Optimization of Online Job Shop Partitioning and Scheduling for Heterogeneous Systems using Genetic Algorithm

Sunny Sharma[#1], *Gurjit Singh Randhawa*[#2]

[#1]*Research Scholar, Department of Computer Science, Guru Nanak Dev University, Amritsar, India*

[#2]*Research Scholar, Department of Computer Science, University of Western Ontario, Canada*

Abstract— *Job Shop Scheduling problem becomes more complex if heterogeneous systems are considered and algorithm is to be implemented for online schedulers. In real time, number of heterogeneous systems connected to online scheduler may vary from time to time. Also, number of different sized jobs may differ at any instant. This problem deals with optimization of job partitioning when maximum partition size is given; and to find out scheduling criteria when new jobs arrive keeping old jobs status in mind. So, partitioning size for any particular job and make span time for given jobs are optimized at any given instant for given set of jobs. This is known to be NP complete problem therefore many techniques based on different heuristics have been proposed to solve partitioning and scheduling problem efficiently and in reasonable amount of time. This paper proposes the solution to this problem using Genetic algorithm. Variation in number of jobs and systems require very flexible algorithm which can adjust its parameters accordingly; the proposed algorithm is capable and very efficient to handle such issues. This paper covers introduction to problem and various terms used, proposed solution using Genetic Algorithm (GA) with newly designed fitness function and performance comparison of proposed GA under various constraints.*

Categories and Subject Descriptors

D.4.1 [**Operating Systems**]: Process Management – *Multiprocessing/multiprogramming/multitasking, Scheduling.*

General Terms

Algorithms, Management, Performance, Design.

Keywords—*Genetic Algorithm, Optimization, Scheduling.*

I. Introduction

The job shop scheduling problem is one of the most well-known problems which can be applied in many fields [8]. This paper focus on specific case where some online scheduler can partition given jobs based upon given maximum partitioning size into limited number of sub tasks and then can schedule them on given online systems. This may be seen as a scenario in process scheduling for which unbounded numbers of heterogeneous systems are considered and different sized jobs arrive at online scheduler after random time intervals. Number of jobs at any time interval may also vary. Unbounded number of heterogeneous systems, variable job sizes, variable number of jobs and variable partitioning size make this problem much more complex and also identical to real life scenario. Though this paper proposed a solution keeping specific scenario of process scheduling with mentioned constraints, this solution can also be applied with little modifications on job shop problems of different nature.

Job shop scheduling problem is regarded as one of the most challenging NP-hard problem. Efficient methods are important for increasing production efficiency, reducing cost and improving product quality [10]. Number of algorithms has been developed to address this task. Some examples include the Giffler and Thompson algorithm, the shifting bottleneck algorithm, Tabu search (TS) [1], [5], simulated annealing (SA) and genetic algorithm (GA) [2], [3] etc. Most of them fail to obtain good solutions solving large scale problems because of the huge memory and lengthy computational time required [14]. On the other hand, heuristic methods include dispatching priority rules, shifting bottleneck etc. are popular alternatives for such problems. Due to some limitations of these techniques and emergence of some new techniques, much attention has been devoted to meta-heuristics. One main class of meta-heuristic is the population based heuristic e.g. Genetic algorithm (GA) [13], particle swarm optimization (PSO), and so on. Among the above methods, GA, proposed by John Holland, is regarded as problem independent approach and is well suited to deal with such hard problems [6], [9].

II. Job Shop Scheduling Problem

Job shop scheduling problem is a specific and well known class of scheduling problems. The problem and various constraints considered are mentioned in this section of the paper. Specific scenario of process scheduling is considered to implement solution for this class of Job shop problem. At any instant of time, n jobs may arrive at online scheduler each with specified size s. Time interval between arrivals of new jobs is taken as random, so any variable n number of jobs may join the queue at any instant of time. Since solution is implemented keeping constraints of process scheduling in mind, so size is considered in MI (million instructions) per job. Each job also specifies maximum partition number. If maximum partition size is given as zero, then that job can't be partitioned further into sub-jobs/sub-tasks (in implemented scenario of problem). The m numbers of heterogeneous systems are available at any particular instance. Since this implementation is designed for online scheduler which is assumed to work in real time scenario, so value of m may vary. As some systems may go offline and some new systems may join the network. Each system has different computational capability/ processing power which is defined as MIPS (million instructions per second). Mathematical representation of these constraints and objective functions is discussed under design methodology.

