execution from different tasks t4, t3, t5 until task t1 executed. So initiation of illegal task needs some information from task t1[18][19].

Table 2: arbitrarily assigned tasks to processor

Processors	Order of Tasks (legal)	Order of Tasks (illegal)
P1	t1, t4, t6, t9	t4, t6, t1, t9
P2	t2, t3, t7	t3, t7, t2
P3	t5, t8	t5, t8

S1 and S2 are the two different schedule on the different set of processor i.e. for uniprocessor and multiprocessor systems. In schedule S1 all tasks execute on the processor P (uniprocessor system) and in S2 schedule tasks are randomly distributed as according to the legal order on all the processors as execution time shown in table 1[19].

Schedule S1 (On Uni-processor):

P: t1→t2→t3→t4→t5→t6→t7→t8→t9

Total finish time = 65, Communication cost = 0

Schedule S2 (On three processor):

P1: t1→t4→t6→t9

P2: t2→t3→t7

P3: t5→t8

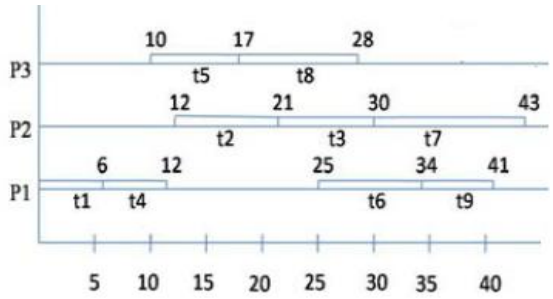


Figure 3: Execution Schedule of S2

Execution concept can be shown in the Gantt chart having two axis, the vertical axes shows the processor side and horizontal axis shows the time side as shown in the figure 3. As shown in the figure 3, the waiting and idle time of processors be shown by the single line and bar with numeric data (above) shows the starting and finishing time of task and name of task (below).

Total finish time = Execution time + communication Cost
= 43 time unit.

From the above two schedule executions, the schedule scheme in the uni-processor schedule S1 shows the total finish time only 65 units time whereas in the case of second schedule S2 with the help of multiprocessor represents the only 43 units time, which is less comparative to the previous one i.e. the proper calculation of fitness operator function reduces the total finish time [20].

3.3 Selection operator

First step in the GA is the implication of selection opertor with the help of the fitness function. 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[2][3][19][20].

3.4 Crossover operator

Crossover operator is second one operator used by the GA for finding the more good results. It randomly selects two parent chromosomes and randomly chooses their crossover points, and mates them to produce two child new chromosomes which is also called new offspring. In other words chromosomes with

higher values have more chance to be selected. Here one point crossover and two point crossover operators can be examined to get the best series of cromosome. First one in the case of one point crossover, the segments to the right of the crossover points are exchanged to form two offspring as shown in Figure 4(a). Second one in the case of two point crossover [19][20], the middle portions of the crossover points are exchanged to form two offspring as shown in Figure 4(b).

