IJCSMS International Journal of Computer Science and Management Studies, Vol. 11, Issue 02, Aug 2011
ISSN (Online): 2231-5268
www.ijcsms.com

GENETIC ALGORITHM FOR MULTIPROCESSOR TASK SCHEDULING

[1]Ritu Verma, [2]Sunita Dhingra

[1]Deptt of CSE, UIET, MDU, Rohtak(India)
ritufbd@gmail.com

[2]Deptt of CSE, UIET, MDU, Rohtak(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 especially due to limitation of uni-processor system for not having sufficient enough capability to execute all the tasks. This paper describes multiprocessor task scheduling in the form of permutation flow shop scheduling, which has an objective function for minimizing the makespan. Here, we will conclude how the performance of genetic algorithms (value of the makespan of the schedule) varies with the variation of Genetic Algorithm (GA) control parameters (population size, crossover probability and mutation probability).

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

1. INTRODUCTION

Multiprocessor task scheduling problem is a generalization of the classical machine scheduling problem by allowing tasks to be processed on more than one processor at a time and it is motivated mainly by computer systems. We consider multiprocessor task scheduling problems in flow-shop environments. The permutation flow shop scheduling problem (PFSP) is a special case of flow shop problem where the processing order of the jobs is same on all the processors.

In solving the problem of scheduling n jobs on m processors, the objective is to minimize the make span (i.e. completion time C of the latest job) of the processor.

To minimize the make span, the elementary criterion is the time, when processor finishes the last job
i.e.

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

The work is based on the deterministic model that is the number of processors, the execution time of tasks is known in advance. In addition, the communication cost between two tasks is considered to be negligible and the multiprocessor system is non-preemptive that is the processors are homogeneous and each processor completes the current task before the new one starts its execution

2. ASSUMPTIONS

- All the jobs and machines are available at time Zero.
- Pre-emption is not allowed.
- Machines never break down.
- All processing time on the machine are known, deterministic, finite and dependent of sequence of the jobs to be processed.
- Each machine is continuously available for assignment.
- The first machine is assumed to be ready whichever and whatever job is to be processed on it first.

- Machines may be idle
- Splitting of job or job cancellation is not allowed.
- In-process inventory is allowed. If the next machine on the sequence needed by a job is not available, the job can wait and joins the queue at that machine.

The permutation flowshop represents a particular case of the flowshop scheduling problem, having as goal the deployment of an optimal schedule for n jobs on m machines.

As a consequence, for the permutation flowshop problem, considering the makespan as objective function to be minimized, solving the problem means determining the permutation which gives the smallest makespan value.

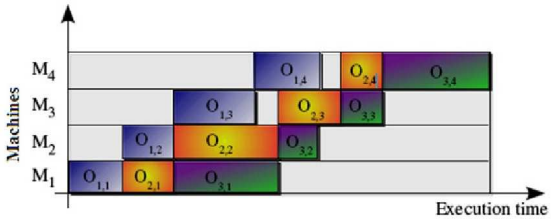

Fig: 1 Permutation flow shop scheduling

III. GENETIC ALGORITHM

A. WORKING PRINCIPLE OF A SIMPLE GA

1. Begin
2. INITIALIZE population with random candidate solutions;
3. EVALUAE each candidate:
4. REPEAT UNTIL (TERMINATION CONDITION is satisfied)
5. DO
   a. SELECT parents;
   b. RECOMBINE pairs of parents;
   c. MUTATE the resulting offspring;
   d. EVALUATE new candidate;
   e. SELECT individuals for the next generation;
6. DO
7. END

B. CONTROL PARAMETERS

The control parameters are crossover probability P_c, mutation probability, P_m, and population size that led to the best results.

The number of individuals with the best fitness values in the current generation that are guaranteed to survive to the next generation. These individuals are called elite children. Setting Elite count to a high value causes the fittest individuals to dominate the population, which can make the search less effective.