III. GENETIC ALGORITHM

Genetic algorithms are search and optimization algorithms based on the mechanics of natural selection and natural genetics [David E. Goldberg]. Here search refers to search for optimal solution in given solution space. These algorithms are useful in solving optimization problems by emulating biological theories [11]. Based on Darwin's theory of evolution, they work for survival of fittest [12]. Genetic algorithms can automatically access the search space and adaptively adjust the search direction to improve solution because they work on probabilistic rules rather than deterministic approach [4]. Most of genetic operators also work in accordance with random approach. The working of basic genetic algorithm is shown in Figure 1. Different components of basic genetic algorithms are explained as follows:

3.1 Population

Population consists of collection of possible solutions (referred as Chromosomes). Initial population is made up of randomly chosen Chromosomes from given search space. Genetic algorithms work in improving average fitness of population from generation to generation keeping in mind objective function of given problem.

3.2 Chromosome

Each individual element of any population is called Chromosome. Chromosome is any feasible solution available in search space represented in structured manner. Representation is problem dependent and can be done in many ways such as these can be encoded in form of Binary String, Integer Strings, Real Strings, and Hybrid Strings etc.

3.3 Fitness Function

Keeping objective function in mind, fitness functions are designed in such manner that strength of each Chromosome can be evaluated using these functions. So, fitness functions provide a quantitative base to check position of current chromosome in given search space and to decide right direction towards optimal solution to make improvements at each and every step.

3.4 Selection

From given population of chromosomes (treated as mating pool), each candidate receives reproduction probability based upon fitness value of its own and fitness value of other chromosomes. This reproduction probability decides whether any chromosome will be selected as parent for mating or not. Any selection criteria can be used based upon problem given such as Roulette Wheel Selection, Rank Selection, and Tournament Selection etc.

3.5 Crossover

This genetic operator operates on every pair of parents selected based upon crossover probability. Child Strings are generated from parent Strings by exchanging information among strings of mating pool. Single Point or Multi Point Crossover can be applied depending upon given problem. Same type of crossover should be used throughout the run to main consistency in approaching optimal solution.

3.6 Mutation

Each chromosome goes under mutation after performing crossover with very low mutation probability. Main objective of using mutation is to select neighbouring point instead of current point in given search space. Sudden alteration is made in few genes of any chromosome under this operator.

3.7 Elitism

Preservation of best fit chromosomes in new generations by replacing newly found chromosomes with lesser fitness value is performed [7].

3.8 Termination Criteria

Termination Criteria could be predefined in form of number of generations to be produced or it could be left for later. In second case, when there is no improvement in overall average for few generations, then decision for termination is made.

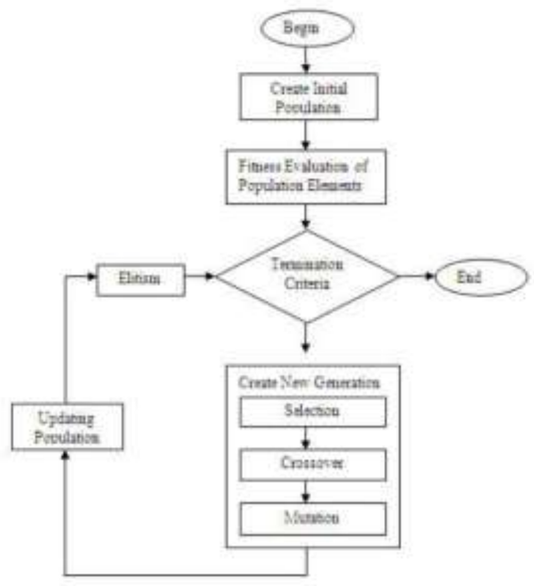

Fig. 1 Flowchart for basic Genetic Algorithm

IV. DESIGN METHODOLOGY

Solution to this problem is proposed and implemented based upon Multi-objective functions. GA works twice. First algorithm tries to optimize partitioning size of given jobs. Second one look for optimization of scheduling criteria. If jobs are partitioned into sub-tasks, then different tasks can be allocated to different systems. So, scheduling is not only based upon given jobs, but it is based upon sub-tasks. When new jobs arrive at next time instant, algorithm keeps allocation of uncompleted jobs fixed. Elitism is applied after each iteration.