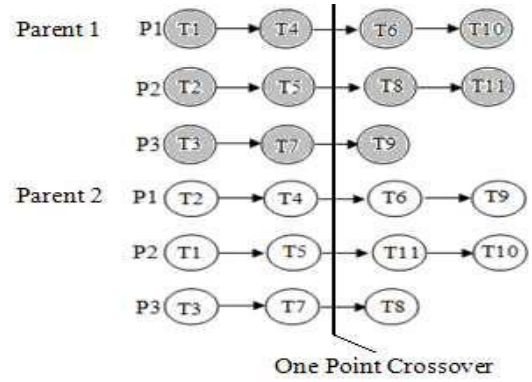


Figure 4(a): One Point Crossover

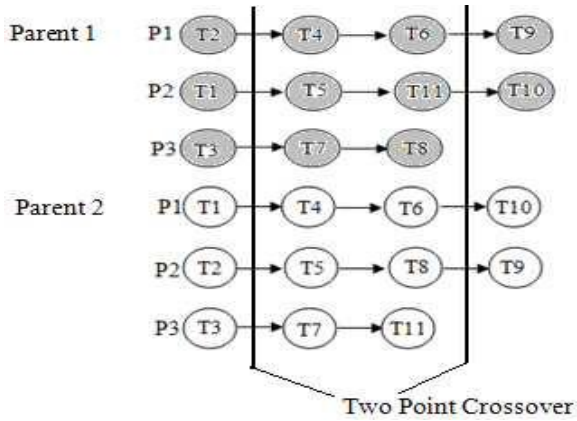


Figure 4(b): Two Point Crossover

3.5 Mutation operator

The probability of finding the optimal solution is never zero can be ensured with the help of a new concept known as mutation. It also acts as a safety net to recover good genetic material that may be lost through selection and crossover. By using the concept of partial-gene mutation randomly which selects a chromosome and changes a randomly selected gene (Ti , Pi) to (Ti , Pj) for which avail time (Pj) is minimum over all the processors for task Ti [19]. It is to notify that partial-gene mutation changes the processor which a task is assigned to, whereas crossover assigns a set of tasks to, probably, different processors. In this light, partial-gene mutation was applied in conjunction with crossover operator[14][19]. With the help of mutation operator idle time can be reduced. The various steps of implemetation of the Genetic Algorithm (GA) in a sequence 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. 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 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 f_c. Also compute the fitness function or fitness value of each node or task. Let it be f_n 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. 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 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.

All the above eight steps generate a valid chromosome generation and gives the best fitness.

4. TASK DUPLICATION

Interprocessor communication is a major hindrance in the task scheduling and only task duplication technique can avoid it. To avoid such type of problem, a node often has to wait for entering communication. If analyze the DAG as shown in figure 2, note that task t2 and task t3 are put in the waiting form due to communication from task t1. During this period both processor p2 and p3 run idle and create a wastage of time. With the help of task duplication, a communication criteria can be set up. Here a communication from task t1 is set local to on each processor as shown in figure 6 of the target system and tasks t3,task t2 and task t5 can start immediately after task t1 finishes. The concept of task duplication was fully explained by the renowned computer scientist in handling the scheduling approaches by Kruatrachue and Lewis, and Ahmed and Kwok. With the help of Genetic algorithm and by using the concept of task duplication, the scheduling criteria can be finalized easily and can get the minimum makespan. Some tasks are unnecessarily duplicated during the scheduling procedure with the help of task duplication[19][20].

5. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

A complete experimental Gantt chart with the help of Genetic algorithm after applying on the given DAG having different task preference as shown in figure 2 and implication of these task onto either a set of homogeneous or a set of

heterogeneous multiprocessor system as shown in the figure 1 is shown in figure 5 i.e. the execution of different tasks on different processors with the help of GA. The major objective is to reduce the total finish time is fulfilled and note that the total completion time is obtained by this technique is only 32 time units which is minimum [15][16][18]. So by doing this experiment, a comparison of results with traditional task scheduling methods like FCFS and priority scheduling and new genetic algorithm heuristic with node duplication etc. can be done. During FCFS policy tasks are assigned to three different processors are as:

P_1: $t_1 \rightarrow t_4 \rightarrow t_7$
P_2: $t_2 \rightarrow t_5 \rightarrow t_8$
P_3: $t_3 \rightarrow t_6 \rightarrow t_9$

After the analysis of the Gantt chat i.e. execution diagram of FCFS like figure 3 having schedule 2, the total completion time is only 43 time units[19][20]. Priority scheduling generates a tasks scheduling according to its priority values in scheduling sequence and so according to their priority value tasks are assigned to the different processors as shown in the schedule S3 below:

S3 (Priority scheduling on multiprocessor):
P_1: $t_1 \rightarrow t_2 \rightarrow t_6$
P_2: $t_3 \rightarrow t_5 \rightarrow t_4 \rightarrow t_8$
P_3: $t_9 \rightarrow t_7$

List scheduling (LS) is another scheduling criteria having minimum start time and generate a schedule S4 as shown below:

S4 (List Scheduling on multiprocessor system):
P_1: $t_1 \rightarrow t_2 \rightarrow t_5 \rightarrow t_6$
P_2: $t_3 \rightarrow t_7$
P_3: $t_4 \rightarrow t_9 \rightarrow t_8$

This schedule also gives the total completion time only 32 units time. After analysis of execution schedule of priority scheduling, the total execution completion time is only 43 time units. Here processors with same priority tasks are arranged in FCFS manner. After applying the genetic algorithm, then the best optimal schedule of tasks having schedule S5 is as shown below:

S5 (GA Scheduling on multiprocessor system):
P_1: $t_1 \rightarrow t_5 \rightarrow t_4 \rightarrow t_8 \rightarrow t_9$
P_2: $t_2 \rightarrow t_6$
P_3: $t_3 \rightarrow t_7$

After applying the new scheduling criteria Genetic Algorithm, the execution time can be reduced i.e by using the GA to the collection tasks, the performance becomes better than other type of scheduling algorithm. The total finish time of enhanced GAs scheduler is only 32 time units as shown in execution graph or Gantt chart as in the figure 5. This Gantt chart shows that the execution time or completion time or makespan can be minimized with the help of a new heuristic technique and Genetic algorithm occurs as the best algorithm as compared to the previous technique FCFS, LS, Priority etc. [19][20].