C. CODING OF SOLUTION

Here, Permutation encoding is used. Each task is present and appears only once in the schedule. A schedule is represented as a list of tasks executed on a processor and order of tasks in the list indicates the order of execution.

D. POPULATION INITIALIZATION

Genetic algorithm (GA) is inspired by Darwin's theory about evolution- the "survival-of-the fittest". It is the way of solving problems by mimicking processes used by nature: selection, crossover, mutation and accepting to evolve a solution to a problem.

WORKING PRINCIPLE OF A SIMPLE GA

1 Begin

2 INITIALIZE population with random candidate solutions;

3 EVALUAE each candidate:

4 REPEAT UNTIL (TERMINATION CONDITION is satisfied)

5 DO

i. SELECT parents;
ii. RECOMBINE pairs of parents;
iii. MUTATE the resulting offspring;

The next step in the GAs is the creation of the initial population. Number of processors, number of tasks and population size are required to generate initial population. The initial population consists of randomly generated individuals. The population size kept constant through the generations. A string of an integer represents the tasks is used to represent a schedule The population contains solution vectors called individuals of the population and each vector represents potential solution for the optimization problem. Many individual solutions are randomly generated to form an initial population.

E. FITNESS VALUE

The fitness of an individual is defined by make span of all the processors (i.e. completion time C of the latest job). To minimizing the make span, the elementary criterion is the time, when processor finishes the last job [6].

i.e.,

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

Individuals from the current population are selected based on their fitness and a mating pool is created for the reproduction stage.

F. SELECTION OPERATOR

The design of the fitness function is the basic of selection operation, the design of the fitness function will directly affect the performance of genetic algorithm. GAs 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. This operator generate next generation by selecting best chromosomes from parents and offspring.

G. CROSSOVER OPERATOR

Crossover operator randomly selects two parent chromosomes (chromosomes with higher values have more chance to be selected) and randomly chooses their crossover points and mates them to produce two child (offspring) chromosomes. Here, two-point crossover operator PMX (Partially Matched Crossover) is used.

In PMX crossover, all positions are found exactly once in each offspring.

H. MUTATION

It ensures that the probability of finding the optimal solution is never zero. It also acts as a safety net to recover good genetic material that may be lost through selection and crossover. An inversion mutation operator is used here in the work.

IV. COMPUTATIONAL EXPERIMENTS

In our computational study, we aim to analyze the performance of genetic algorithm in minimizing the make span of the processor. Furthermore, we also investigate the effects of varying control parameters of the GA on the performance of genetic algorithms. GA algorithm is implemented using MATLAB at command line.

We have implemented MPTS (in permutation flow shop scheduling).

Example: No of processor=4, No of jobs=15

Following table describes the time taken by the job on a particular processor.

JOB → / PROCESSOR ↓	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	6	2	6	8	9	4	7	9	1	6	3	10	5	10	7
2	10	10	10	8	9	2	9	5	7	9	6	1	6	3	1
3	6	5	2	10	3	7	1	3	1	10	5	8	7	3	5
4	10	1	10	2	10	1	4	3	7	4	3	6	5	8	9

The make span for the different feasible schedules of the jobs in a generation is computed using fitness function.

GA termination results in a schedule of the jobs with minimum make span value in the latest generation of the schedules.

In MATLAB, we have to specify crossover probability and elite count (the number of individuals with the best fitness values in the current generation that are guaranteed to survive to the next generation.). The individuals of the population are from crossover kids and elite kids, mutated kids. The no of mutated kids are counted automatically in MATLAB.

No of mutated kids= population size - (Z1+Z2) Where
Z1 = Population size - Elite count
$Z2 = (Z1*P_c)$
(Where P_c is crossover probability.)

The efficiency of genetic algorithm is closely related to control parameters. In our experiments, we have tested the performance of genetic algorithms for these parameters such as

The efficiency of genetic algorithm is closely related to control parameters. In our experiments, we have tested the performance of genetic algorithms for these parameters such as population size, the crossover probability (P_c) and the mutation probability (P_m).

