SOLVING JOB SHOP SCHEDULING PROBLEM WITH GENETIC ALGORITHM

Saleha Noor[1], M. IkramUllah Lali[1], M. Saqib Nawaz[2*]
[1]Department of Computer Science & IT, University of Sargodha, Sargodha, Pakistan
[2]Department of Computer Science, The University of Lahore, Sargodha Campus, Pakistan
[*]saqib_dola@yahoo.com

ABSTRACT: *Genetic algorithms (GAs) are search algorithms that are used to solve optimization problems in theoretical computer science. Job shop scheduling (JSS) problem is a combinatorial optimization problem where main goal is to find a schedule with minimum makespan for processing of j jobs on a set of m machines. The schedule for jobs processing in JSS problem is subjected to some constraints, such as only one operation of a job can be processed on one machine at a time, operations of a job has to be processed in a certain order and preemption of any operation is forbidden. In this article, we have suggested a new genetic algorithm (GA) to solve the JSS problem which uses a new genetic representation (coding) for scheduling of jobs and machine distribution. Followed by genetic representation, an initial population is randomly generated. Then crossover and mutation operator of GA are applied on the population for the creation of new off-springs until some stopping criterion is reached. Furthermore, experiments are performed to show the performance and applicability of proposed GA.*

Keywords: Crossover Operator; Genetic Algorithms; Job Shop Scheduling; Mutation Operator; Makespan.

1. INTRODUCTION

Scheduling of operations is considered the most critical issues in the manufacturing processes management and planning. Scheduling is concerned with jobs assignment to production resources and specifying the sequence in order to optimize certain objective functions. In manufacturing environment, scheduling depends on the environment of the shop floor such as job-shop, flow-shop and open-shop [1]. Job Shop Scheduling (JSS) problem can be explained as: a job set *j* contains *n* jobs and these jobs are processed on one of the machine selected from machine set *m*. Each job is comprised of operations O = O_1, O_2, ..., O_n. Each machine can process one operation of a job at a time and when an operation is assigned to a machine, the machine must complete processing of operation without any interruption. Furthermore, operations of the job have to be processed in a certain given order. The problem is to find the schedule for operations on machines, keeping in mind the precedence constraints that will minimize the makespan. Makespan (C_{max}) is the time in which all operations of each job are completed.

Job-shop scheduling is a well-known and strongly NP-hard problem [2] and has also proven to be computationally challenging. JSS problem analysis provides relevant insight into the solution of the scheduling problems that are faced in more complicated and realistic systems. Therefore, heuristics are preferred for job shop scheduling [3]. GA is considered the best-known optimizing technique for a class of combinatorial problems [4, 5].

Genetic algorithms (GAs) introduced by Holland [6], are search algorithms based on evolutionary process. GAs are well-known to solve optimization problems in theoretical computer science. In GA, if the solution in the population is suitably encoded and by mimicking the process of GA, it has the ability to evolve solutions to the problems of real world [7]. Analysis of GAs have begun years ago to understand the working of genetic algorithms and how to use them to get the best solution [8]. In applying GA to a problem, a suitable representation (encoding) for the given problem must be designed. A fitness function is also required which is used to select individuals solutions from population. During run, selected solutions go through crossover (reproduction) and mutation operation to generate new off-spring. Main idea here is that off-springs will perform better than their corresponding parents. First GA based technique for scheduling problem was proposed by Davis [9] in 1985. After that GAs have been frequently used to solve the scheduling problems. In this study we have presented a GA to solve the JSS problem.

The rest of the article is divided in three main sections. In Section 2, literature review in the area of GA and JSS problem is discussed. In Section 3, proposed GA is presented by detailing the strategies of encoding scheme, the fitness function, the criteria for selection, crossover and mutation operators adopted for the generation of off-springs. Experiments are performed in Section 4 to show the performance of proposed GA and conclusions are drawn in Section 5.

