Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Genetic Algorithm- Implementation & Approach To Sort out Scheduling Problems

Gaytri[*]
*M.Tech (C.S.E)(P)**
B.I.T.S.Bhiwani M.D.U
India.

Ramesh Yadav
A.P. (C.S.E)
B.I.T.S.Bhiwani M.D.U
India.

Priyanka
*M.Tech (C.S.E)(P)**
B.I.T.S.Bhiwani M.D.U
India.

Abstract— The scheduling problem in a multilayer multiprocessor environment finds its relevance in many industrial and computing applications. In this paper, we present a genetic algorithm for the solution. Extensive computational experiments demonstrated that genetic algorithm performs well. In literature, several heuristic methods have been developed that obtain suboptimal solutions in less than the polynomial time. Recently, Genetic algorithms have received much awareness as they are robust and guarantee for an effective solution. Genetic algorithm is wildly used to solve Flexible Task Scheduling Problem. The genetic algorithm we proposed uses many different strategies to get a better result. During the phase of create initial population, the genetic algorithm takes into account the number of operations in each job. And the intelligent mutation strategy is used which makes every individual and gene have different probability to mutate. In this paper, the object of scheduling algorithm is to get a sequence of the operations on machines to minimize the make span.

Keywords— Cp.-Crossover Probability, Mp- Mutation probability, GA- Genetic Algorithm, task scheduling, Chromosome.

1. Introduction

1.1 Scheduling of Tasks

For performing multiple task over a time we must assign them in proper sequence or phase so that they utilize resources in best suited manner i.e. Known as task scheduling.Many parallel applications consist of multiple computational components. While the execution of some of these components or tasks depends on the completion of other tasks, others can be executed at the same time, which increases parallelism of the problem. The task scheduling problem is the problem of assigning the tasks in the system in a manner that will optimize the overall performance of the application, while assuring the correctness of the result. The task scheduling problem can be modeled as a weighted directed acyclic graph (DAG). A vertex represents a task, and its weight the size of the task computation. An arc represents the communication among two tasks, and its weight represents the communication cost. The directed edge shows the dependency between two tasks.In computer science, scheduling is the method by which threads, processes or data flows are given access to system resources (e.g. Processor time, communications bandwidth). This is usually done to load balance a system effectively or achieve a target quality of service. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously).

2. Task Scheduling in Multiprocessor

Task Scheduling in Multiprocessor is a term that can be stated as finding a schedule for a general task graph to be executed on a multiprocessor system so that the schedule length can be minimized. Task scheduling in multiprocessor systems also known as multiprocessor scheduling. A major factor in the efficient utilization of multiprocessor system is the proper assignment and scheduling of computational tasks among processors.

2.1 GENETIC ALGORITHMS

Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime.

Chromosome generation

The details to generate a chromosome can be seen in following steps:
1:Select randomly a task from the entire entry tasks. Set this task as the first task in SQ.
2: Repeat step 3 for (v-1) times.
3: Randomly select a task who is not in SQ and whose predecessors all have been in SQ, and add this task to SQ.

4: For SP part, randomly generate an integer number between 1 and m for each task in SQ and add it to SP.

3.Problem Description & Objective

To solve out the multi processer scheduling problem which is defined as the assignment of a given set of tasks to a set of processors in such a way that scheduling length of task is minimum & all precedence tasks run effectively (minimum execution time).

