MINIMIZING MAKESPAN FOR AN AEROSPACE MANUFACTURING SYSTEM USING SIMPLE GENETIC OPERATIONS

Alvin Mark Windsor[1], V.Vivekanand[2], G.S.Prakash[3], Sriram Anupindi[4]

[1]UG student, Department of IEM, MSRIT, Bengaluru
[2]Assistant Professor, Department of IEM, MSRIT, Bengaluru
[3]Professor & Head, Department of IEM, MSRIT, Bengaluru
[4]UG student, Department of CSE, MSRIT, Bengaluru

Abstract

In a job shop scheduling problem as the number of jobs and machines becomes larger, identifying the optimum schedule to achieve a particular objective becomes complex. Recent development in this field has led to approximating algorithms designed to handle a large solution space. Some of which include Genetic Algorithm, Tabu Search, Simulated Annealing and Particle Swarm Optimization. These algorithms can be used to obtain a near optimal schedule with improved solution efficiency. This paper discusses the development of a deterministic model to schedule 12 jobs on 5 machines with the objective of minimizing makespan using simple genetic operators.

Keywords: *Job Shop, Genetic Algorithm, Makespan, Scheduling, Sequencing.*

----------***----------

1. INTRODUCTION

The job shop problem is a highly complex combinatorial optimization problem. It is typically categorized as Non-Deterministic Polynomial Hard (NP-Hard). The solution to such problems may not be obtained in polynomial time using non-deterministic Turing machine.

To gain a very brute understanding of NP-Hard problems, consider a decision problem. The solution to a decision problem always ends in a "yes" or "no" solution. A computer is an example of a Turing machine which can be used to answer decision problems like does 1+2=3.

A non-deterministic Turing machine ideally has unlimited parallelism. It can take multiple paths at a decision point. In a job shop scheduling problem [JSSP] there are multiple feasible schedules. Thereby, such problems become more complex to handle.

The size of the solution space for a job shop scheduling problem can be estimated using the expression below. If there are X jobs, each job passing through Y machines. Then the number of possible solutions Z is given below.

$$Z = (X!)^Y$$

As the solution space increases conventional methods like mixed integer linear programming (MILP) prove inefficient in solving them. Recent developments in this field have led to approximating techniques such as genetic algorithm, simulated annealing and tabu search. These algorithms were developed to solve problems in which the solution space is so vast that brute force optimization algorithms would take significantly more time.

Genetic algorithm has wider scope and is more abstract when compared to simulated annealing or tabu search. The algorithm is built with the objective of solving problems with a large solution space. When genetic algorithm is applied to a job shop scheduling problem, each schedule is considered as an entity in the population. Each entity is then evaluated on the basis of a defined fitness function value which correlated to the objective function. The algorithm executes in iterations, each iteration is called a generation.

At each generation the new schedules are evolved from the members of the previous generation. Therefore, it evolves an approximate solution to the problem based on the fitness value of the previous generation through the use of genetic operations.

This paper focuses on building a deterministic model using simple genetic operations to solve 12 jobs on 5 machines scheduling problem. The objective is to minimize makespan (i.e. the time taken to process a set of jobs). The selection and permutation genetic operators are used to evolve a feasible schedule with minimum makespan.

2. PROBLEM DESCRIPTION

A (12Jx5M) scheduling problem was considered.

