Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Analysis of Genetic Algorithm for Multiprocessor Task Scheduling Problem

Bhawna Gupta*, Sunita Dhingra
Department of Computer Science & Engineering
University Institute of Engineering & Technology
Maharishi Dayanand University Rohtak-124001 Haryana INDIA

Abstract - Multiprocessor task scheduling (MPTS) is an important and computationally difficult problem. Multiprocessors have emerged as a powerful computing means for running real-time applications. The problem of task scheduling on multiprocessor systems is known to be NP-complete in general.Genetic algorithm (GA) is one of the widely used techniques for constrained optimization. Genetic algorithms are known to provide robust, stochastic solutions for numerous optimization problems. The results of experimental comparison of six different combinations of crossover (i.e. PMX, OX and CX) and mutation (i.e. Insertion, Swap) operators for the MPTS are presented. In this paper, the experimental results shows that PMX and Insertion Mutation combination enables to achieve a better solution than other operators combination tested.

Keywords- Genetic Algorithm (GA), crossover, mutation, Multiprocessor task scheduling (MPTS), permutation flow shop scheduling.

I. Introduction

Multiprocessor task scheduling problem is a generalization of the classical machine scheduling problem as it allows tasks to be processed on more than one processor at a time [1]. It is also an NP-hard optimization problem. Given a set J of jobs where job j_i has length l_i and a number of processors m, what is the minimum possible time required to schedule all jobs in J on m processors such that none overlap [2]. The processes are unrelated and each one can be scheduled without regard to the other ones. Genetic algorithms (GAs) are general purpose search algorithms which uses principles inspired by natural genetic populations to evolve solutions to problems[3].This technique is used in computing to find true or approximate solutions to optimization and search problems. Genetic algorithms are powerful methods of optimization used successfully in different problems. Their performance is depending on the encoding scheme and the choice of genetic operators especially, the selection, crossover and mutation operators. A variety of these latest operators have been suggested in the previous researches [4]. This paper has presented analysis of various combinations of existing crossover and mutation operators, for minimizing the value of makespan for multiprocessor task scheduling problem.

II. Assumptions

The various assumptions considered for multiprocessor task scheduling (MPTS) are ([1], [5]):

- 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

Outline of 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 [6]. Here Permutation encoding is used. Each task is present and appears only once in the schedule. In permutation encoding, every chromosome is a string of numbers that represent a position in a sequence. Permutation encoding is useful for ordering problems.

B. Initial Population

The space of all feasible solutions (the set of solutions among which the desired solution resides) is called search space (also state space). Genetic algorithm search many nodes in search space. If we are solving a problem, we are usually looking for some solution which will be the best among others. Looking for a solution is then equal to looking for some extreme value (minimum or maximum) in the search space.

This requires us to generate an initial population of the search node randomly. The population size is typically problem dependent and has to be determined experimentally [7].

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).

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

D. Selection

GA uses selection operator to select the superior and eliminate the inferior. The individuals are selected according to their fitness value. Once fitness values have been evaluated for all chromosomes, good chromosomes can be selected through rotating roulette wheel strategy [1].

Example – If, a marble is thrown in the roulette wheel, then the chromosome where it stops is selected. Clearly, the chromosomes with bigger fitness value will be selected more times.

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 [8].

1) Crossover: The search of the solution space is done by creating new chromosomes from old ones. The most important search process is crossover. The crossover operator combines the genes of two or more parents to generate better offspring. It is based on the idea that the exchange of information between good chromosomes will generate even better offspring, if it takes the best characteristics from each of the parents ([8], [9]). It ensures that individuals in the parent generation will not reappear in the next generation.

This paper presented three crossover operators i.e. Partially mapped crossover, Cycle crossover and Order crossover, for the said multiprocessor task scheduling problem.

2) Mutation: Mutation is performed after crossover. The role of mutation in GAs is that of restoring lost or unexplored genetic material into the population. It prevents the premature convergence of the GA to suboptimal solutions [10]. The mutation operator randomly selects a position in the chromosome and changes the corresponding allele, thereby modifying information. The need for mutation comes from the fact that as the less fit members of successive generations are discarded; some aspects of genetic material could be lost forever.

Swap and insertion mutation has been used for analysis of the said problem.

F. Termination Conditions

Stopping criteria for the said multiprocessor task scheduling problem is number of generations. Here, the processes of fitness computation, selection, crossover, and mutation are executed for a minimum number of iterations [11]. The best string seen up to the last generation provides the solution for multiprocessor task scheduling problem. Elitism has been implemented at each generation to preserve best chromosome.

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. So choice of appropriate operators is an important task. The paper presented a genetic algorithm analysis by applying several combinations of crossover and mutation. The following combinations of operators have been used in the experimentation:-

- PMX and Insertion Mutation
- PX and Swap Mutation
- 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. We have implemented MPTS (in permutation flow shop scheduling).

These are the various parameters of genetic algorithm for analyzing the effects of different combinations of crossover and mutation on the performance of genetic algorithm.