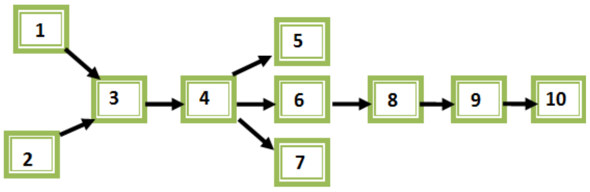
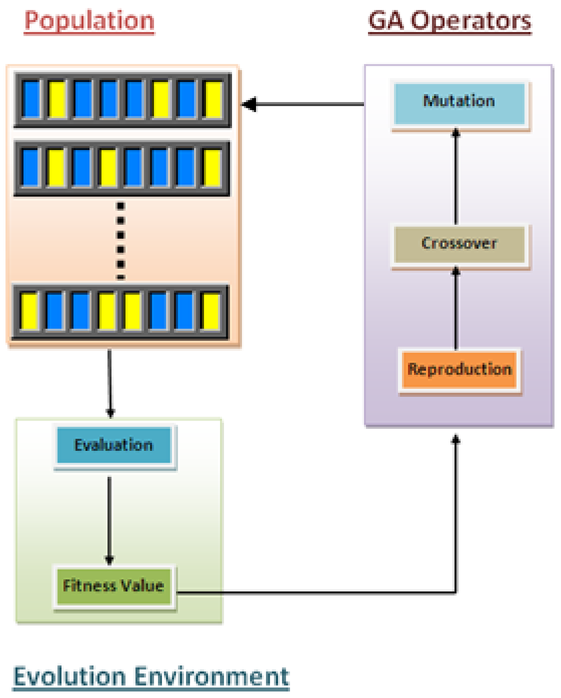


Fig. 1 Directed Acyclic Graph of 10 nodes and their dependencies on each other

4. Methodology

In our work, we implemented Genetic algorithm for solution of multiprocessor task scheduling problem. Detail block diagram (Fig.) represents the sequence of operations to get the desired results.



Typically, a genetic algorithm consists of the following steps:
GA1: Initialization –initialize the population.
GA2: Evaluation –evaluate each chromosome using fitness function.
GA3: Genetic operations –Select the parent and apply genetic operators on them to produce new chromosomes.
GA4: Repeat steps GA2 and GA3 until termination condition reached.

From the above steps, we can see that genetic algorithms utilize the concept of survival of the fittest; passing “good” chromosomes to the next generation, and combining different strings to explore new search points. This is a searching process based on the laws of natural selection and genetics. Usually, a simple GA consists of three operations: Selection, Genetic Operation, and Replacement The population comprises a group of chromosomes from which candidates can be selected for the solution of a problem. Initially, a population is generated randomly. The fitness values of the all chromosomes are evaluated by calculating the objective function in a decoded form (phenotype). A particular group of chromosomes (parents) is selected from the population to generate the offspring by the defined genetic operations. The fitness of the offspring is evaluated in a similar fashion to their parents. The chromosomes in the current population are then replaced by their offspring, based on a certain replacement strategy. Such a GA cycle is repeated until a desired termination criterion is reached (for example, a predefined number of generations is produced). If all goes well throughout this process of simulated evolution, the best chromosome in the final population can become a highly evolved solution to the problem.

5. Result & Discussion

In our work, we implemented an algorithm for solution of multiprocessor task scheduling problem. We assume that the number of tasks considered in the problem is 10 and processors are 3, but these can be varied. There can be maximum 6 number of predecessors of a node shown in Fig. 1. For performance evaluation of our algorithm we take different parameters values for different parameters. As two important GA control parameters, crossover probability (pc) and mutation probability (pm) affect the performance of GAs drastically. A set of good GA parameters help in improving the ability of a GA to search for near global optimal solutions.

We also examines the effects of GA numerical parameters on its performance in terms of average fitness found for this specific problem domain. It also describes the observed behavior of genetic algorithm in case of task scheduling. How the average fitness increases or decreases when the parameters of GA are varied? Some results are taken here after varying some parameters of genetic algorithm. These parameters are:

1. Population size (POP)
2. Number of generations (GEN)
3. Crossover probability (Cp)

This chapter is divided into two parts:
Part-a: Fitness Versus Generation
Part-b: Finish Time Versus Generation

Parameters used for implementation:
Population size= 20
Probability of Crossover= 0.5
Probability of Mutation: 0.5
Maximum no. of generation: 20
Note: These are the "initial" values for all parameters. All of these parameters can be varied. Chapter 5 gives results and their corresponding parameters values.