4.1 Partitioning Algorithm

Partitioning algorithm is genetic algorithm which tries to optimize number of partitions for each job bounded by given maximum possible partition size. This is not wise approach to evaluate fitness of the solution every time based upon possible scheduling criteria. Moreover, scheduling criteria can't be accurately found till optimized partitioning number for each job is available. So, fitness function is based upon how the load will be divided among available systems. Stress is given to partition in such a way so that more tasks can be executed on systems with higher processing ability. Number of generations for this algorithm is taken as 10. So, whole procedure is repeated 10 times. Different parameters settings are explained as follows:

4.1.1 Encoding Chromosome

Each random chromosome is encoded with integers. Each chromosome for particular instance has same length which is equal to n- number of given jobs. Suppose at one particular instance 6 jobs arrive, so chromosome length will be 6. Value of any particular bit is between 0 and maximum partitioning number which is given for that particular job. Suppose first three jobs have maximum partitioning number 2 and rest have maximum partitioning number 3. Then any random chromosome will look as follows:
Sample Chromosome 1: 201231
Sample Chromosome 1: 212203

4.1.2 Population

Population size is fixed as 10. So, for initial population 10 random chromosomes are generated with above specified constraints.

4.1.3 Fitness Function

Status for any system at start up is initialized to 1, which represent its availability. It is reset to 0, when it goes busy from idle. Penalty of 100 is added every time if system is not available to reduce probability of survival of respective chromosome. Mathematical representation of Fitness function for optimization of job partitioning can be given as:

$$Fitness_c = Max\,\mathrm{S_i} \quad \mathrm{where}\, 0 \le \mathrm{i} \le \mathrm{m}$$

$$\mathrm{S} = \sum_{i=1}^{n} \left(\sum_{j=1}^{m} \left(\begin{array}{l} if\,(S_i\,.status = 1), then \\ S_i \;= S_i \;+ n.size / C_i \\ else \\ S_i \;+ Penalty \end{array} \right) \right)$$

4.1.4 Selection

Once the population of chromosomes has been generated, the selection of parents has been done based upon a respective fitness function. The Roulette wheel selection method or the fitness proportionate selection method has been used for this problem. Under this method, chromosome with higher fitness has higher chances of being selected as parent. Five pairs of parents are selected for each iteration from one current population to form new population.

4.1.5 Crossover Operator

Single point crossover is adopted for this problem since the string length is not very large. Crossover is applied with some probability on each pair of selected parents. The crossover probability for this problem is fixed at 0.8.

4.1.6 Mutation Operator

Mutation alters one or more gene values in a chromosome from its initial state. For each bit value, alteration is done which is bounded between 0 and maximum possible partition size given for that respective job. The mutation probability for this problem is set to be 0.6. Mutation is performed with mentioned probability on each bit of every chromosome in mating pool.

4.2 Scheduling Algorithm

Once partition number for each job is optimized by partitioning algorithm, then comes the scheduling of partitioned jobs. Any job i can be partitioned into t tasks. t is optimized partition number found by partitioning algorithm and is bounded between 0 and maximum possible partitioning number given for that particular job i. Number of generations for this algorithm is taken as 20. So, whole procedure is repeated 20 times.

4.2.1 Encoding chromosome

Chromosome is encoded with integer values. Length of chromosome is sum of optimized number of partitions for all jobs. In mathematical representation this length is T. Each bit value is any possible integer value in range of 0 to number of m, where m is number of heterogeneous systems available at respective time instant. 0 denote that respective task is unallocated for which fitness function will add some penalty to make span time. As, after number of generations, algorithm automatically tries to improve fitness of every individual chromosome, all zeros are most likely to be eliminated from final solution with some feasible integer value.

4.2.2 Population

Population size is fixed as 10. So, for initial population 10 random chromosomes are generated with above specified constraints.

