Genetic Algorithm for Permutation Flowshop Scheduling Problem to Minimize the Makespan

Pervaiz Iqbal[1], P.S. Sehik Uduman[2]

[1]Research Scholar, [2]Professor & Head,

[1,2]Department of Mathematics, B.S. Abdur Rahman University, Chennai, INDIA

pervaizmaths@gmail.com

Abstract: Generally the Flowshop Scheduling Problem (FSSP) is a production environment problem where a set of n jobs has to visit a set of m machines in the same order. In permutation flowshops the sequence of jobs is the same on all machines with the objective of minimizing the sum of completion times using Genetic Algorithm. A significant research effort has been devoted for sequencing jobs in a flowshop for minimizing the makespan. No machine is allowed to remain idle when a job is ready for processing. This paper, describes the Permutation Flowshop Scheduling Problem (PFSSP) solved by using Genetic Algorithm (GA) to minimize the makespan. The basic concept of genetic algorithm is, that it is developed for finding near to optimal solution for the minimum makespan of the n jobs, m machines permutation flowshop scheduling problem. It shows that the innovative genetic algorithm approach which provides competitive results for the solution of Permutation Flowshop Scheduling Problem.

Key Words: Flowshop Scheduling, Permutation Flowshop Scheduling, Genetic Algorithm, Makespan.

1. Introduction.

Scheduling of operations is one of the most critical issues in the planning and scheduling of manufacturing processes. Finding the best schedule can be very easy or difficult, depending on the shop environment. Flowshop scheduling problem have been extensively studied due to its application in industrial engineering and has attracted attention from many researchers. The general flowshop scheduling problem is a production engineering problem where a set of n jobs have to be processed with identical flow pattern on m machines. Flowshop problem is usually known as NP-hard problem.

The permutation flowshop scheduling problem (PFSSP) is a generalization of the classical Flow Shop Scheduling Problem (FSSP) where operations allowed and processed on the sequence of jobs is the same on all machines with the objective of minimizing the sum of completion times. PFSSP is more difficult than the classical FSSP, since it introduces a further decision level beside the sequencing one, i.e., the job routes. When the sequence of job processing on all machines is same, then it is said to have the permutation flowshop sequencing production environment. As passing of the jobs are not allowed, and the number of possible schedules for n jobs is $n!$. Usually, the schedule performance measure is related to an efficient resource utilization looking for a job sequence that minimizes the makespan which is the total time to complete the schedule. Generally, scheduling problem depends on following assumptions:

i) A job has some operations that each of them is to be performed on a specified machine. Some jobs may not be processed on some machines so that the processing time for them is zero.
ii) Jobs are allowed to wait between two stages, and the storage is unlimited.
iii) Setup times are included in the processing times and the operations are sequence-independent.
iv) At a time all jobs are processed on only one machine and every machine processes only one job.
v) The preemption of the job operations on the machines may not be allowed.
vi) No more than one operation of the same job can be executed at a time as well as machines cannot process more than one job at the same time.

The flowshop scheduling problem discussed here, the order in which jobs are processed on various machines is the same and is completely specified. Since, the number of machines is arbitrary, the machines may be numbered such that jobs are processed on machine 1 first, machine 2 second,..., and machine m last [1]. With such a nomenclature of machines, the flowshop scheduling problem may be stated as:

"Given n jobs has to be processed on m machines in the same order, the processing time of job i on machine j being $t_{ij}(i = 1,2,...,n; j = 1,2,...,m)$, the problem is to

find the order in which these n jobs should be processed on the m machines such that the total elapsed time (makespan) is minimum". Consider a partial sequence or containing k, i.e. (k < n) jobs and the augmentation of job α to σ represented by the concatenation of α and σ (written as $\sigma\alpha$). Following the assumptions outlined by R.A. Dudek et al. [2] the recursive relation for the completion time of the partial sequence $\sigma\alpha$ of length $(k+1)$ at machine m, $T(\sigma\alpha, m)$, is as follows [3, 4, 5].

$$T(\sigma\alpha, m) = \max[T(\sigma, m); T(\sigma\alpha, m-1)] + t_{\alpha m}$$

where

$$T(\phi, m) = T(\sigma, O) = 0 \qquad \text{for all } \alpha \text{ and } m.$$

Then, the flowshop scheduling problem as stated above is to minimize $T(\sigma\alpha, m)$ where α ranges over all the n jobs and σ ranges over all possible sequences of $(n-1)$ jobs not containing job α [4].