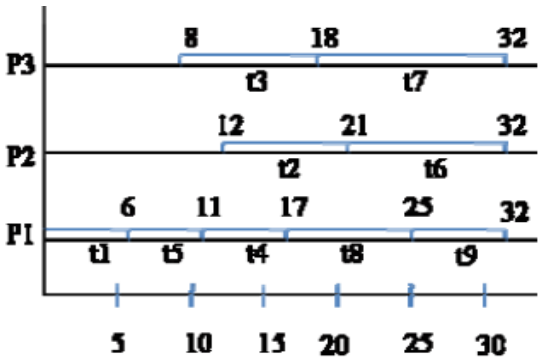


Figure 5: Execution schedule of task with GA

Here vertical and horizontal axes description is the same as explained in the Gantt chart i.e. in the figure 3. Now another new technique task duplication genetic algorithm (TGA) is applied and the total finish time or completion time or execution time or make span of TGA is 26 time units as shown in the Gantt chart i.e. in the figure 6. With the help of Genetic algorithm and task duplication technique the optimal schedules for different tasks on multiprocessor system as shown in the schedule S6 is below:

S6 (GA with task duplication i.e TGA)
P_1: $t_1 \rightarrow t_5 \rightarrow t_4 \rightarrow t_8 \rightarrow t_9$
P_2: $t_1 \rightarrow t_2 \rightarrow t_6$
P_3: $t_1 \rightarrow t_3 \rightarrow t_7$

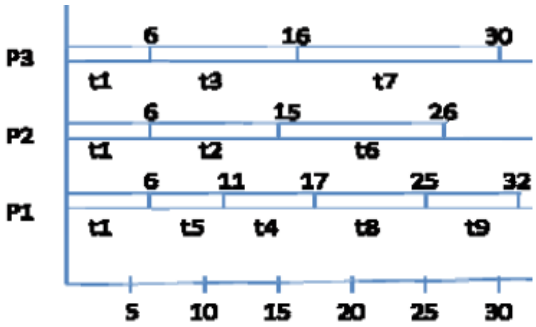


Figure 6: Execution schedule of task with Task duplication Genetic algorithm (TGA)

Performance analysis

Performance analysis is used to get the speed and efficiency of all the above scheduling algorithms[20].

Speed up (T_{sp}):

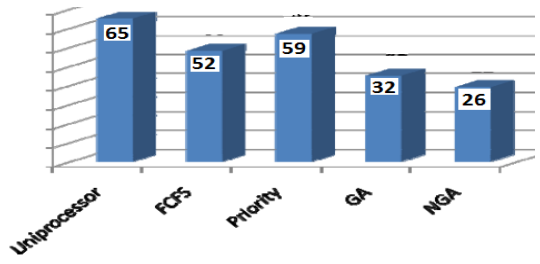
Speed up is defined as the ratio of completion time on uniprocessor system to completion time of multiprocessor system.

In case of Task duplication Genetic Algorithm (TGA):
Speed up (T_{sp}) = 65/34 = 1.9117

Efficiency (Eff.)

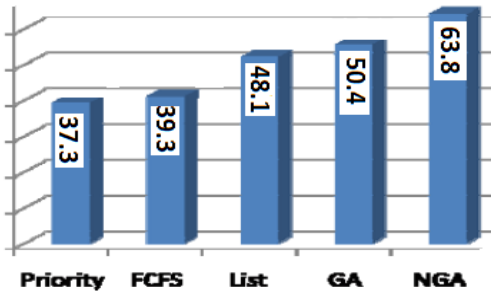
Eff.= (T_{sp} * 100) / n;
where n is the number of processors in the system.
Eff. = (1.9117 * 100) / 3 = 63.8%
In case of Genetic Algorithm (GA):
T_{sp} = 65 / 43 = 1.5116
Eff. = (1.5116 * 100) / 3 = 50.4%
In case of List Scheduling with start time minimization (LS):
T_{sp} = 65 / 45 = 1.4444
Eff. = (1.4444 * 100) / 3 = 48.1%

In case of FCFS Scheduler:
T_{sp} = 65 / 55 = 1.1818
Eff. = (1.1818 * 100) / 3 = 39.3%
In case of Priority Scheduler (PS):
T_{sp} = 65 / 58 = 1.1206
Eff. = (1.1206 * 100) / 3 = 37.3%
The bar graph representation for the time analysis is as shown below:



