International Journal of Computer Techniques -– Volume 2 Issue 1, 2015

RESEARCH ARTICLE OPEN ACCESS

Comparative Study of Genetic Operators and Parameters for Multiprocessor Task Scheduling

Bhawna Gupta
Department of Computer Science & Engineering
I.T.S Engineering College, Greater Noida

--**************************************--

Abstract :

Task scheduling in multiprocessor systems is one of the main factors of systems performance. In this paper, the problem of scheduling of tasks in Multiprocessor system is described as finding optimal sequence of the task (called schedule) such that makespan can be minimized. Finding the optimal solution of scheduling the tasks into the processors is NP- Complete. Genetic Algorithm (GA) has been developed as a powerful tool for solving constrained optimization problems. This paper presented the results of experimental comparison of six different combinations of crossover (i.e. PMX, OX and CX) and mutation (i.e. Insertion, Swap) operators and also analyzed the effect of varying the genetic control parameters(like population size, crossover fraction, No. of generations and elite count) on objective function for considered scheduling problem.

Keywords- **Multiprocessor task scheduling (MPTS), Genetic Algorithm (GA), Crossover, Mutation, Makespan, Genetic control parameters.**

--**************************************--

I. INTRODUCTION

The study of scheduling started about sixty years ago, being initiated by seminal papers by Johnson (1954) and Bellman (1956) [1]. Scheduling is defined as the assignment of all tasks to available processors with the objective of optimizing one or several performance measures. Scheduling on multiprocessor system is computationally difficult and tedious problem. The importance of the multiprocessor task scheduling problem led to several comparative studies. Several heuristics & metaheuristics have been developed for the solution of the multiprocessor task scheduling [2].

In this paper task scheduling problems in multiprocessor system is considered in flow-shop environments. The permutation flow shop scheduling problem (PFSP) is a special case of flow shop scheduling problem where the processing order of the jobs is same on all the processors [3]. The precedence relation between two jobs i and j represents the situation where job j cannot start until job i completes. The scheduling algorithms can be rated based on different parameters like flowtime, communication cost, reliability cost and makespan [4]. The scheduling problem proposed here is a single-objective problem i.e. minimizing the makespan of the system. Makespan is the time taken for a multiprocessor system to finish the last task.

There are different algorithms available for optimization of objective functions such Evolutionary Programming, Evolutionary Strategies, Simulated Annealing, Tabu search, Particle swarm optimization, Ant colony search and Genetic Algorithm[5]. In this paper genetic algorithm is used as optimization algorithm for tasks scheduling in multiprocessors system. Genetic Algorithms are the heuristic search and optimization techniques that mimic the process of natural evolution. Genetic algorithms are powerful methods of optimization used successfully in different problems. Their performance is depending on the choice of genetic operators especially, the selection, crossover and mutation operators and on the different genetic control parameters like population size, crossover fraction and elite count etc. This paper has presented analysis of various combinations of existing crossover and mutation operators with different values genetic control parameters, for minimizing the value of makespan for multiprocessor task scheduling problem.

II. MULTIPROCESSOR TASK SCHEDULING PROBLEM

In the present work, a multiprocessor task scheduling problem with 'n' tasks & 'm' processors has been considered with the objective of optimizing the make span. The various assumptions considered for multiprocessor task scheduling (MPTS) are ([1], [6]):

All the tasks and processors are available at time Zero.

Pre-emption is not allowed.

Processors never break down.

- All processing time on the processors are known, deterministic, finite and dependent on sequence of the tasks to be processed.
- Each processor is continuously available for assignment.
- The first processor is assumed to be ready whichever and whatever task is to be processed on it first.
- Processors may be idle
- Splitting of task or task cancellation is not allowed

III. GENETIC ALGORITHM

Genetic algorithm (GA) is a technique that is applied to a number of optimization problems to obtain optimal solutions. A Genetic Algorithm (GA) is a search heuristic that mimics the process of natural evolution.