PART-A: Fitness Versus Generation

5.1 Analyzing The Effect of Varying GA Parameters on Average Fitness When Crossover and Mutation Probability is 0.5

The relevant data about these graphs is given in Annexure.
Fig. 5.1 shows the Graph between Generation and Average fitness for the following parameters values:

Parameters Values:
Population size=20 Number of generations=20 Crossover probability=0.5 Mutation probability=0.5

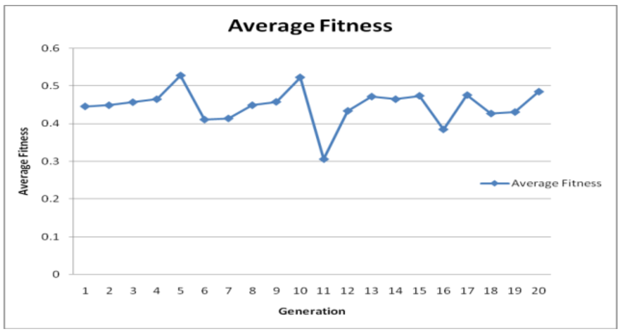


Fig. 5.1: Fitness vs. Generation
(when POP=20, GEN=20, Cp=0.5, Mp=0.5)

As the generation increases, average fitness increases and decreases with the generation. This effect is shown in the Graph below. Average and minimum fitness values are shown below:

Average Fitness Value	**Minimum Fitness Value**
0.528	0.004

Fig. 5.2 shows the Graph between Generation and Average fitness for the following parameters values:

Parameters Values:
Population size=30 Number of generations=20 Crossover probability=0.5 Mutation probability=0.5

Here, the GA parameter "population size" is changed, in the previous graph the population size was 20 and in this case population size is increased. Here it is 30. The rest of all parameters will be same for this case.

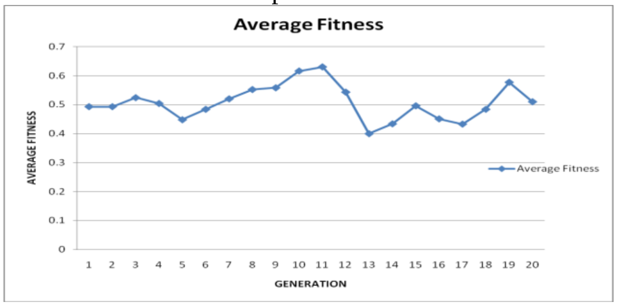


Fig. 5.2: Fitness vs. Generation
(when POP=30, GEN=20, Cp=0.5, Mp=0.5)

Average Fitness Value	**Minimum Fitness Value**
0.616	0.003

This change affects the "Average Fitness" of chromosomes. It can be seen that in the last case the average fitness value is 0.528 and in this case the average fitness is 0.616 i.e. average fitness is increased after varying the population size 30 instead of 20.

5.2 Analyzing The Effect of Varying GA Parameters on Average Fitness When Crossover Probability is 0.33 and Mutation Probability 0.67

Fig 5.3 shows the Graph between Generation and Average fitness for the following parameters values:

Parameters Values:
Population size=20 Number of generations=20 Crossover probability=0.33 Mutation probability=0.67

This graph shows the effect of crossover and mutation probabilities on average fitness. Here the crossover operator is applied on population/3 part of the chromosomes and mutation is applied on the rest of the chromosomes.

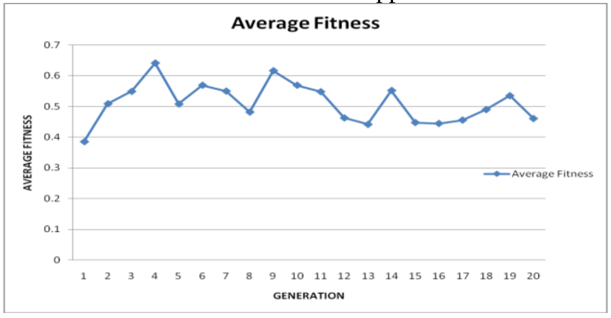


Fig. 5.3: Fitness vs. Generation
(when POP=20, GEN=20, Cp=0.33, Mp=0.67)

It is observed that when the crossover and mutation probabilities are changed, it directly affects the average fitness value. Average and minimum fitness is given below:

Average Fitness Value	**Minimum Fitness Value**
0.616	0.004

Fig 5.4 shows the Graph between Generation and Average fitness for the following parameters values:

Parameters Values:
Population size=30 Number of generations=20 Crossover probability=0.33 Mutation probability=0.67

This graph shows the variation in "population size". If we take population size 30, as effect of this the average fitness will be decreased. It can be seen here, in the last case when the population size is 20 the average fitness was 0.616 and here is this case after changing the population size 20 to 30, will directly decrease the average fitness.