1.1. Flowshop Scheduling Problem

The Flowshop Scheduling Problems (FSSP) determines an optimum sequence of n jobs to be processed on m machines in the same order i.e. every job must be processed on machines $1,2,...,m$ in this same order.

1.2. Permutation Flowshop Scheduling

The Permutation Flowshop Scheduling Problems (PFSSP) is a special case of FSSPs where same job sequence is followed in all machines i.e. processing order of the jobs on the machines is the same for every machine.

1.3. Makespan

It is the completion time between the start of the first job on first machine and the completion of last job on last machine.

1.4. Genetic Algorithm in PFSSP

In scheduling, genetic algorithm represents the schedules as individuals or a population's members. Each individual has its own fitness value which is measured by the objective function. The procedure works iteratively and this iteration is a generation. Each generation consists of individuals who survive from the previous generations. Usually, the population size remains constant from one generation to the next generation.

2. Literature Review

Flowshop scheduling is one of the most well-known problems in the area of scheduling in Production Engineering. GA has been applied to combinatorial optimization problems such as traveling salesman problem, shortest path problem and scheduling problem etc. A significant research effort has been devoted for sequencing jobs in a permutation flowshop scheduling with the objective of finding a sequence that minimizes the makespan. For problems with 2 machines, or 3 machines under specific constraints on job processing times, the efficient Johnson's Algorithm i.e., S.M. Johnson [6] obtains an optimal solution for the problem. However, this scheduling problem is a NP-hard problem. In Garey et al. [7] the search for an optimal solution is of more theoretical than practical importance. Since, 1960s a number of heuristic methods have provided an optimal solution with limited computation effort for flowshop sequencing.

One of earliest genetic algorithms for the PFSP was proposed by Chen et al. [8]. Murata et al. [9] applied GA to flowshop scheduling problems and examined the GA with other search algorithms. First, they examined various genetic operators to design a genetic algorithm with an objective of minimizing the makespan. They use two point crossover and the shift change mutation for the problem and compared the GA with other search algorithms, e.g. Local Search, Tabu Search and Simulated Annealing. The algorithm is hybridized with local search which resulted in a clear performance gain over a non hybrid version. Another hybrid GA is due to Reeves [10]. More recently, Ponnambalam et al. [11] evaluated a genetic algorithm with a Generalized Position Crossover or GPX crossover, a shift mutation and random population initialization.

Recently, Ruiz et al. [12] have proposed two new advanced genetic algorithms for the problem considered here that provided good results at the expenses of some added complexity in the proposed methods. In Pervaiz Iqbal et al. [13], job sequencing problem using advanced heuristics techniques, in which row sum method is discussed for solving the job sequencing problem in order to minimize the total elapsed time of the sequence. Pervaiz Iqbal et al. [14] deals with the behavior of the digestive system of a paper making plant based on queuing theory. By assuming the serving rate of first part is much lower than the serving rate of last and are taken into consideration that the capacity of each part are different.

2.1. Genetic Algorithm

The Genetic Algorithm (GA) is a powerful optimization search technique based on the principle of natural selection of the genes for constrained and combinatorial optimization problems. The GA was proposed by John Holland in 1975 to encode the factors of a problem by chromosomes where each gene represents a feature of the problem. A GA allows a population composed of many individuals to evolve under

specific selection rule to a state that maximize the fitness function (i.e., minimize the cost function). Here in n job m machine sequencing problem, job is represented as a string, the two point crossover and mutation operators to produce an offspring are used. The crossover probability is calculated as the minimum elapsed time taken over by maximum elapsed time of the optimal sequence. The basic purpose of using the crossover probability is to select the better individuals for crossover and mutation. In this work a sequence of jobs is formed from the given set of jobs using GA.