Basic genetic algorithm.

```
1. BEGIN
2. INITIALIZE Generate random population of n chromosomes
3. FITNESS Evaluate the fitness f(x) of each chromosome x in the population
4. NEW POPULATION Create a new population by repeating following steps until the new population is complete
    a. SELECTION Select parents according to their fitness
    b. CROSSOVER Cross over the parents to form new offspring (children).
    c. MUTATION Mutate new offspring at locus (position in chromosome).
    d. ACCEPTING Place new offspring in the new population.
5. REPLACE Use new generated population for a further run of the algorithm
6. VERIFY If the end condition is satisfied, stop, and return the best solution in current population
7. ITERATE Goto step 2
```

A. *Encoding Schemes*

The encoding scheme is a key issue in any GA because it can severely limit the window of information that is observed from the system [8]. Traditionally, solutions are represented in binary as strings of 0's and 1's, but other encodings are also possible [7]. Here Permutation encoding is used.

B. *Initial Population*

The initial population of candidate solutions is usually generated randomly across the search space. However, domain-specific knowledge or other information can be easily incorporated [9]. The population size is typically problem dependent and has to be determined experimentally [10].

C. *Fitness Evaluation*

Each chromosome contains a string, called genes (tasks) and has an associated value called a fitness value, which is evaluated by a fitness function. Several optimization criteria can be considered for the said problem. The elementary criterion is that of minimizing the makespan, that is, the time when processor finishes the last task (Cj).

$$\text{make span} = \min_{S_j \in \text{Sched}}\{\max C_j\}$$

D. *Selection*

GA uses selection operator to select the superior and eliminate the inferior thus imposes the survival-of-the-fittest mechanism. Selection of proper couples from parent is performed according to their fitness value. For the present work fitness values have been evaluated for all chromosomes and good chromosomes has been selected through roulette wheel strategy [3].

E. *Genetic search operators*

The selection process and the crossover and mutation operators establish a balance between the exploration and exploitation of the search space which is very adequate for a wide variety of problems [12].

Crossover: Crossover is a mechanism that produces new offspring that have some parts of both parent's genetic material [11]. There are many ways of accomplishing this mechanism but competent performance depends on a properly designed crossover mechanism [9]. Partially mapped crossover(PMX), Cycle crossover(CX) and

Order crossover(OX) has been implemented for the said multiprocessor task scheduling problem.

1) Mutation: Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to next [13]. There are many variations of mutation, but it usually involves one or more changes being made to an individual's trait or traits. In other words, mutation performs a random walk in the vicinity of a candidate solution. Swap and insertion mutation has been used for analysis of the said problem.

F. Termination Conditions

For the said multiprocessor task scheduling problem if generation has reached some predefined generation value then this process will stop and the best chromosome will be the answer.

IV. IMPLEMENTATION RESULTS

Experiments are conducted to evaluate the performance of GA, in minimizing the objective function (i.e. make span of the processor). Genetic algorithm is as good as its operators and parameters i.e. for improving the performance of Genetic Algorithm optimal parameter selection are required. In previous work the results have been calculated for different combinations of operators with some parameters (i.e. crossover probability, elite count, size of population, number of generations etc.) have been fixed. The present work considers the optimization of different parameters and operators of genetic algorithm for the said multiprocessor task scheduling problem. The different parameters and different combinations of operators of genetic algorithm greatly determine the degree of solution accuracy.

The efficiency of genetic algorithm is closely related to control parameters. In the following experiments, genetic algorithms performance have been tested for parameters such as population size, the crossover probability, elite count and Number of generations with different combinations of operators. The following combinations of operators have been used in the experimentation:-

- PMX and Insertion Mutation(IM)
- PMX and Swap Mutation(SM)
- CX and Insertion Mutation
- CX and Swap Mutation
- OX and Insertion Mutation
- OX and Swap Mutation

GA algorithm is implemented using MATLAB at command line. MPTS (in permutation flow shop scheduling) have been implemented.

To compare statistically all the combinations of operators,and to get a precise value of genetic control parameters one value of burst time is generated randomly and saved. Then all the combinations of operators and some different values of genetic control parameters are applied one by one on the saved value.

This process is repeated for multiple values of burst time to get more precise view about the results (i.e. to check the accuracy of best combination). From multiple different values of burst time one value is shown in table I, for four processors and ten tasks, whose results are explained below.