Table 1: Summary of the (12Jx5M) scheduling problem

Job Designation	Machine Allocated	Processing Designation	Processing Time
J1	M13	P13,1,1	1490
	M13	P13,1,2	1240
	M9	P9,1,1	3360
	M9	P9,1,2	1860
	M6	P6,1,1	2360
	M6	P6,1,2	2360
J2	M13	P13,2,1	412.5
	M13	P13,2,2	378
	M9	P9,2,1	774
	M9	P9,2,2	567
	M6	P6,2,1	636
	M6	P6,2,2	636
J3	M13	P13,3,1	1907.5
	M13	P13,3,2	1574
	M9	P9,3,1	4362
	M9	P9,3,2	2361
	M6	P6,3,1	3028
	M6	P6,3,2	3028
J4	M13	P13,4,1	399.5
	M13	P13,4,2	443
	M9	P9,4,1	1520
	M9	P9,4,2	505
	M9	P9,4,3	650
J5	M11	P11,5,1	1860
	M11	P11,5,2	1260
	M13	P13,5	1860
	M11	P11,5,3	1740
	M14	P14,5	3660
J6	M11	P11,6,1	1360
	M11	P11,6,2	960
	M13	P13,6	1320
	M11	P11,6,3	1240
	M14	P14,6	2560
J7	M14	P14,7	3760
	M11	P11,7,1	1360
	M11	P11,7,2	1360
	M11	P11,7,3	2160
J8	M14	P14,8	5460
	M11	P11,8,1	1860
	M11	P11,8,2	1860
	M11	P11,8,3	3060
J9	M11	P11,9,1	760
	M11	P11,9,2	760
J10	M11	P11,10,1	760
	M11	P11,10,2	760
J11	M6	P6,11,1	1376
	M6	P6,11,2	741
	M11	P11,11,1	868
	M11	P11,11,2	868
J12	M6	P6,12	1360
	M14	P14,12	2360

The first column represents the jobs (J_1, J_2,.., J_{12}). The second column represents the processing sequence of the respective job. For example, J_2 has the following processing sequence: M_{13}» M_{13}» M_9» M_9» M_6» M_6. The third column represents the process designation; each process was given a unique notation in the form of $P_{I,J,K}$. Where I represent the machine (M_I), J represents the job (J_J) and K represents the sub-process. The fourth column represents the processing time (in minutes) of the respective job J_J on machine M_I.

The objective is to build a deterministic model that provides a schedule to process these jobs with minimum makespan. The constraints to be taken into consideration are as follows:
[1] Each machine can process at most one job at any given time.
[2] The precedence constraint of any job cannot be violated. For example J_2 can be processed on M_9 only after its process completion on M_{13}.

3. LITERATURE REVIEW

The productivity and growth of a manufacturing system can be maximized if the available resources are utilized effectively. Optimized utilization of resources can only be possible if a proper scheduling system is in place. This makes scheduling systems a highly important aspect of a manufacturing system.

David Applegate and William Cook [1] define job shop scheduling problem as a problem which deals with scheduling a set of jobs to one or more machines, subject to constraints that each machine can handle at most one job at a time and each job has a specified processing order through the machines.

Brucker P. [2], through his study on scheduling algorithm comments on the NP-Hard nature of typical job shop problems. They belong to a group of complex combinatorial optimization problems.

The flowchart shows the various techniques to solve JSSP. Optimization based techniques are limited to smaller sized problems whereas approximating algorithms are more efficient in handling slightly larger problems. Approximating algorithms provide sub optimal schedules.

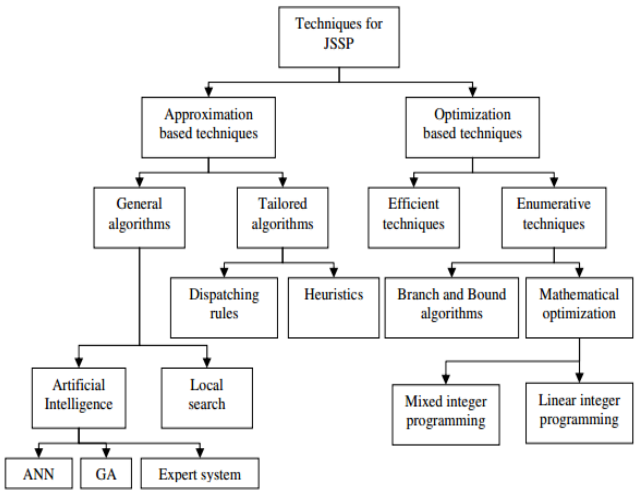

Fig. 1: Flowchart showing the various techniques to solve JSSP[3]

Lestan Z. et al. [4] concludes "The algorithms which are based on the method of branch and bound are useful only for solving small instances (around 20 jobs). For large scale problems approximating algorithms such as shifting bottleneck, genetic algorithm, simulated annealing, tabu search, priority dispatch, etc are used".