2. LITERATURE REVIEW

In literature, various GA approaches have been used to solve the JSS problem. These approaches are different from one another on the basis of representation scheme and GA that are used, the constraints handling and the pursued goals. However, all approaches have one common thing: knowledge is required in order to generate the competitive schedules for processing of jobs. In 1985, Davis [9] first used the GA to solve the JSS problem. Jain and Meeran [10] provide a comprehensive survey of methods which are used to address JSS problem. They concluded that among various techniques to address JSS problem, meta-heuristic technique is best for JSS problem. The combined use of heuristic and a GA is called as meta-heuristics. With the growth in the popularity of GA in the mid 1980's, different researchers used GA to solve the JSS problem.

A two-row chromosome structure is defined by Li and Chen [11] that is based on the working order of procedures and machine distribution to minimize the makespan in JSS problem. Lee et al. [12] proposed a framework that utilizes GA along with machine learning and heuristics space to find the sequence of job release order and the sequence for dispatching jobs at each individual machine. According to Onwubolu and Davendra [13], GA is the best technique that is used to optimize JSS problem. Whereas some researchers, Bierwirth et al. [14], Meeran and Morshed [15] have opinion that hybrid GA performance to optimize JSS problem is far better than standard GA.

To address job shop problem, numbers of techniques other than GAs are also used in practice such as enumerative technique, approximation based approach and artificial neural networks. Williamson et al. [16] demonstrate that deciding the presence of a schedule with a makespan of three could be carried out in polynomial time as long as the aggregate preparing time needed by all the operations on each one machine is close to three. The fundamental centering of enumerative methodologies for the job shop is climb and bound techniques. The two most regular branching procedures are Generating Active Schedules (GAS) and Settling Essential Conicts (SEC) proposed by Lageweg et al. [17]. For practical application, standard GA may not be flexible enough and this becomes increasingly apparent when problem is complicated. Furthermore, Uckun et al. [18] concluded that GAs rapidly inclined towards possible solutions.

3. PROPOSED GENETIC ALGORITHM

The overall structure of our proposed GA is described in following steps:

1. Representation: Each chromosome (solution) in the population is represented in a format on which further GA operators will be applied;
2. Selection: Selection of solutions from the population for reproduction;
3. Off-spring generation: New solutions are obtained by changing the operations sequence (order crossover) and by changing the assignment of machines to the operations (uniform crossover);
4. Mutation: Randomly flips some bits in a schedule;
5. Fitness function: Makespan of each schedule obtained after step 4 is computed;
6. Termination criterion: Fixed number of generations is reached. When termination criterion is reached, GA returns the best schedule along with the corresponding makespan as output.

The above six steps that we have used to solve the JSS problem are explained next.

3.1. Encoding Scheme

Encoding is a crucial and vital element of GA. Finding optimal solution for a problem using GA depends greatly on encoding scheme. GA generally process populations of strings. Initial population is randomly generated. Each chromosome represents a solution (schedule) for the JSS problem. Assignments of jobs operations to the machines in the schedules are described by the genes of the chromosomes and the order of operation that appears in the chromosome represents operations sequence. In our proposed GA, we have used three row structure to represent candidate chromosome (schedule). Figure 1 shows three row structure for a 4×4 JSS problem.

$O_{31}O_{21}O_{32}O_{41}O_{22}O_{11}O_{33}O_{42}O_{23}O_{34}O_{12}O_{43}O_{13}O_{44}O_{14}O_{24}$

3 1 2 3 3 1 1 4 4 4 3 2 2 4 4 2

0 0 O_{31} *0* O_{21} *0* $O_{32}O_{41}O_{22}O_{33}O_{11}O_{42}O_{12}O_{43}O_{13}O_{23}$

Figure 1. Encoding scheme

In Figure 1, first row represents the operations of jobs. For example O_{31} implies operation 1 of Job3 (J_3), O_{21} implies operation 1 of Job2 (J_2), O_{33} implies operation 3 of Job3 (J_3), O_{43} implies operation 3 of Job4 (J_4) and so on. Second row represents machine distribution for each operation. 3 implies that operation O_{31} will go to machine number 3 for processing, 1 implies that machine 1 is assigned to operation O_{21} and so on. Third row shows operation dependencies of jobs operations. Here, *0* implies that for O_{31} to be executed there is no precedence restriction exits. Same is also the case for first operation of other jobs. O_{31} implies that processing of operation O_{32} will start when operation O_{31} is processed.