2.2. Critical Block and Neighborhood Search

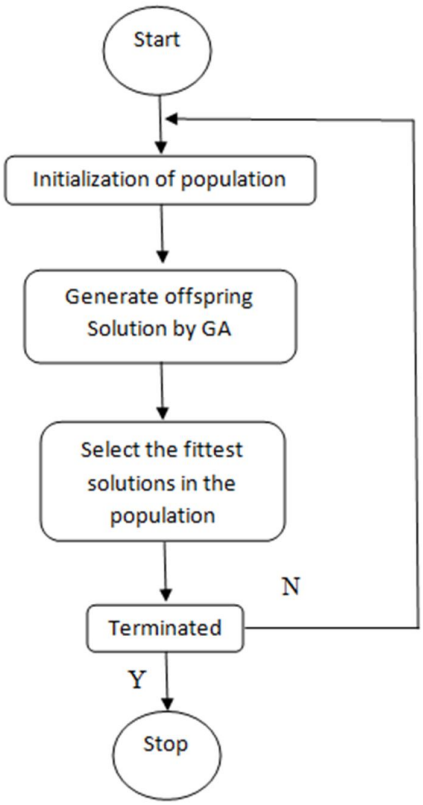

Figure 1: Flow Chart of genetic Algorithm

In flowshop scheduling problem, the critical path could be found. Critical path is defined as the longest paths taken from the first operation processed until the last operation leaves the workspace. All operations in this path are called critical operations and the critical operation on the same machine are called critical block. The distance between any two schedules by the number of differences in the processing orders of operation on each machine is known as disjunctive graph distance (DG distance). The neighborhood search is the most widely used technique in combinatorial problems. A solution S is represented as a point in the search space. The feasible solutions in the neighborhood that can be reached from S with exactly one transition are defined by N(S). Neighborhood search is categorized according to the given criteria for selecting a new point from the neighborhood Yamada et al. [15]. Neighborhood is defined if the DG (Disjunctive Graph) distance between S and $y \in N(S)$ is equal to one. Another type of transition that is also well known is called Adjacent Swapping (AS). This transition operator exchanges a pair of consecutive operations only on critical path to form neighborhood. According to Yamada et al. [16] another transition operator that is more powerful than AS is called Critical Block neighborhood (CB Neighborhood). CB neighborhood permutes the order of operations in critical block by moving the operation to the beginning or end of the critical block. Nagar et al. [17] proposed a combined branch-and-bound and genetic algorithm based procedure for a flow shop scheduling problem with objectives of mean flow time and make-span minimization. Similarly, Neppalli et al. [18] used genetic algorithms in their approach to solve the 2-machine flow shop problem with the objective of minimizing make-span and total flow time.

2.3. Selection Criteria

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time consuming.

Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Popular and well studied selection methods include roulette wheel selection and tournament selection.

2.4. Crossover Analysis

Next step is to perform crossover. This operator selects genes from parent chromosomes and creates a new offspring. The simplest way is to choose randomly some crossover point and everything before this point copy from a first parent and then everything after a crossover point copy from the second parent. Crossover can then look like this (| is the crossover point):

Table 2.1: Single Point Crossover

Chromosome 1	**1110110 | 1001001101100**
Chromosome 2	1110110 | 1011000011110
Offspring 1	**1110110** | 1011000011110

Offspring 2	1110110 | 1**001001101100**

There are many other ways to make the crossover. For example, more crossover points are chosen based on research problem. Crossover can be rather complicated and depends on encoding of the chromosome. Specific crossover made for a specific problem can improve performance of the genetic algorithm.

2.5. Mutation

After the crossover operator is performed, mutation operator takes place. This is to prevent falling all solutions in population into a local optimum of solved problem. Mutation changes randomly the new offspring. For binary encoding we can switch a few randomly chosen bits from 0 to 1 or from 1 to 0. Mutation can then be following:

Table 2.2: Mutation

Original offspring 1	**1110110**1011000011110
Mutated offspring 1	11100101110000111100
Original offspring 1	**1110110**1011000011110
Mutated offspring 1	11100101110000111100

Since, mutation depends on encoding of genes for offspring as well as crossover. For example while encoding permutations; mutation could be exchanging two genes.