The best results are obtained with the use of hybrid methods involving genetic algorithm, tabu search, simulated annealing and the shifting bottleneck approach. This paper highlights the development of a simple model which attempts to find the schedule with the smallest makespan for the given (12Jx5M problem).

Wallace J. Hopp and Mark L. Spearman[5] in their book Factory Physics define makespan as the time required for processing a set of jobs.

4. METHODOLOGY

The model was coded using Python language on Jupyter Notebook compiler. The extensive literature survey revealed a common challenge of infeasible schedules faced in genetic algorithm. Infeasible schedules would be of little importance in a practical job shop situation. Therefore, this model was used to evolve and test feasible schedules for optimality.

The methodology can be divided into three stages.

Fig. 2: Flowchart showing the various stages of solving the JSSP using the deterministic model

4.1 Evolving Schedules

Each schedule is represented by a string in the model. A string is composed of elements each of which is represented as a triplet. A triplet is composed of the job, respective machine and corresponding process designation. The triplet $[J_4,M_9,P_{9,4,1}]$ represents the processing of job (J_4) on machine (M_9) sub-process ($P_{9,4,1}$).

Every job has a fixed processing sequence therefore the precedence constraint cannot be violated. A schedule is said to be feasible only if it doesn't violate the precedence constraint of any job. In contrast, to introduce variability in the search mechanism random schedules have to be created. Therefore, a randomizer was used to evolve schedules without violating the precedence constraint. The table below shows an example.

Table 2: Example of a (3Jx3M) scheduling problem

Jobs	Processing Order			Processing Times		
J_1	M_1	M_3	-	$P_{1,1}$	$P_{3,1}$	-
J_2	M_2	M_1	M_3	$P_{2,2}$	$P_{1,2}$	$P_{3,2}$
J_3	M_1	M_3	M_2	$P_{1,3}$	$P_{3,3}$	$P_{2,3}$

The table above shows an example of three jobs, their respective processing sequence and the processing times. To create a schedule while maintaining feasibility separate lists are created for each job.

J_1: (J_1, M_1, $P_{1,1}$), (J_1, M_3, $P_{3,1}$)
J_2: (J_2, M_2, $P_{2,2}$), (J_2, M_1, $P_{1,2}$), (J_2, M_3, $P_{3,2}$)
J_3: (J_3, M_1, $P_{1,3}$), (J_3, M_3, $P_{3,3}$), (J_3, M_2, $P_{2,3}$)

Let S_1 represent the string to be created. A randomizer is used to allocate elements to this string. The randomizer function is used to select a job between J_1, J_2 and J_3. In this example consider J_3 is selected first. Then the first element from the J_3 list is selected and placed in string S_1, following which the element is deleted from the respective job's list.

S_1: [(J_3, M_1, $P_{1,3}$)]

J_1: (J_1, M_1, $P_{1,1}$), (J_1, M_3, $P_{3,1}$)
J_2: (J_2, M_2, $P_{2,2}$), (J_2, M_1, $P_{1,2}$), (J_2, M_3, $P_{3,2}$)
J_3: (J_3, M_3, $P_{3,3}$), (J_3, M_2, $P_{2,3}$)

This process is repeated till the entire schedule is built.