Table I
EXAMPLE of BURST TIME

No. of Tasks → / **No. of Processors** ↓	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	22	13	3	32	5	21	14	18	29	6
2	2	13	36	2	15	23	24	17	26	6
3	12	13	3	6	25	21	14	18	9	4
4	1	17	6	4	35	21	14	18	2	4

A. Effect of population size on the performance of genetic algorithm

To analyse the effect of population size on the performance of genetic algorithm following parameter setting is done

No. of generations	100
Elite Count	2
Crossover Probability	0.6
Time Limit	Infinite
Fitness Limit	-Infinite
Stall Generation Limit	Infinite
Stall Time Limit	Infinite

Table II
REPRESENTS THE MAKESPAN VALUES OF DIFFERENT COMBINATION ON THE VARIATION OF POPULATION SIZE

Operat-ors Combin-ations → / **Populat-ion Size**	**PMX & IM**	**PMX & SM**	**CX & IM**	**CX & SM**	**OX & IM**	**OX & SM**
50	182.6	183.3	184.6	183	183.6	182
100	181.3	182	182.3	180.3	181.3	181.6
150	182	180.6	182.6	181.3	182.6	182
200	181	181.3	182.6	181.3	181.6	182
250	181	180.6	182.6	180.6	180	181
300	180	180.6	183.3	182	180	180

shows that as we increase the population size GA performs better but at the cost of computation time. So, an appropriate combination of crossover and mutation operator is also required so that we can get best results without compromising cost. So, after testing multiple combinations of operators for this particular approach, it has been observed that order crossover and insertion mutation are giving best results for the said problem. This combination has given good results even for less population size. Population size is one of the most important parameters of GA. The above table II, shows the value of makespan with the variation in population size and genetic operators. The value of makespan in the above table

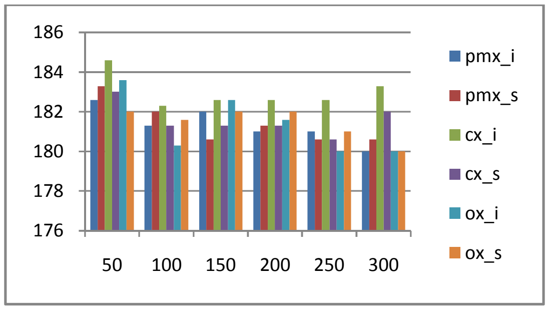


Figure 1. Effect of population size on makespan

It can be easily viewed from the above figure that order crossover and insertion mutation are giving best results even with less population size.

B. *Effect of Crossover probability on the performance of genetic algorithm*

To analyse the effect of crossover probability on the performance of genetic algorithm following parameter setting is done

No. of generations	100
Elite Count	2
Population Size	100
Time Limit	Infinite
Fitness Limit	-Infinite
Stall Generation Limit	Infinite
Stall Time Limit	Infinite

Here, it is observed that crossover probability affects the performance of genetic algorithm. For the said problem, if crossover probability is set to high value then performance degrades. It can be easily observed from figure 2, that crossover probability from 0.4 to 0.7 gives minimum values of makespan. In the specified range best performance is observed for every combination. Optimal value of make span is obtained at crossover probability values 0.4 and 0.6.

Table III
REPRESENTS THE MAKESPAN VALUES OF DIFFERENT COMBINATION ON THE VARIATION OF CROSSOVER PROBABILITY

Operat-ors → Combin-ations / Crossov-er Probabi-lity	PMX & IM	PMX & SM	CX & IM	CX & SM	OX & IM	OX & SM
0.1	180.6	183.3	181.3	180	182.6	182.6
0.2	180	183.3	181.3	182	182	182
0.3	180.6	182	182.6	182.3	180.3	181.3
0.4	180.6	182.6	180	180	180	180.6
0.5	181	181	181.6	181.3	180.3	181.6
0.6	181	180.6	181.3	181.3	180	181
0.7	182	181.3	181.3	180	180.3	182.6
0.8	182	183.3	180	182.6	181.3	182.3
0.9	184	183.3	184	183.3	182	182
1	184	184	182.6	183	183	186.3