3. Flow Chart of GA.

1. **Start**: - Generate random population of n chromosomes (suitable solutions for the problem).
2. **Fitness**: - Evaluate the fitness $f(x)$ of each chromosome x in the population.
3. **New population**: - Create a new population by repeating following steps until the new population is complete.
   - 3.1 **Selection**: - Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).
   - 3.2 **Crossover**: - With a crossover probability parents form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
   - 3.3 **Mutation**: - With a mutation probability mutate new offspring at each locus (position in chromosome).
   - 3.4 **Accepting**: - Place new offspring in a new population.
4. **Replace**: - Use new generated population for a further run of algorithm.
5. **Test**: - If the end condition (for example number of populations or improvement of the best solution) is satisfied, stop and return the best solution in current population.
6. **Loop**: - Go to step 2.

4. Results and Discussion.

Consider a problem of 5 jobs and 5 machines with the operation sequence and the processing time for each operation have been determined in the Table 4.1 and Table 4.2 given below.

The programme is run for five times using the population size = 100, number of iterations for mutation is 0.015 and crossover is 0.90. The algorithm was terminated after 200 generations, from the result in Table 4.3, it can be shown that the combination of critical block, and genetic algorithm could provide a better result compared to other methods. From Table 4.3, it is seen that the last job (i.e. job 4) is processed on machine 5 and job 5 is processed on machine 4, the makespan value for both is same 36. There is a tie, so we need to select any one of the makespan. The result also gives us the job sequence for each machine to process, the starting time and the finish time for each operation. For example, on machine 1, job 3 at time 0 is started to process and finished at 7. Then job 1, followed by job 5, job 2 and job 4 is processed.

Table 4.1: Processing Time for each Operation

	Machine-1	**Machine-2**	**Machine-3**	**Machine-4**	**Machine-5**
Job-1	8	4	2	6	7
Job-2	3	6	5	2	4
Job-3	7	3	9	4	8
Job-4	4	5	5	4	3
Job-5	3	6	7	4	5

Table 4.2: Operation Sequence

	Sequence-1	**Sequence-2**	**Sequence-3**	**Sequence-4**	**Sequence-5**
Jo	Machi	Machi	Machi	Machi	Machi

b-1	ne-3	ne-1	ne-2	ne-4	ne-5
Job-2	Machine-2	Machine-3	Machine-5	Machine-1	Machine-4
Job-3	Machine-1	Machine-5	Machine-4	Machine-3	Machine-2
Job-4	Machine-4	Machine-3	Machine-2	Machine-1	Machine-5
Job-5	Machine-5	Machine-3	Machine-1	Machine-2	Machine-4

Table 4.3: Result and Analysis

Machines/ Operations Sequence	Starting Time	Processing Time	Finishing Time
Machine-1			**25**
Job-3	0	7	7
Job-1	7	8	15
Job-5	23	3	26
Job-2	26	3	29
Job-4	29	4	33
Machine-2			**24**
Job-2	0	6	6
Job-1	15	4	19
Job-4	19	5	24
Job-5	26	6	32
Job-3	32	3	35
Machine-3			**28**
Job-1	0	2	2
Job-2	6	5	11
Job-4	11	5	16
Job-5	16	7	23
Job-3	23	9	32
Machine-4			**20**
Job-4	0	4	4
Job-3	15	4	19
Job-1	19	6	25
Job-2	29	2	31
Job-5	32	4	**36**
Machine-5			**27**
Job-5	0	5	5
Job-3	7	8	15
Job-2	15	4	19
Job-1	25	7	32
Job-4	33	3	**36**

Both types of initial population to the data are used. First, the combination of schedules is used, which generated the priority rules and the randomly generate schedules as the initial population. From the five runs, the optimum result found before the generation exceeded 100. Also it could be concluded that if randomly generated schedules as initial population is used, then only the optimum value at generation larger than 100 is found. However, both results gave the same makespan value which is 36.