Time Analysis Bar Graph

The bar graph representation for the performance analysis is as shown below:



Performance Analysis Bar Graph

6. CONCLUSION

After studying and analysing this paper, it is concluded that a proposal was prepared having a GA with the task duplication (TGA) based technique for deterministic either a set of homogeneous or the set of heterogeneous multiprocessor system which comprises the communication cost in precedence to minimize the total execution time or completion time or finish time or makespan . By doing so it improve the throughput of system. After a complete comparisons it is noticed that the TGA is the best algorithm or technique as compared to the previous algorithms or techniques or criteria's. Here Task duplication introduces a new concept that is more computation load into the multiprocessor parallel system in order to decrease the cost of crucial communication. Here consideration was done that which task should be duplicated to reduce the overall time. This technique can be used for future for enhancing the jobs execution for the implementation of this proposed TGA method to solve the problem of some nondeterministic homogeneous or heterogeneous multiprocessor systems used for real life and real time execution.

7. REFERENCES

[1] Sara Baase, Allen Van Gelder, "Computer Algorithms", Published by Addison Wesley, 2000.
[2] Sartaj Sahni, "Algorithms Analysis and Design", Published by Galgotia Publications Pvt. Ltd., New Delhi, 1996.
[3] Anup Kumar, Sub Ramakrishnan, Chinar Deshpande, Larry Dunning, "IEEE Conference on Parallel processing", 1994, Page no.83-87.
[4] Ananth Grama, Georage Karypis, Anshul Gupta, Vipin Kumar, "Introduction to parallel computing", Published by Pearson Education, 2009.
[5] Elnaz Zafarani, Amir Masoud Rahmani, Mohammad Reza feizi Derakhshi, "IEEE Proceeding 2008, Page No. 248-251
[6] David E. Goldberg, "Genetics Algorithms in Search Optimization and machine Learning", Published by Pearson Education, 2004, Page No. 60-83.
[7] Mitchell, Melanie, "An Introduction to Genetic Algorithms", published Bu MIT Press 199.
[8] Sung-Ho Woo, Sung –Bong Yang, Shin-Dug Kim and Tack-don Han."IEEE Trans on Parallel System", 1997, Page 301-305
[9] M. Salmani Jelodhar, S.N Fakhraie, S.M Fakhraie, M.N Ahmadabadi, "IEEE Proceeding", 1999, Page No. 340-347.
[10] Man Lin and Laurene Tianruo Yang, "IEEE Proceeding", 1999 Page No. 382-387
[11] Yajun Li, Yuhang Yang, Maode Ma, Rongbo Zhu, "IEEE Proceeding", 2008.
[12] Michael J Qumn, "Parallel Programming", Published by Tata McGraw Hill Education Private Ltd, Page No. 63-89.
[13] John L Hennessy, David A Pattern, "Computer Architecture, 3rd Edition", Published by Morgan Kaufmann & Elsevier India, Page No. 528-590.
[14] David E Culler, "Parallel Computer Architecture", Published by Morgan Kaufmann & Elsevier India.
[15] J P Hayes, "Computer Architecture and Organization", Published by McGraw Hill International Edition.
[16] J D Carpinalli, "Computer System Organization & Architecture", Published by Pearson Education.
[17] Sung-Ho Woo, Sung-Bong Yang, Shin-Dug Kim, and tack-Don Han, "IEEE Trans on parallel System", 1997, Page 301-305.
[18] M.Salmani Jelodar, S.N.Fakhraie, S.M.fakharie, M.N. Ahmadabadi, "IEEE Proceeding", 2006, Page No 340-347.
[19] K. Kamaljit, C. Amit, Singh Gurvinder, "Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System", International Journal of Computer Science and Security (IJCSS), pp. 183 – 198.
[20] Singh J., Singh H., "Efficient Tasks scheduling for heterogeneous multiprocessor using Genetic algorithm with Node duplication", Indian Journal of Computer Science and Engineering, Vol. 2 No. 3 Jun-Jul 2011, pp. 402-410.
[21] Sharma, Manik, Gurdev Singh, and Harsimran Kaur. "A Study Of BNP Parallel Task Scheduling Algorithms Metric's For Distributed Database System." *International Journal of Distributed and Parallel Systems* 3.1 (2012): 157.
[22] Sharma, Manik, et al. "A comparative study of static object oriented metrics." *International Journal of Advancements in Technology* 3.1 (2012): 25-34.