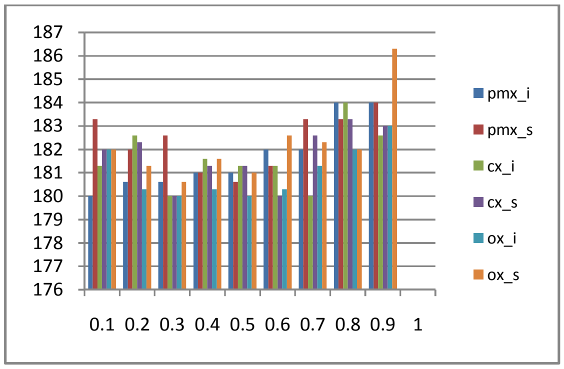


Figure 2. Effect of Crossover probability on makespan

C. *Effect of elite count on the performance of genetic algorithm*

To analyse the effect of elite count on the performance of genetic algorithm following parameter setting is done

Crossover Probability	0.6
No. of generations	100
Population Size	100
Time Limit	Infinite
Fitness Limit	-Infinite
Stall Generation Limit	Infinite
Stall Time Limit	Infinite

It is observed from the above table that the value of makespan increases with the increase in value of elite count that means performance degrades. Elite count value 2 is giving best results i.e. minimum makespan.

Table IV
REPRESENTS THE MAKESPAN VALUES OF DIFFERENT COMBINATION ON THE VARIATION OF ELITE COUNT

Operat-ors combin-ations → / elite count	PMX & IM	PMX & SM	CX & IM	CX & SM	OX & IM	OX & SM
1	184	182	185.3	183.3	183.6	180
2	181.3	180.6	184.6	182	180	180
3	183	181.6	183	182.6	181.3	180
4	182	182	182	181.6	182.6	181.3
5	182.6	182.6	182.6	182	180	181.3
6	182	182.6	182	180.6	182.6	183

This best value is associated with six different combination of crossover and mutation operator, and from all the different combinations, order crossover and insertion mutation are giving best results. It can also be observed from the experiments that, setting Elite count to a high value causes the fittest individuals to dominate the population, which can make the search less effective.

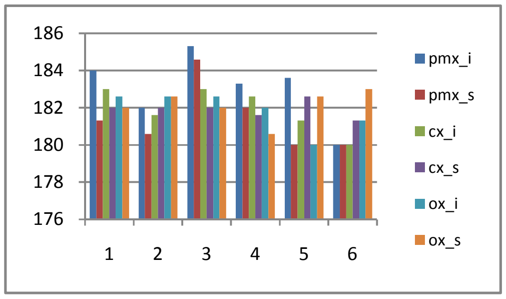


Figure 3. Effect of number of generations on makespan

D. Effect of number of generations on the performance of genetic algorithm

To analyse the effect of number of generations on the performance of genetic algorithm following parameter setting is done

Crossover Probability	0.6
Elite Count	2
Population Size	100
Time Limit	Infinite
Fitness Limit	-Infinite
Stall Generation Limit	Infinite
Stall Time Limit	Infinite

Table V
REPRESENTS THE MAKESPAN VALUES OF DIFFERENT COMBINATION ON THE VARIATION OF NUMBER OF GENERATIONS

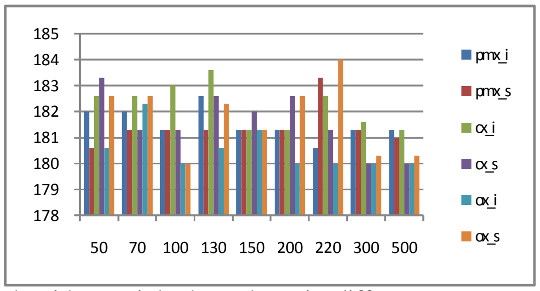
Operat-ors Combi-nations → / No. of Genera-tions	PMX & IM	PMX & SM	CX & IM	CX & SM	OX & IM	OX & SM
50	182	180.6	182.6	183.3	180.6	182.6
70	182	181.3	182.6	181.3	182.3	182.6
100	181.3	181.3	183	181.3	180	180
130	182.6	181.3	183.6	182.6	180.6	182.3
150	181.3	181.3	181.3	182	181.3	181.3
200	181.3	181.3	181.3	182.6	180	182.6
220	180.6	183.3	182.6	181.3	180	184
300	181.3	181.3	181.6	180	180	180.3
500	181.3	181	181.3	180	180	180.3