S_1: [(J_3, M_1, $P_{1,3}$), (J_2, M_2, $P_{2,2}$), (J_3, M_3, $P_{3,3}$), (J_1, M_1, $P_{1,1}$), (J_2, M_1, $P_{1,2}$), (J_1, M_3, $P_{3,1}$), (J_3, M_2, $P_{2,3}$), (J_2, M_3, $P_{3,2}$)]

The above string represents a feasible schedule. For the given (12Jx5M) problem, eight such schedules were created. Each schedule consisting of 51 triplets was tested for the fitness function value based on the objective of minimizing makespan.

4.2 Selection Operation

Selection operator is the only genetic operation free from encoding. As selection does not change the string we do not face issues with maintaining feasibility. D. Shiffman [6], in his book, "The nature of Code" discusses other genetic operators like the crossover operation often produce infeasible schedules, which is difficult to repair.

From the tested schedules, the string with the best fitness function value is selected using the selection genetic operator.

4.3 Permutation Operation

The selection operator identifies the string with the best fitness function. Permutation operation is applied on this string; this operation shuffles the elements within a string. When performing the permutation shuffles it is critical to maintain feasibility. Consider the example shown above.

S_1: [{(J_3, M_1, $P_{1,3}$), (J_2, M_2, $P_{2,2}$), (J_3, M_3, $P_{3,3}$), (J_1, M_1, $P_{1,1}$), (J_2, M_1, $P_{1,2}$), (J_1, M_3, $P_{3,1}$), (J_3, M_2, $P_{2,3}$), (J_2, M_3, $P_{3,2}$)}]

The permutation shuffling is applied to the elements within the curly braces without violating the precedence constraints.

S_1: [{(J_1, M_1, $P_{1,1}$), (J_2, M_2, $P_{2,2}$), (J_3, M_1, $P_{1,3}$), (J_1, M_3, $P_{3,1}$), (J_3, M_3, $P_{3,3}$), (J_2, M_1, $P_{1,2}$), (J_2, M_3, $P_{3,2}$), (J_3, M_2, $P_{2,3}$)}]

After shuffling the string is tested for its fitness function value. If the value is better than the original string it stores this string in the memory function and performs the permutation operation again.

This process terminates when the defined termination condition is encountered.

5. RESULTS AND DISCUSSION

From the eight randomly generated strings tested in the model, string 1 was selected using the selection operation. It had a fitness function value of 26690, being the lowest of the eight in comparison.

```
In [16]: print "Terminated String with minimum fitness function value",term
         print termlist

Terminated String with minimum fitness function value 26690
[['J2,M13,P;13;2;1', 'J2,M13,P;13;2;2', 'J4,M13,P;13;4;1', 'J4,M13,P;13;4;2', 'J12,M6,P;6;12', 'J6,M1
1,P;11;6;1', 'J6,M11,P;11;6;2', 'J5,M11,P;11;5;1', 'J5,M11,P;11;5;2', 'J7,M14,P;14;7', 'J1,M13,P;13;
1;1', 'J11,M6,P;6;11;1', 'J8,M14,P;14;8', 'J1,M13,P;13;1;2', 'J7,M11,P;11;7;1', 'J8,M11,P;11;8;1', 'J
1,M9,P;9;1;1', 'J7,M11,P;11;7;2', 'J1,M9,P;9;1;2', 'J7,M11,P;11;7;3', 'J11,M6,P;6;11;2', 'J8,M11,P;1
1;8;2', 'J1,M6,P;6;1;1', 'J4,M9,P;9;4;1', 'J4,M9,P;9;4;2', 'J1,M6,P;6;1;2', 'J4,M9,P;9;4;3', 'J2,M9,
P;9;2;1', 'J2,M9,P;9;2;2', 'J2,M6,P;6;2;1', 'J8,M11,P;11;8;3', 'J9,M11,P;11;9;1', 'J2,M6,P;6;2;2', 'J
9,M11,P;11;9;2', 'J6,M13,P;13;6', 'J6,M11,P;11;6;3', 'J3,M13,P;13;3;1', 'J3,M13,P;13;3;2', 'J5,M13,P;
13;5', 'J11,M11,P;11;11;1', 'J3,M9,P;9;3;1', 'J3,M9,P;9;3;2', 'J5,M11,P;11;5;3', 'J5,M14,P;14;5', 'J1
1,M11,P;11;11;2', 'J12,M14,P;14;12', 'J6,M14,P;14;6', 'J3,M6,P;6;3;1', 'J3,M6,P;6;3;2', 'J10,M11,P;11;
10;1', 'J10,M11,P;11;10;2'], 'J99,M11,P;99;99', 'J99,M11,P;99;99', 'J99,M11,P;99;99', 'J99,M11,P;99;9
9']
```