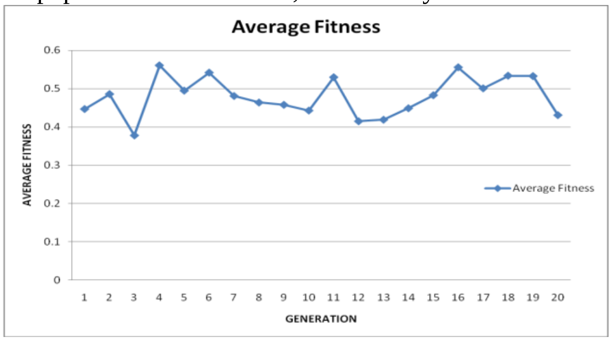


Fig. 5.4: Fitness vs. Generation
(when POP=30, GEN=20, Cp=0.33, Mp=0.67)

Average and minimum fitness values are:

Average Fitness Value	**Minimum Fitness Value**
0.561	0.003

This graph shows the variation in "population size". If we take population size 30, as effect of this the average fitness will be decreased. It can be seen here, in the last case when the population size is 20 the average fitness was 0.616 and here is this case after changing the population size 20 to 30, will directly decrease the average fitness.

5.3 Analyzing The Effect of Varying Ga Parameters on Average Fitness When Crossover Probability is 0.67 And Mutation Probability 0.33

Fig 5.6 shows the Graph between Generation and Average fitness for the following parameters values:

Parameters Values:
Population size=20 Number of generations=20 Crossover probability=0.67 Mutation probability=0.33

The graph shown below gives the average fitness of 0.532 which is not much high. This average fitness can be improved by varying the parameters (population size or no. of generations) for the same crossover and mutation probability.

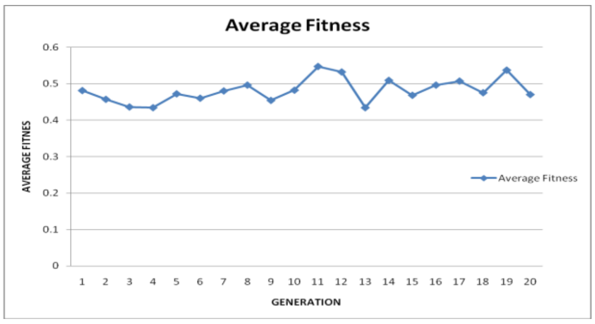


Fig. 5.6: Fitness vs. Generation
(when POP=20, GEN=20, Cp=0.67, Mp=0.33)
Average fitness and minimum fitness values are:

Average Fitness Value	**Minimum Fitness Value**
0.532	0.004

Fig 5.7 shows the Graph between Generation and Average fitness for the following parameters values:

Parameters Values:
Population size=30 Number of generations=20 Crossover probability=0.67 Mutation probability=0.33

The graph shown below gives the average fitness of 0.616 which is also not much high. This average fitness can be improved by varying the parameters (population size or no. of generations) for the same crossover and mutation probability.