4.2.3 Fitness Function

Make span time is used to evaluate the fitness of every individual chromosome. For any bit value 0 i.e. for any unallocated task, penalty of 100 is added to fitness value. This reduces the total fitness value and probability of survival for such chromosomes automatically reduces. If few jobs are uncompleted when new jobs arrive, then allocation of previous uncompleted jobs is kept fixed. If new jobs are allocated to same system then waiting time is added. Respective Fitness function to evaluate fitness of chromosomes representing scheduling criteria can be represented as:

$$Fitness_c = Max\ \mathrm{S_i}\ \ \mathrm{where}\, 0 \le \mathrm{i} \le \mathrm{m}$$

$$S = \sum_{i=1}^{\mathrm{m}} \left(\sum_{j=1}^{T} if\,(C_j = S_i\,), then\, \mathrm{S_i} = S_i + j.Size\,/\,m.cap \right)$$

$$Penalty = \sum_{j=1}^{T} if\,(C_j = 0), then\ \mathrm{Penalty} = \mathrm{Penalty} + 100$$

$where,\ \mathrm{c} =$ Chromosome under processing

$\mathrm{m} =$ Number of systems

$\mathrm{T} =$ Sum of optimizd partition number t for every job

j.size = Size in MI for any job j

m.cap= Computation ability in MIPS of any systemm

Table I shows definition of various parameters used in showing sample results. Results for one sample iteration using above fitness function are shown in Table II. It can be seen that average fitness value is comparatively reduced. Actually, as this is minimization problem, so lesser value is the better. For Selection process, a large normalization constant is used to convert this minimization problem into maximization problem and then vice versa.

Table I: Notations used for Sample Result

Symbol	Definition
SL	Schedule Length to be minimized
FV	Fitness Values which corresponds Schedule Length
NSL	Normalization Constant(Taken to be 500) – Schedule Length
f_i	Fitness of a string (chromosome)
Σf	Sum of fitness values
CSelect	f_i / Σf
F	Average of fitness values
ECount	Expected Count = f_i / f
RW	Actual Count From Roulette Wheel (By Rounding off ECount)
C. Site	Crossover Site
M.S	Mates Selected randomly

Table II: Sample Result for single iteration

Initial Population	FV	NSL	CSelect	ECount	RW	M.S	C.Site	New Population	FV
7141741	87	113	0.16	0.96	1	1	3	7242466	80
1222776	100	100	0.141	0.84	1	3	3	1322743	62
1324446	70	130	0.183	1.10	1	2	5	1262231	97
1332557	109	91	0.129	0.77	1	6	5	7737616	64
2255331	55	145	0.20	1.23	1	4	4	1232375	87
7777666	70	130	0.183	1.10	1	5	4	2255551	50
Sum	491	709							440
Average		118							
Max		145							

4.2.4 Selection

Once the population of chromosomes has been generated, the selection of parents has been done based upon a respective fitness function. The Roulette wheel selection method or the fitness proportionate selection method has been used for this problem. Under this method, chromosome with higher fitness has higher chances of being selected as parent. Five pairs of parents are selected for each iteration from one current population to form new population.

4.2.5 Crossover Operator

Single point crossover is adopted for this problem. Crossover is applied with some probability on each pair of selected parents. The crossover probability for this problem is kept fixed at 0.8. Figure 2 shows sample of crossover operation done at locus 3.

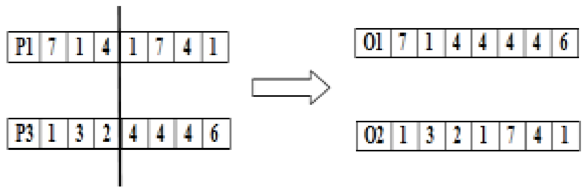

Fig. 2 Crossover Operation

4.2.6 Mutation Operator

Mutation alters one or more gene values in a chromosome from its initial state. For each bit value, alteration is done which is bounded between 1 and m, where m is number of heterogeneous systems available at respective time instant. The mutation probability for this problem is set to be 0.6. Mutation is performed with mentioned probability on each bit of every chromosome in mating pool. Figure 3 shows sample of mutation operation