3.2. Parent Selection

After the representation of schedule, next step is the selection of schedules from the population. Selected schedules are called parent schedules. Different techniques are used for the selection of parent schedules. In order to avoid the premature convergence, we randomly choose two schedules from the population. This means that all the individuals (schedules) in the population have the same chances of selection as parents.

3.3. Crossover Operator

Crossover operator of GA is used to exchange the operations of jobs and assignment of particular operation to a machine in two selected parent in order to minimize the makespan. In our proposed algorithm, we have used order crossover for operations shuffling between two parent schedules. Whereas, uniform crossover operator is used for shuffling of machines assignment to operations.

3.3.1. Order Crossover

We have used order crossover [19] to breed child schedules from parent's schedules. We have put 2 restrictions on order crossover operator to meet precedence constraints. These restrictions are following:

- Crossover points must be greater than $\frac{J\times O}{2}$, where *J* represents total number of jobs and *O* represents total number of operations.
- No consecutive execution of operations of same job.

We have explained order crossover with a simple example. Let two selected parent schedules are:

P_1:$O_{31}O_{21}O_{32}O_{41}O_{22}O_{11}O_{33}O_{42}O_{23}O_{34}O_{12}O_{43}O_{13}O_{44}O_{14}O_{24}$

P_2: $O_{11}O_{21}O_{12}O_{41}O_{22}O_{31}O_{13}O_{42}O_{23}O_{14}O_{32}O_{43}O_{33}O_{44}O_{34}O_{24}$

Select any two random crossover points *j* and *k* for instance (*j* = 13 and *k* = 16). The substring s(P_1), obtained from the parent P_1, contain those operations of P_1 that occupy the positions between *j* and *k*, s(P_1) = $O_{13}O_{44}O_{14}O_{24}$. We now delete in P_2 the operations s(P_1), obtaining:

P_2\` = $O_{11}O_{21}O_{12}O_{41}O_{22}O_{31}O_{42}O_{23}O_{32}O_{43}O_{33}O_{34}$.

Finally, we insert in P_2\` the substring s(P_1) resulting in the new schedule:

$O_{11}O_{21}O_{12}O_{41}O_{22}\ O_{31}O_{42}O_{23}O_{32}O_{43}O_{33}O_{34}O_{13}O_{44}O_{14}O_{24}$

Thus, after the order crossover, we obtain a schedule which has between *j* and *k* the corresponding operations from P_1 and in the other positions, the operations of list P_2\` which is obtained as the difference between the elements of P_2 and s(P_1). The resultant off-springs after order crossover operation are:

OS₁: $O_{31}O_{21}O_{32}O_{41}O_{22}O_{11}O_{42}O_{23}O_{12}O_{13}O_{14}O_{43}O_{33}O_{44}O_{34}O_{24}$

OS₂: $O_{11}O_{21}O_{12}O_{41}O_{22}O_{31}O_{42}O_{23}O_{32}O_{33}O_{34}O_{43}O_{13}O_{44}O_{14}O_{24}$

Resultant off-spring schedules show that precedence constraints for operations are satisfied. Every operation is being executed after the execution of its dependent operation.