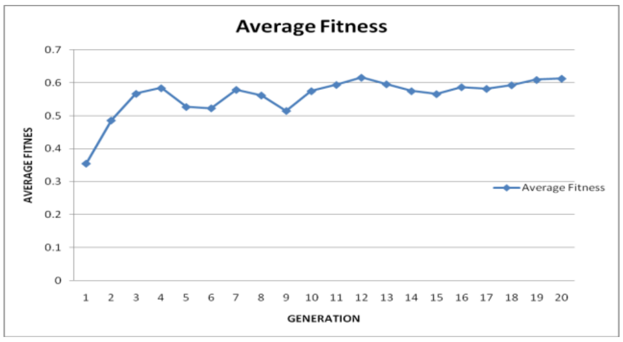


Fig. 5.7: Fitness vs. Generation
(when POP=30, GEN=20, Cp=0.67, Mp=0.33)

Average fitness and minimum fitness values are:

Average Fitness Value	**Minimum Fitness Value**
0.616	0.004

5.4 Analysing The Effect of Varying GA Parameters on Average Fitness When Crossover Probability is O.75 and Mutation Probability 0.25

Fig 5.8 shows the Graph between Generation and Average fitness for the following parameters values:

Parameters Values:
Population size=20 Number of generations=20 Crossover probability=0.75 Mutation probability=0.25

This graph shows the effect of crossover and mutation probability on the performance of genetic algorithm. It is observed that all the parameters of GA affect its performance, but crossover and mutation highly affects its performance.

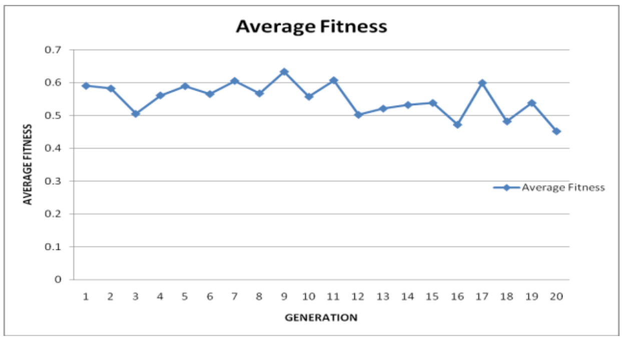


Fig. 5.8: Fitness vs. Generation
(when POP=20, GEN=20, Cp=0.75, Mp=0.25)

Average fitness and minimum fitness values are:

Average Fitness Value	**Minimum Fitness Value**
0.633	0.004

Fig 5.9 shows the Graph between Generation and Average fitness for the following parameters values:

Parameters Values:
Population size=30 Number of generations=20 Crossover probability=0.75 Mutation probability=0.25

As it can be seen above in the parameters values, population size is changed here from 20 to 30. And this graph is for population size 30. And it affects the performance of GA, the average fitness value is decreased here from 0.633 to 0.601.

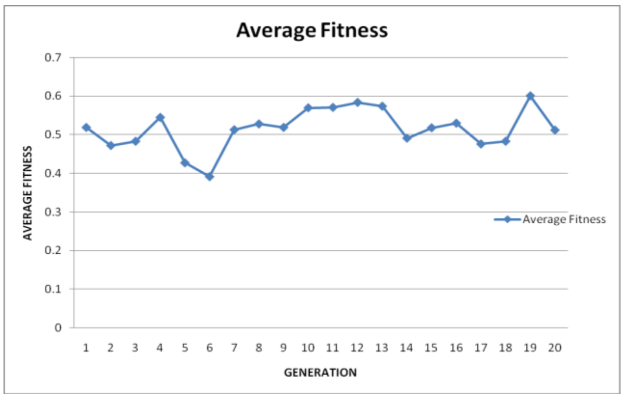


Fig. 5.9: Fitness vs. Generation
(when POP=30, GEN=20, Cp=0.75, Mp=0.25)

Average fitness and minimum fitness values are:

Average Fitness Value	**Minimum Fitness Value**
0.601	0.003

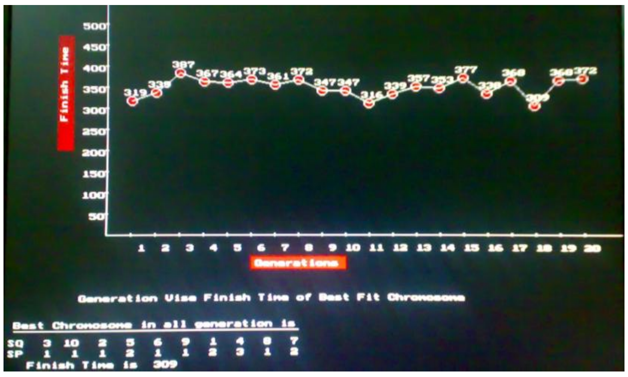
In our study we evaluate the changes in the performance of GA with respect to the changes in the control parameters.