Crossover Probability	0.8
Elite Count	2
No. Of Generations	20
Population Size	100
Time Limit	Infinite
Fitness Limit	-Infinite
Stall Generation Limit	Infinite
Stall Time Limit	Infinite

To compare statistically all the combinations of operators, one value of burst time is generated randomly and saved. Then all the combinations is 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). Analysis of objective function with eight different value of burst time for all combinations is shown in table II.

From eight different values of burst time one value is shown in table I, for four processors and ten tasks

Table I

EXAMPLE of BURST TIME

No. of Tasks → / **No. of Processors ↓**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	8	13	3	5	10	19	21	3	23	4
2	3	5	17	25	27	7	1	2	6	11
3	1	10	9	26	11	8	4	31	2	19
4	1	2	16	6	4	9	18	24	32	5

While calculating the results for different combinations some parameters (i.e. crossover probability, elite count, size of population, number of generations etc.) have been fixed.

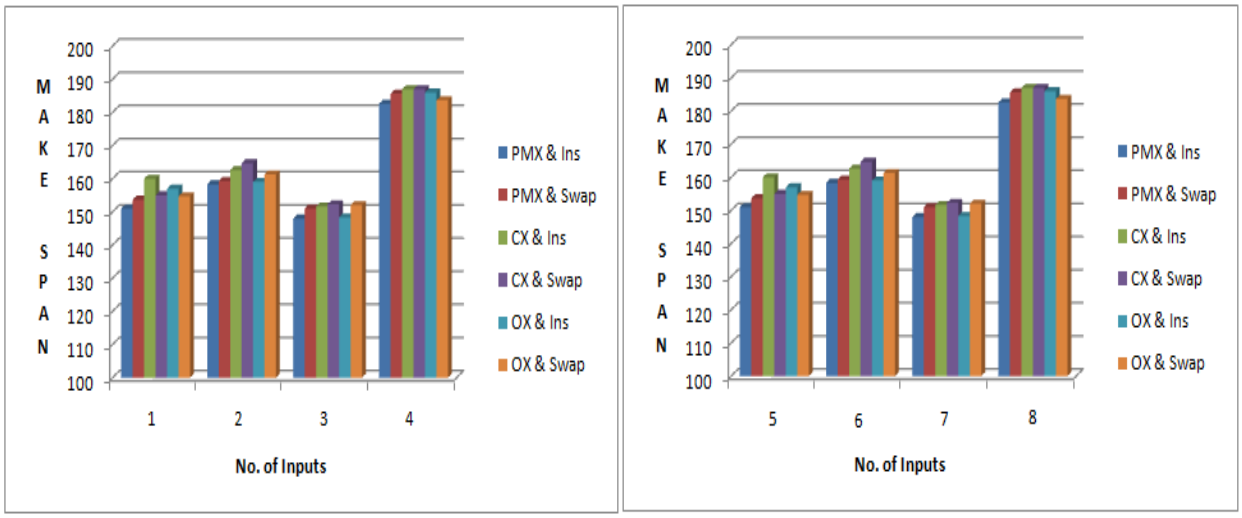
Table II

REPRESENTS AVERAGE RESULTS for ALL COMBINATIONS

S.No.	**PMX & Insertion mutation**	**PMX & Swap mutation**	**CX & Insertion mutation**	**CX & Swap mutation**	**OX & Insertion mutation**	**OX & Swap mutation**
1	151	153.667	160	155	157	154.6667
2	158.3333	159.3333	162.6667	164.6666	159	161.3333
3	148	151	151.6667	152.3333	148.3333	152
4	182.6667	185.6667	187	187	186	183.6667
5	728	734.6667	730.6667	730.6667	735	729

6	632.667	645.3333	641.6667	674.6667	648.6667	648.6667
7	628	647.6667	643	643	632	642
8	823	831.3333	825	825	825	825

Here, we can observe, for our MPTS (in permutation flow shop scheduling) problem, partially mapped crossover and insertion mutation gives best results.
So we can say, different combinations enables the genetic algorithm to explore search space differently and thereby obtaining better results.



(a) (b)

Fig. 1. Depicts change in make span for different combinations

Fig. 1.a) and 1.b) shows the variations in values of makespan for different combinations.
Six different colored bars represents different combinations of crossover and mutation and multiple bars of same color means different inputs for a particular combination.

A. 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	0.8
Elite Count	2
Population Size	100
Time Limit	Infinite
Fitness Limit	-Infinite
Stall Generation Limit	Infinite
Stall Time Limit	Infinite

Table III
REPRESENTS THE VALUES OF DIFFERENT COMBINATION ON THE VARIATION OF NUMBER OF GENERATIONS

No. of generations	**PMX & Insertion mutation**	**PMX & Swap mutation**	**CX & Insertion mutation**	**CX & Swap mutation**	**OX & Insertion mutation**	**OX & Swap mutation**
20	148	151	153	153	153	149
50	146	146	153	152	146	151
100	146	146	151	153	149	146

It can be observed from table III, as the number of generations increases, GA performs better (i.e. value of make span decreases) or some time remains same. So we can say that, the setting of generations affects the behavior of the algorithm but at the cost of computational 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.