3.3.2. Uniform Crossover

In JSS problem, machine assignment to a particular operation of a job is known in advance. We have used uniform crossover to change machine assignment among operations of jobs in order to minimize the makespan. In uniform crossover, when same operation of a job in two schedules has different machines assignment, then shuffle the machine number among them. Working of uniform crossover for machine shuffling is explained with following example. Let two schedules obtained after order crossover operation are:

S_1: $O_{31}O_{21}O_{32}O_{41}O_{22}O_{11}O_{33}O_{42}O_{23}O_{34}O_{12}O_{43}O_{13}O_{44}O_{14}O_{24}$
M: 2 1 3 4 3 4 1 4 3 4 1 3 1 2 2 1
S_2: $O_{11}O_{21}O_{12}O_{41}O_{22}O_{31}O_{13}O_{42}O_{23}O_{14}O_{32}O_{43}O_{33}O_{44}O_{34}O_{24}$
M: 4 4 3 3 3 2 2 2 2 1 3 1 1 4 4 1

After uniform crossover, new machine assignment to each operation in schedules S_1 and S_2 will be:

S_1: $O_{31}O_{21}O_{32}O_{41}O_{22}O_{11}O_{33}O_{42}O_{23}O_{34}O_{12}O_{43}O_{13}O_{44}O_{14}O_{24}$
M: 2 4 3 3 3 4 1 2 2 4 3 1 2 4 1 1
S_2: $O_{11}O_{21}O_{12}O_{41}O_{22}O_{31}O_{13}O_{42}O_{23}O_{14}O_{32}O_{43}O_{33}O_{44}O_{34}O_{24}$
M: 4 1 1 4 3 2 1 4 3 2 3 3 1 2 4 1

3.4. Mutation Operator

Lee et al. [20] used precedence preserving shift mutation (PPS) operator in their proposed genetic algorithm. We have also used PPS operator with little modification. Following steps show how our modified PPS operator works and how schedules obtained after uniform crossover operator can be mutated.

- Select any position *i* from the schedule.
- Find the position (*po*) of the predecessor operation of operation selected in step 1.
- Select a position *x* in range of selected position *i* and *po*.
- Move operation at position *i* to position *x*.
- Move operation already at position *x* and onward operations one position ahead.

Suppose operation at position 16th is selected as shifting operation. Its *po* is 9th position operation.

po *x* *i*

$O_{31}O_{21}O_{32}O_{41}O_{22}O_{11}O_{33}O_{42}O_{23}O_{34}O_{12}O_{43}O_{13}O_{44}O_{14}O_{24}$

New schedule obtained after modified PSS operator is:

$O_{31}O_{21}O_{32}O_{41}O_{22}O_{11}O_{33}O_{42}O_{23}O_{34}O_{12}O_{24}O_{43}O_{13}O_{44}O_{14}$

3.5. Fitness Function

Main objective of JSS problem is to find a schedule with lowest makespan value. The makespan can be calculated from the information of the sequence of operation processing on each machine, and the operation processing starting times. We have selected makespan as the fitness function. Since we want to find schedules which have lower values of the makespan, therefore, the genetic evolution will prefer schedules with lower fitness values. During each generation, all the selected schedule are evaluated, and the best individual is noted.

4. RESULT AND DISCUSSIONS

We have applied the proposed GA to a 4×4 JSS problem to inspect the applicability and effectiveness of adopted methodology. Operations processing sequence of each job and processing time for each operation is given in Table 1.

Table 1. Operations assignment to machines and time required for execution of each operation.

	Machine1	Machine2	Machine2	Machine4
Job 1	O_{14}(3)	O_{21}(5.5)	O_{34}(2)	O_{41}(8.5)
Job 2	O_{13}(3.5)	O_{22}(5)	O_{31}(7)	O_{42}(8)
Job 3	O_{11}(6)	O_{23}(4)	O_{32}(2.5)	O_{43}(2.5)
Job 4	O_{12}(6.5)	O_{24}(4.5)	O_{33}(7.5)	O_{44}(1)

Initial population that we considered is 50 and we run the GA 30 times under the condition that the crossover rate is 0.8 and the mutation rate is 0.2. Result in Table 2 shows that our proposed GA provides a result as good as other methods. It is also clear from Table 2 that the last job processed is O_{44} on machine 4. Therefore, makespan value for this JSS problem is 26.5.

The result in Table 2 also provides information about the sequence of job for each machine, starting and the finish time for each operation. For example, on machine 1, O_{21} is started at time 0 and finished at 5.5. Then operation O_{41} is started at time 5.5 and ended at time 14 and so on. The Gantt chart of processing route is shown in Figure 2. Obtained makespan after each generation is shown in Figure 3. From Figure 3, it is clear that the optimum value of makespan is found at generation 12.