5. Conclusions

In this paper, the problem of scheduling the jobs in permutation flowshop scheduling problem is considered. The main aim of this research is to explore the potential of genetic algorithms. However, as an observation, it is seen that for certain types of problem, it may not be worth using sophisticated procedures. Since, a simple neighborhood search can obtain solutions of comparable quality very easily. The overall implication of the studies carried out that the Genetic Algorithm (GA) produce good results for permutation flowshop scheduling problem for most sizes and types of problem, and that it will reach a near to optimal solution rather more quickly. The study of permutation flowshop scheduling problem using genetic algorithm provides a rich experience for the constrained combinatorial optimization problems. Application of genetic algorithm always gives a good result most of the time.

In near future, this technique could be applied to a larger size problem to check out its compatibility and adaptability.

References

[1] J.N.D. Gupta, “M-stage Scheduling Problem – A Critical Appraisal”, The International Journal of Production Research, 8, No. 2, 1971, pp. 276-281.
[2] R.A. Dudek and O.F. Teuton, “Development of M-stage decision rule for scheduling n-jobs

through M-machines", Ops Res, Vol.12, pp. 471, 1964.
[3] A.P.G. Brown and Z.A. Lomnicki, "Some Applications of the Branch and Bound Algorithm to the Machine Scheduling Problem", Operational Research Quarterly, 17, No. 2, 1966, pp. 173-186.
[4] J.N.D. Gupta, "A General Algorithm for the n × M Flowshop Scheduling Problem", The International Journal of Production Research, 7, No. 3, 1969, pp. 241-247.
[5] J.N.D. Gupta, "M-stage Flowshop Scheduling Problem by Branch and Bound", Opsearch, (India) 7, No. 1, 1970, pp. 37-43.
[6] S.M. Johnson, "Optimal two-and three-stage production schedules with setup times included", Naval Research Logistics Quarterly, Vol.1, pp.61-68, 1954.
[7] M.R. Garey, D.S. Johnson and R. Sethi, "Complexity of flow-shop and job-shop scheduling," Mathematics of Operations Research, Vol.1, Issue 2, pp.117-129, 1976.
[8] C.L. Chen, V.S. Vempati and N. Aljaber, "An application of genetic algorithms for flow-shop problems", European Journal of Operational Research, Vol.80, pp.389-396, 1995.
[9] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flow shop scheduling problems", Computers and Industrial Engineering, Vol.30, pp.1061-1071, 1996.
[10] C.R. Reeves, "A Genetic algorithm for flow-shop sequencing", Computers and Operations Research, Vol.22, Issue 1, pp.5-13, 1995.
[11] S.G. Ponnambalam, P. Aravindan, and S. Chandrasekaran, "Constructive and improvement flow shop scheduling heuristics: an extensive evaluation", Production Planning and Control, Vol.12, Issue 4, pp.335-344, 2001.
[12] R. Ruiz, and C. Maroto, "A comprehensive review and evaluation of permutation flow-shop heuristics", European Journal of Operational Research, Vol.165, pp.479-494, 2004.
[13] Pervaiz Iqbal, P.S. Sheik Uduman and S. Srinivasan, "Job sequencing problem using advanced heuristics techniques", Proceedings of the International Conference on Applied Mathematics and Theoretical Computer Science, Vol.1, pp.15-18, 2013.
[14] Pervaiz Iqbal and P.S. Sehik. Uduman, "Mathematical modeling and behavior of the digestive system of a paper making plant based on queuing theory", International Journal of Pure and Applied Mathematics, Vol. 90, Issue 1, pp.43-56, 2014.
[15] T. Yamada and R. Nakano, "A Genetic Algorithm with Multi-Step Crossover for Job-shop Scheduling Problems", International Conference on Genetic Algorithms in Engineering Systems: Innovations and Application (GALESIA '95), 1995.
[16] T. Yamada and R. Nakano, "Genetic Algorithms for Job-shop Scheduling Problems", Proceedings of the Modern Heuristic for Decision Support. pp. 67-81, UNICOM Seminar, 8-19 March 1997, London, 1997.
[17] A. Nagar, S.S. Heragu, J. Haddock, "A combined branch-and-bound and genetic algorithm based approach for a flowshop-scheduling problem", Annal of Operations Research, 63 (1996), 397–414.
[18] V.R. Neppalli, C.L. Chen, J. N. D. Gupta, "Genetic algorithms for the two-stage bicriteria flow shop problem", European Journal of Operations Research, 95 (1996), 356–373.