PART-B: Finish Time Versus Generation

The snapshots shown below are captured on run time. The graphs are between finish time of chromosomes and generation. These graphs are taken after varying the crossover and mutation probability. Here, in these graphs, effects of crossover and mutation probability are shown on finish time calculation of chromosomes. How the finish time increases or decreases when the probability of crossover and mutation is varied?

Fig. 5.10 shows the graph between finish time and generation. The minimum execution time is shown in graph, is 309.

Parameters Values:
Population size=20 Number of generations=20 Crossover probability=0.50 Mutation probability=0.50



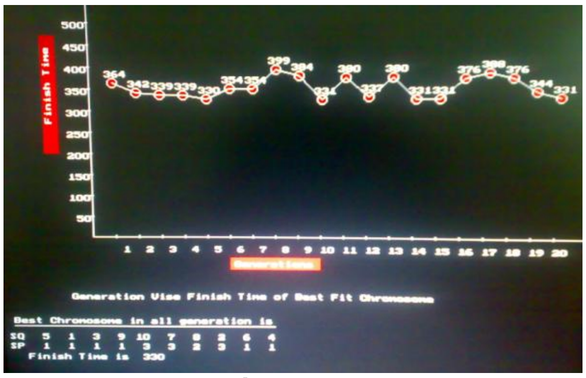
```
Generation Wise Finish Time of Best Fit Chromosome

Best Chromosome in all generation is
SQ   3  10   2   5   6   9   1   4   8   7
SP   1   1   1   2   1   1   2   3   1   2
  Finish Time is   309
```

Fig. 5.10: Finish Time Vs. Generation
(When POP=20, GEN=20, Cp= 0.5, Mp=0.5)

Fig. 5.11shows the graph between finish time and generation. The minimum execution time is shown in graph, is 330 which is slightly different from the previous.

Parameters Values:
Population size=20 Number of generations=20 Crossover probability=0.67 Mutation probability=0.33



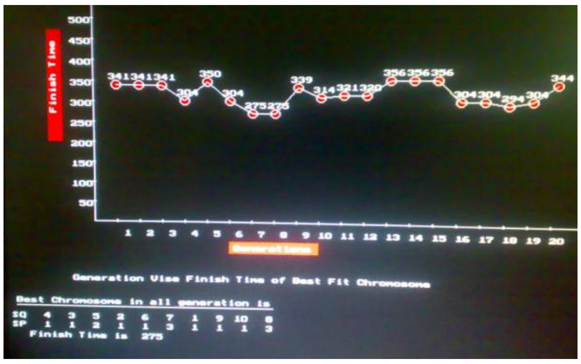
```
Generation Vise Finish Time of Best Fit Chromosome

Best Chromosome in all generation is
SQ   5  1  3  9  10  7  8  2  6  4
SP   1  1  1  1  3   3  2  3  1  1
  Finish Time is   330
```

Fig 5.11: Finish Time Vs. Generation
(When POP=20, GEN=20, Cp= 0.67, Mp=0.33)

Fig. 5.12 shows the graph between finish time and generation. The minimum execution time is shown in graph, is 275.