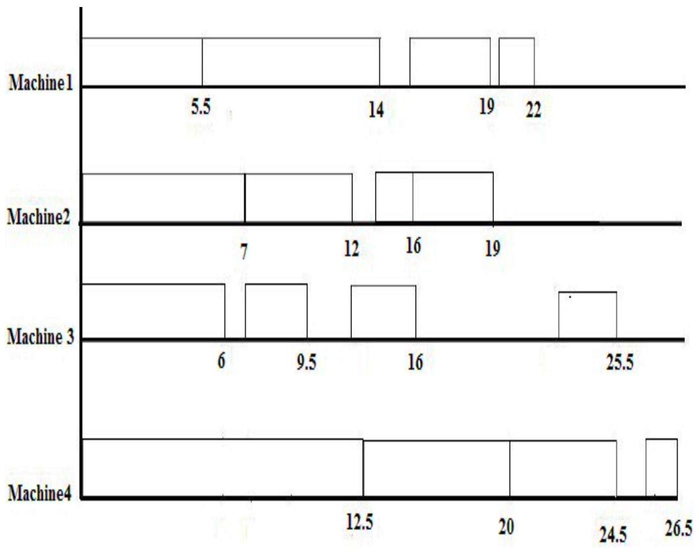

Figure 2. Gantt chart

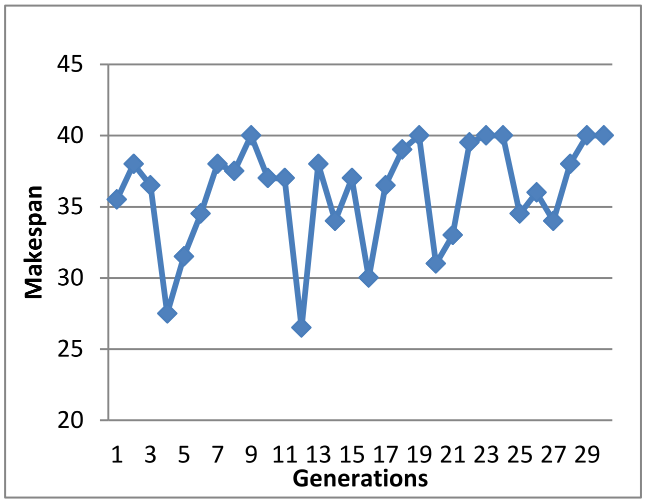

Figure 3. Result obtained using the randomly generated schedules as initial population

Table 2. Obtained machine assignment to operations and start and end time of each operation

Machine1	**Processing Time**	**Start Time**	**End Time**
O_{21}	5.5	0	5.5
O_{41}	8.5	5.5	14
O_{14}	3	16	19
O_{34}	2	20	22
Machine2	**Processing Time**	**Start Time**	**End Time**
O_{22}	6.5	5.5	12
O_{31}	7	0	7
O_{42}	8	14	24
O_{13}	3.5	12.5	16
Machine3	**Processing Time**	**Start Time**	**End Time**
O_{11}	6	0	6
O_{32}	2.5	7	9.5
O_{23}	4	12	16
O_{43}	1.5	24	25.5
Machine4	**Processing Time**	**Start Time**	**End Time**
O_{12}	6.5	6	12.5
O_{33}	7.5	9.5	20
O_{24}	4.5	16	24.5
O_{44}	1	25.5	26.5

5. CONCLUSION

This article presented a genetic algorithm (GA) for the job shop scheduling (JSS) problem. Computational results show that the proposed GA is efficient and effective. GA and JSS problem provide a framework for evolutionary computation and an insight into the combinatorial optimization problems. Although GA has the weakness that it takes a lot of time in order to find the better solution yet it offers a flexible framework for evolutionary computation. In future, we would like to see how proposed GA will be applied on a larger size problem in order to see its performance on big problems.