Following experiments shows effect of variation of GA parameters on its performance.

A. Following table shows the effect of variation of population size on the performance of genetic algorithms in permutation flow shop scheduling

Parameter settings are:

CROSSOVER PROBABILITY	0.6
ELITE COUNT	2
NO OF GENERATIONS	100
TIME LIMIT	INFINITE
FITNESS LIMIT	INFINITE
STALL GENERATION LIMIT	INFINITE
STALL TIME LIMIT	INFINITE

Results taken are:

POP_SIZE	Make Span Run#1	Make Span Run#2	Make Span Run#3
50	108	109	107
100	105	107	105
150	104	105	105
200	105	105	105
250	105	104	105
300	104	105	104

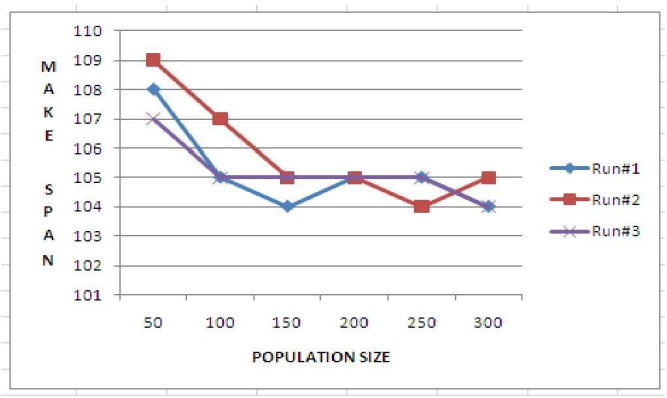

Fig 2: Effect of population size on GA Performance

Here, we can observe, for our problem MPTS (in permutation flow shop scheduling), as the population size increases, the GA performs better

(i.e. value of make span decreases). So we can say, increasing the population size enables the genetic algorithm to search more points and thereby obtain a better result.

B. Following table shows the effect of variation of crossover rate on the performance of genetic algorithms in permutation flow shop scheduling.

Parameter settings are:

POPULATION SIZE	100
ELITE COUNT	2
NO OF GENERATIONS	100
TIME LIMIT	INFINITE
FITNESS LIMIT	INFINITE
STALL GENERATION LIMIT	INFINITE
STALL TIME LIMIT	INFINITE

Results taken are:

Crossover Probability	Make Span Run#1	Make Span Run#2	Make Span Run#3
0.1	105	107	105
0.2	105	105	104
0.3	105	105	105
0.4	105	105	105
0.5	105	105	105
0.6	104	104	105
0.7	105	105	106
0.8	106	107	104
0.9	107	107	106
1	107	107	108

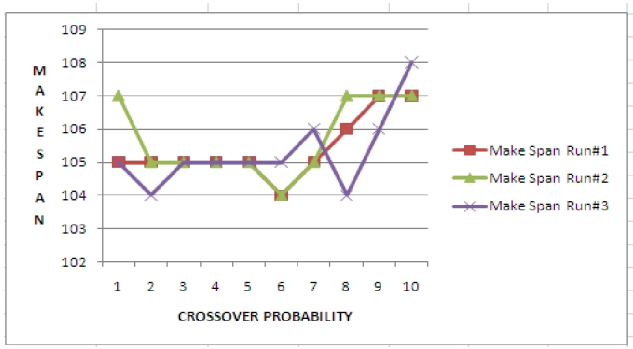

Fig 3: Effect of crossover Probability on GA Performance

The genetic algorithm minimizes the fitness function. Here, we can observe, for our problem MPTS (in permutation flow shop scheduling), as the as the crossover probability increases till 0.6, the GA performs better (i.e. value of make span decreases), but after crossover probability 0.6, the performance of GA degrades. So we can say, for this fitness function, setting Crossover fraction to 0.6 yields the best result.

C. Following table shows the effect of variation of Elite Count on the performance of genetic algorithms in permutation flow shop scheduling.