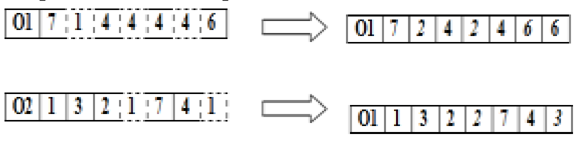

Fig. 3 Mutation Operation

V. PERFORMANCE COMPARISON

The performance of the system with time i.e. the changes in crossover and mutation probability with time and the effect of probability of crossover and mutation on the number of generations has been studied. The crossover probability is analysed, by keeping the population size, number of generations and the mutation probability fixed at a certain value. The Figure 3 shows that the probability of crossover, pc, gives better results at a value pc = 0.6 to pc = 0.8. Another analysis is also done with combination of both crossover and mutation probabilities. Hence for this problem, a crossover probability of 0.8 has been chosen for better results.

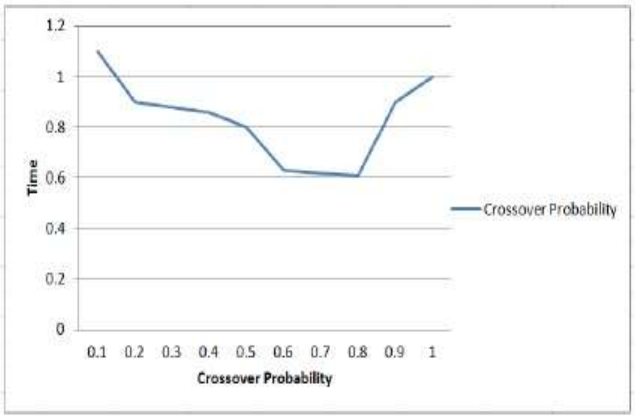

Fig. 4 Crossover Probability Comparison Graph

The probability of mutation has been analysed by keeping the size of the population, the number of generations and the crossover probability fixed at a certain value. The analysis graphical representation as in Figure 4 shows that, the probability of mutation, pm, gives optimal results at a value pm = 0.4 to pm = 0.6. Thus for this problem, the mutation probability is chosen to be 0.6.

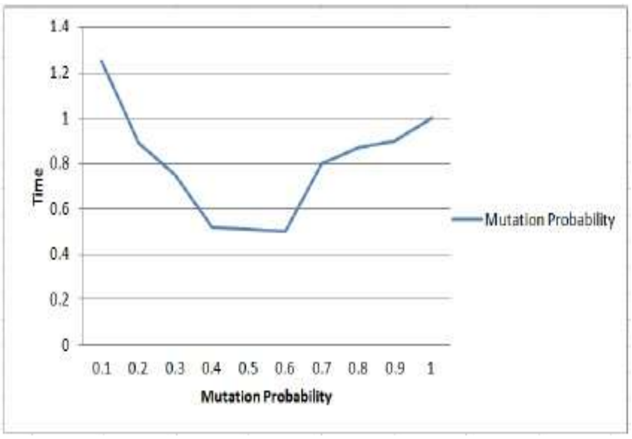

Fig. 4 Mutation Probability Comparison Graph

The effect on the number of generations has been analysed with respect to the probability of crossover and mutation, by keeping the population size fixed and varying the probability of crossover and mutation from 0.1 to 1. The Figure 5 shows that the near optimal results are achieved at crossover probability, pc= 0.8 and mutation probability, pm= 0.6.