REFERENCES

1. Phanden, R. K., Jain, A. and Verma, R., "A genetic algorithm based approach for job shop scheduling." *Journal of Manufacturing Technology*, **23**(7): 937-946 (2012).
2. Garey, M. R., Johnson, D. S. and Sethi R., "The complexity of flow shop and job shop scheduling." *Mathematical Operational Research*, **1**(2): 117–129 (1976).
3. Maccarthy, B. L. and Liu, J., "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling." *International Journal of Production Research*, **31**(1): 59-79 (1993).
4. Jia, H. Z., Fuh, J. Y., Nee, A. Y. C. and Zhang, Y. F., "Integration of genetic algorithm and Gantt chart for job shop scheduling in distributed manufacturing systems." *Computers & Industrial Engineering*, **53**(2): 313-320 (2007).
5. Bancila, D. and Buzatu, C., "A hybrid algorithm for job shop scheduling." *In: Proceedings of 6th International DAAAM Baltic Conference*, Industrial Engineering, pp. 1-6 (2008).
6. Holland, J. H., "Adaptation in Natural and Artificial Systems." University of Michigan Press, Ann Arbor, MI, USA (1975).
7. Mathew, T. V., "Genetic algorithm." Indian Institute of Technology, Mumbai (2012).
8. Nawaz M. S., Lali M. I. and Pasha M. A., "Formal verification of crossover operator in genetic algorithms using Prototype Verification System (PVS)", *In: Proceedings of 9th IEEE International Conference on Emerging Technologies*, pp. 1-6 (2013).
9. Davis, L., "Job shop scheduling with genetic algorithms.' *In: Proceedings of International Conference on Genetic Algorithms and their Applications*, Vol. 140 (1985).
10. Jain, A. S. and Meeran, S., "Deterministic job-shop scheduling: past, present and future." *European Journal of Operational Research*, **113**(2): 390–434 (1999).
11. Li, Y. and Chen, Y., "A genetic algorithm for job-shop scheduling." *Journal of Software*, **5**(3): 269-274 (2010).
12. Lee, C. Y., Piramuthu, S. and Tsai, Y. K., "Job shop scheduling with a genetic algorithm and machine learning." *International Journal of Production Research*, **35**(4): 1171-1191 (1997).
13. Onwubolu, S. and Davendra, D., "Scheduling flow shops using differential evolution algorithm." *European Journal of Operational Research*, **171**(2): 674-692 (2006).
14. Bierwirth, C., Kopfer H., Mattfeld D. C. and Rixen I., "Genetic algorithm based scheduling in a dynamic manufacturing environment." *In: proceedings of International Conference on Evolutionary Computation*, Vol. 1, pp. 439 (1995).
15. Meeran, S. and Morshed, M. S., "A hybrid genetic tabu search algorithm for solving job shop scheduling

problems. A case study." *Journal of Intelligent Manufacturing*, **23** (4): 1063-1078 (2012).

16. Williamson, D. P., Hall, L. A., Hoogeveen, J. A., Hurkens, C. A. J., Lenstra, J. K., Sevast'Janov, S. V. and Shmoys, D. B., "Short shop schedules." *Operational Research*, **45** (2): 288-294 (1997).
17. Lageweg, B. J., Lenstra, J. K. and Rinnooy Kan, A. H. G., "Job-shop scheduling by implicit enumeration." *Management Science*, **24** (4): 441-450 (1997).
18. Uckun, S., Bagchi. S., Kawamura, K. and Miyabe, Y., "Managing genetic search in job shop scheduling." *IEEE Expert*, **8** (5): 15-24 (1993).
19. Davis, L., "Applying adaptive algorithms to Epistatic domains." *In: Proceedings of the International Joint Conference on Artificial Intelligence, pp.* 162-164 (1985).
20. Lee, K. M., Yamakawa. T. and Lee. K. M. "A genetic algorithm for general machine scheduling problems." *In: Proceedings of 2nd International Conference on Knowledge-Based Intelligent Electronic Systems, pp.* 60–66 (1998).