Parameters Values:
Population size=20 Number of generations=20 Crossover probability=0.75 Mutation probability=0.25



```
Generation Vise Finish Time of Best Fit Chromosome

Best Chromosome in all generation is
SQ   4  3  5  2  6  7  1  9  10  8
SP   1  1  2  1  1  3  1  1  1   3
  Finish Time is   275
```

Fig 5.12: Finish Time Vs. Generation
(When POP=20, GEN=20, Cp= 0.75, Mp=0.25)

Fig. No.	Population Size (POP)	No.Of Generations (GEN)	Crossover Probability	Mutation Probability	Minimum Execution Time
5.10	20	20	0.5	0.5	309
5.11	20	20	0.67	0.33	330
5.12	20	20	0.75	0.25	**275**

6. Conclusion

Genetic algorithm are better approach to find better solutions for task scheduling in multiprocessor as genetic algorithm are flexible algorithm for problem solving because parameter changes rapidly influence the performance. Traditional methods does not accept the change frequently due to this, problems are more complicated but genetic algorithm accept change and tries to find next best solution . As we observe in our work that genetic algorithm is highly influence by parameter like generation population, Mp, Cp, average fitness and finish time .We find that generation is directly proportional to average fitness when Mp, Cp change its effect the average fitness where as population remain same if Cp, Mp remains saturated and we population is increased then average fitness increases . In some case lesser value of Cp, and high value of Mp in comparison with Cp result out increase in the genetic algorithm performance. Cp and Mp also effect the execution time (finish time). Genetic algorithm is better than traditional rigid approach like DFS, BFS, TABU search etc. Genetic algorithm is simple to use ,requires minimum problem specific information and support dynamic environment and based on numeric computational parameter .Task scheduling in multiprocessor system using genetic algorithm is an efficient way due to the characteristics of Genetic Algorithm as : quality of solution ,robustness and guarantee for good solution effect of mutation probability on the performance on Genetic Algorithm.

References:

[1] Neelu Sahu, Sampada Satav "*Task Scheduling Using Compact Genetic Algorithm for Heterogeneous System*" [2012]

[2] Rachhpal Singh " *Modified Genetic Algorithm Approach to Optimize Task Scheduling on Heterogeneous Multiprocessor Parallel System using Node Duplication*"[2012]

[3] Mostafa r. Mohamed, Medhat h. A. Awadalla"*Hybrid Algorithm for Multiprocessor Task Scheduling*" IJCSI International Journal of Computer Science Issues, Vol. 8, No. 2, May 2011

[4] Sachi Gupta, GAurav AGArwal, "*Task Scheduling in Multiprocessor System Using Genetic Algorithm*", Second International Conference on Machine Learning and Computing, IEEE 2010.

[5] Adel Manaa and Chengbin Chu, "*Scheduling multiprocessor tasks to minimize the makespan on two dedicated processors*", European Journal of Industrial Engineering, pp. 265 – 279, Volume 4, 2010.

[6] Intisar A.Majied Al-Said, Nedhal Al-Saiyd, Firas Turki Attia, "*Multiprocessor scheduling based on genetic algorithms*", 2009.

[7] Amir Masoud Rahmani and Mojtaba Rezvani, "*A Novel Genetic Algorithm for Static Task Scheduling in Distributed Systems*", 2009.

[8] Ali Pedram, "*A method for scheduling multi processing systems with genetic algorithm*", International Journal of Engineering and Technology Vol. 1, No. 2, June, 2009.

[9] S.N.Sivanandam, S.N.Deepa, "*Introduction to Genetic Algorithms*", Springer-Verlag Berlin Heidelberg, 2008.

[10] Amir Masoud Rahamani, Mohamad Ali Vahedi "*A novel Task Scheduling in Multiprocessor Systems with Genetic Algorithm by using Elitism stepping method*", 2008.

[11] Yajun Li, Yuhang Yang, Maode Ma, Rongbo Zhu, "*A Problem-Specific Genetic Algorithm for Multiprocessor Real-time Task Scheduling*", The 3rd Intetnational Conference on Innovative Computing Information and Control (ICICIC'08), IEEE 2008.

[12] Peyman Almasi Nejad, Ahmad Farahi, Davood Karim ZadeGAn Moghadam, Reza AsGAry Moghadam, "*An Intelligent Method for Multi Processor Scheduling using Genetic Algorithms*", International Conference on MultiMedia and Information Technology, IEEE 2008.

[13] Dr. Franz Rathlauf, "*Representations for Genetic and Evolutionary Algorithms*", 2nd edition, @ Springer. 2006.