Parameter settings are:

POPULATION SIZE	100
CROSSOVER PROBABILITY	0.6
NO OF GENERATIONS	100
TIME LIMIT	INFINITE

FITNESS LIMIT	INFINITE
STALL GENERATION LIMIT	INFINITE
STALL TIME LIMIT	INFINITE

Results taken are:

Elite count	Make Span Run#1	Make Span Run#2	Make Span Run#3
2	104	104	105
3	107	105	105
4	104	105	105
5	105	108	105
6	105	104	104
7	105	105	106

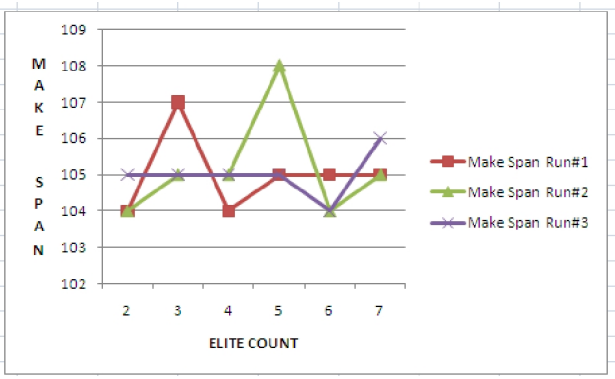

Fig 4: Effect of elite count on GA Performance

Since the genetic algorithm minimizes the fitness function. Here, for our problem MPTS (in permutation flow shop scheduling), as the value of elite count increases, the performance of GA degrades (i.e. value of make span increases). So we can say, setting Elite count to a high value makes the search less effective.

V. CONCLUSION

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 (on minimize maximum make span of the processor) with the variation of its control parameters. Here we have concluded that:

A. Increasing the population size enables the genetic algorithm to search more points and thereby obtain a better result.

B. For a fitness function, a setting for Crossover probability can yield the best result. The genetic algorithm minimizes the fitness function.

C. Setting Elite count to a high value causes the fittest individuals to dominate the population, which makes the search less effective.

VI. REFERENCES

[1] Zbigniew Michalewicz, "Genetic Algorithms + Data Structures=Evolution Programs", Springer-verlag.Practical genetic algorithms By Randy L. Haupt, S. E. Haupt

[2] MATLAB TUTORIAL by Edward Kamen and Bonnie Heck, published by Prentice Hall

[3] MATLAB by RUDRA PRATAP, tenth edition, published by Oxford University Press

[4] Michalewicz, Z. and Attia, N., "Evolutionary Optimization of Constrained Problems. In Sebald, A. and Fogel, L. (Eds.), Annual Conference on

[5] EvolutionaryProgramming, World Scientific Publishing, 1994 Kamaljit Kaur, Amit Chhabra and Gurvinder Singh., 2010, Modified Genetic Algorithm for Task Scheduling in Homogeneous Parallel System Using Heuristics proceeding of International Journal of Soft Computing Vol. 5, Issue: 2,Page No.: 42-51

[6] Javier Carretero, Fatos XhafaAjith Abraham, "GENETIC ALGORITHM BASED SCHEDULERS FOR GRID COMPUTING SYSTEMS" published in International Journal of Innovative, Computing, Information and Control, Volume 3, Number 6, December 2007.

[7] Prof. Sanjay R Sutar, Jyoti P. Sawant, Jyoti R. Jadhav , Task Scheduling For Multiprocessor Systems Using Memetic Algorithms

[8] Sachi Gupta, Gaurav Agarwal, Vikas Kumar, Task Scheduling in Multiprocessor System Using Genetic Algorithm, 2010 Second International Conference on Machine Learning and Computing

[9] EL-REWINI, H., ALI, H. H., AND LEWIS, T. G.1995. Task scheduling in multiprocessor systems. IEEE Computer

[10] ALI, S., SAIT, S. M., AND BENTEN, M. S. T. 1994.GSA: Scheduling and allocation using genetic algorithm. In Proceedings of the Conference on EURO-DAC