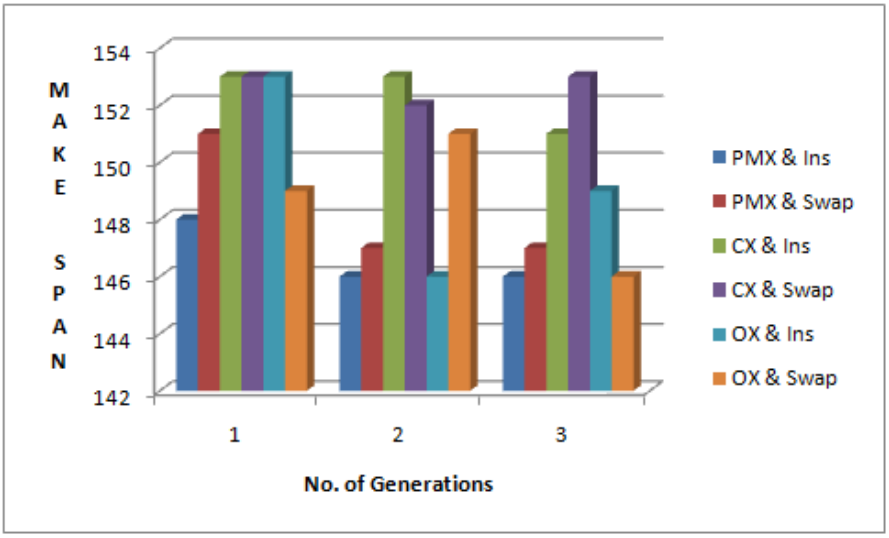


Fig. 2 Effect of number of generations on makespan

It is clear from the Fig. 2 that as the number of generations increases, the value of objective function is minimized. Number of generations is an important parameter for stopping condition.

V. Conclusions

In this paper, we have implemented a genetic algorithm for MPTS in permutation flow shop scheduling environment, where the processing order of the jobs is same on all the processors. Genetic Algorithms is applied for the solution of this problem. We evaluate the performance of the GA with the variation of combinations of genetic operators. Here we have concluded that:

- Analyzing genetic algorithm with different combinations of genetic operators results in different and better outputs.
- Partially mapped crossover and insertion mutation is giving best results as compared to other combinations for the said problem. This combination is minimizing the objective function (i.e. makespan) keeping other parameters as constant values.
- Partially mapped 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.

References

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

[2] M. R Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, pp 238. ISBN 0716710445, 1979.

[3] F. Herrera, M. Lozano & J.L. Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," *Artificial Intelligence Review 12: 265–319. Kluwer Academic Publishers. Printed in the Netherlands*, 1998.

[4] A. Otman & A. Jaafar, "A Comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem," *International Journal of Computer Applications* (0975 – 8887) Volume 31– No.11, October 2011.

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

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

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

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

[9] Y. Kaya, M. Uyar, and R. Tekin, "A Novel Crossover Operator for Genetic Algorithms: Ring Crossover," *presented at CoRR*, 2011

[10] K. Deep and H. Mebrahtu, "Combined Mutation Operators of Genetic Algorithm for the Travelling Salesman problem," *International Journal of Combinatorial Optimization Problems and Informatics*, Vol. 2, No.3, pp. 1-23, ISSN: 2007-1558, Sep-Dec 2011.

[11] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *the journals of Pattern Recognition* 33, pp. 1455-1465, (2000).

[12] R.K. Gupta, "Genetic Algorithms: An Overview," *Impulse*, vol. 1, 2006

[13] K.S.Tang, K.F.Man, S. Kwong and Q. he, "Genetic algorithm and their applications," *IEEE Signal Processing Magazine*, November 1996.

[14] T. Gonzalez and S. Sahni, "Flowshop and Jobshop schedules: complexity and approximation," *Operations research society of America*, vol. 26, No. 1, January- February 1978.

[15] S. Funk, J. Goossens, and S. Baruah ,"On-line Scheduling on Uniform Multiprocessors," *22nd IEEE Real-Time Systems Symposium (RTSS'01),* pp. 183-192, London, England, December, 2001.

[16] A. Dhingra & P. Chandna, "A bi-criteria M-machine SDST flow shop scheduling using modified heuristic genetic algorithm," *International Journal of Engineering, Science and Technology* Vol. 2, No. 5, pp. 216-225, 2010.

[17] S. Singh and E. A. Lodhi, "Study of Variation in TSP using Genetic Algorithm and Its Operator Comparison," *International Journal of Soft Computing and Engineering (IJSCE)* ISSN: 2231-2307, Volume-3, Issue-2, May 2013.

[18] W. Atmar, "Notes on simulation of evolution," *IEEE Trans. Neural Networks 5***,** 130–147, 1 (Jan. 1994).