It can be observed from Table V, that as the number of generations increases, GA performs better (i.e. value of make span decreases) or some time remains same. So for given MPTS problem the value 500 for number of generations is showing minimum makespan for all combination of operators. But choosing this value for number of generations will increase the computation time. So the value for number of generations has been chosen in such a way that the objective function is minimized without compromising the computational time. This can be done by choosing a combination of operators which is minimizing the objective function even with lesser number of generations. So order crossover and insertion mutation is the appropriate choice.

Figure 4. Effect of number of generations on makespan

E. CONCLUSIONS

In this paper, a genetic algorithm for MPTS in permutation flow shop scheduling environment has been implemented. The solution of genetic



algorithm mainly depends on its different operators and parameters like type of crossover, mutation; crossover probability etc. and every problem have specific GA parameters. The performance of the GA has been evaluated with the variation of combinations of genetic operators along with the variation in genetic control parameters. It has been concluded that:

- Genetic algorithm performs better with the increase in population size. Order crossover and insertion mutation outperformes even with less population size.
- For the said problem an even mixture of crossover and mutation children gives better results instead of any of them alone.
- Small value of elite count makes the search more efficient.
- Increasing the number of generations enables the genetic algorithm to explore the search space and thereby obtaining better results. Order crossover and insertion mutation is giving best results even with lesser number of generations which other combinations has not given even with higher number of generations.

F. REFERENCES

[1] Johnson, S.M. "Optimal two- and three-stage production schedules with setup times included", Naval Research Logistics Quarterly 1, 1954, 61–68.

[2] Dhingra S., Bal Gupta S.B., Biswas R. "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 Vol:8 No:4, 2014

[3] Verma R. and Dhingra S. (2011), "Genetic Algorithm For Multiprocessor Task Scheduling" IJCSMS International Journal of Computer Science and Management Studies, Vol. 11, Issue 02, ISSN (Online): 2231-5268.

[4] Devi M.R. and Anju A. "Multiprocessor Scheduling of Dependent Tasks to Minimize Makespan and Reliability Cost Using NSGA-II" , International Journal in Foundations of Computer Science & Technology (IJFCST), Vol.4, No.2, March 2014

[5] Patel F.M., Panchal N.B. "The Matlab Solution of Mathematic Equation Using Genetic Algorithm for Optimum Result", GJRA - Global Journal For Research Analysis, Volume:3, Issue : 1, ISSN No 2277 – 8160, Jan 2014.

[6] Dhingra A. & Chandna P., "Hybrid Genetic Algorithm for Multicriteria Scheduling with Sequence Dependent Set up Time," International Journal of Engineering (IJE), Volume (3): Issue (5), 2009.

[7] Panwar P., Chauhan S., "Optimization of Multiprocessor Scheduling using Genetic Algorithm", International Journal of Computer Applications (0975 – 8887) Volume 78 – No.4, September 2013.

[8] Koza J.R., "Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems," Report No. STAN- CS-90-13 14, Stmdford University, 1990.

[9] Vishwanath A., Vulavala R., Prabhu S.U., "Task Scheduling in Homogeneous Multiprocessor Systems Using Evolutionary Techniques", International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 2, February 2014.

[10] Harsora V. and Dr. Shah A., "A Modified Genetic Algorithm for Process Scheduling in Distributed System," IJCA Special Issue on "Artificial Intelligence Techniques - Novel Approaches & Practical Applications" AIT, 2011.

[11] Gupta S., Agarwal G., and Kumar V., "An Efficient and Robust Genetic Algorithm for Multiprocessor Task Scheduling", International Journal of Computer Theory and Engineering, Vol. 5, No. 2, April 2013.

[12] Ortiz-Boyer D., "A Crossover Operator for Evolutionary Algorithms Based on Population Features," Journal of Artificial Intelligence Research 24 1-48, 2005.

[13] Garg R., mittal S., "Optimization by Genetic Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 4, April 2014.