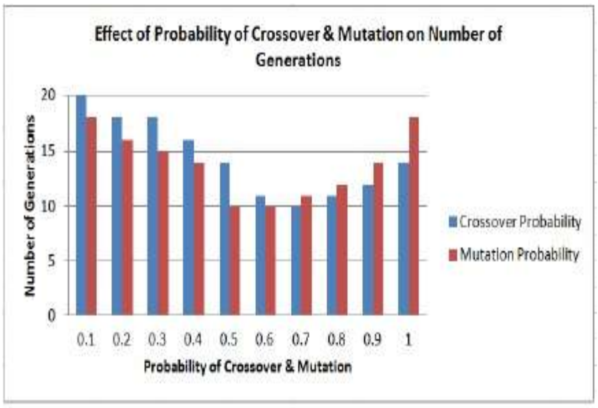

Fig. 6 Effect on number of generations

VI. CONCLUSIONS

Job Shop scheduling problem is NP- Hard problem for which many algorithms under different categories have been proposed before. This paper proposed a solution to this multi objective problem to optimize both job partition size and scheduling criteria. Solution is implemented under meta heuristic category of algorithms using Genetic Algorithm. Apart for traditional binary string representation of genetic algorithms, the chromosomes are modified to be integer strings representation. New fitness functions are also proposed. Results show that final outcome of the algorithm is near optimal all the time, but the hardness of problem is resolved in the sense that computation time is reduced by big margin. This approach is very practical and helpful especially for real time online schedulers which can't spend much time to find optimal scheduling criteria and keep arriving jobs in waiting queue for long time.

ACKNOWLEDGMENTS

Our thanks to Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar, India for providing us all the required resources necessary to study and implement the solution to this problem. We also want to thank all the authors of referenced papers which guided us all the time as base for this study.

REFERENCES

[1] A. M. Dell, M. Trubian, "Applying tabu-search to job shop scheduling problem," Annals of Operations Research, vol. 41, no. 3, pp. 231-252, 1993.

[2] Albert Y.Zomaya, Chris Ward and Ben Macey, "Genetic Scheduling for Parallel Processor Systems: Comparative Studies and Performance Issues", IEEE Transactions on Parallel and Distributed systems, Vol. 10, No.8, pp.795-812,August 1999.

[3] Amir Masoud Rahmani and Mojtaba Rezvani, "A Novel Genetic Algorithm for Static Task Scheduling in Distributed Systems", International Journal of Computer Theory and Engineering, Vol. 1, No. 1, 1793-8201, April 2009

[4] David. E. Goldberg, "Genetic algorithms in Search, Optimization & Machine learning", Addison Wesley, Publishing Co. Inc. ,Boston, MA, pp- 1-25, 1990.

[5] E. Nowicki, C. Smutnicki, "A fast taboo search algorithm for the job shop scheduling problem", Management Science, vol. 41, no. 6, pp. 113-125, 1996.

[6] Edwin.S.H Hou, N.Ansari, H.Ren , "A Genetic Algorithm for Multiprocessor Scheduling", IEEE Transaction on Parallel and Distributed Systems, vol. 5,no. 2, pp.113-120,Feb. 1994.

[7] Erick Cantú-Paz, "A Survey of Parallel Genetic Algorithms", Department of Computer Science and Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign,1998.

[8] Gerasoulis and Yang, "DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors.", IEEE Transactions on Parallel and Distributed Systems, vol. 5, no. 9, pp. 951-967, Sep. 1994.

[9] Imad Fakhri Al Shaikhli and Ismail Khalil , "An Improved Genetic Algorithm for Solving The Multiprocessor Scheduling Problem", Australian Journal of Basic and Applied Sciences, ISSN 1991-8178, Vol.5, No.12, pp : 947-951, 2011.

[10] Liang Sun, Xiaochun Cheng, Yanchun Liang, "Solving Job Shop Scheduling problem Using Genetic Algorithm with Penalty Function", International Journal of Intelligent Information Processing, Volume 1, Number 2, December 2010.

[11] Melanie Mitchell, "An Introduction to Genetic algorithms", The MIT press, Cambridge, Massachusetts, England, 5th printing, pp 2-20, 1999.

[12] Pratibha Bajpai et al., "Genetic Algorithm- an Approach to Solve Global Optimization Problems", Indian Journal of Computer Science and Engineering, 2010.

[13] Yi-Hsuan Lee and Cheng Chen, "A Modified Genetic Algorithm for Task Scheduling in Multiprocessor Systems", Proceedings of 6th International Conference Systems and Applications, IEEE Computer Society, Washington DC,USA, pp. 382-387, 1999.

[14] Wei Wu, Junhu Wei and Xiaohong Guan, "Hybrid Nested Partitions Algorithm for scheduling in job shop problem", "Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics", December 19-23, 2009, Guilin, China.

[15] Sharma MS, Virk RS. A Review towards Evolutionary Multiobjective optimization Algorithms, http://ijoes.vidyapublications.com/paper /Vol13/37-Vol13.pdf.