Fig. 3: Showing the numerical string output of the deterministic model.

The numerical string generated is difficult to visualize therefore it string was plotted as a Gantt chart using Microsoft Excel.

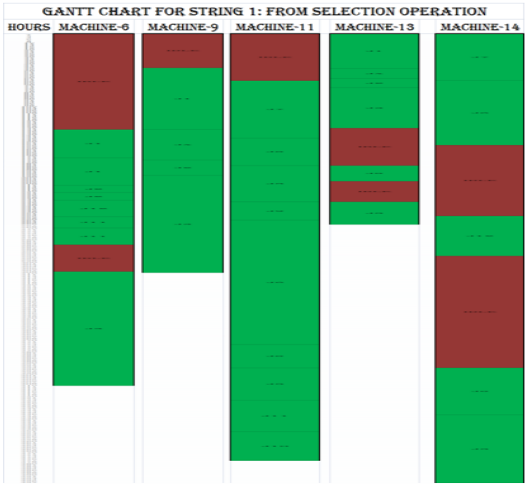

Fig. 4: The Gantt chart for string-1 after the selection operation.

The cells highlighted in green represent the busy state while red represents idle state for the respective machine. The table below summarizes the Gantt chart from the selection stage.

Table 3: Showing machine wise completion time for string-1 after selection operation.

Machine	Completion Time (hrs)
Machine-6	409.99
Machine-9	309.5
Machine-11	476.97
Machine-13	265.68
Machine-14	497.36

The makespan for string-1 after applying the selection operator is 497.36 hours. This string is then subjected to permutation operation. The elements within the string are shuffled without violating the precedence constraints. At each stage the fitness value of the changed string is compared with the original value. On encountering its termination condition the numerical string is plotted as a Gantt chart for better visualization.

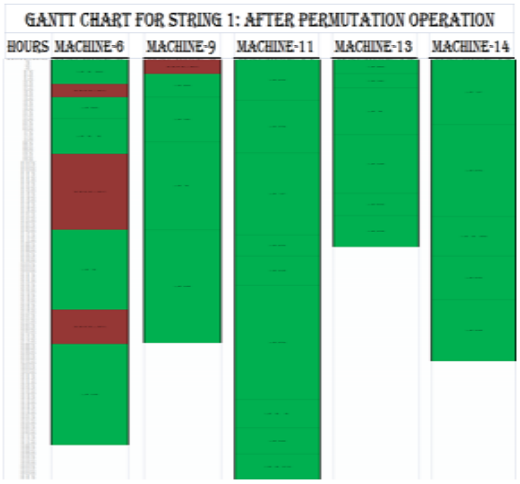

Fig. 5: The Gantt chart for string-1 after the permutation operation.

After applying the permutation operation to string-1 the makespan is reduced to 414.31 hours. This is the optimal schedule generated by the deterministic model.

Table 4: Showing machine wise completion time for string-1 after permutation operation on encountering the terminating condition.

Machine	Completion Time (hrs)
Machine-6	**380.17**
Machine-9	**279.23**
Machine-11	**414.31**
Machine-13	**183.62**
Machine-14	**296.67**

6. CONCLUSION

This deterministic model is used to test a sample set of eight randomly generated schedules. The selection operation is used to select one string from the sample set based on the fitness function value; this string is then optimized using permutation operation. The terminated string has a makespan of 414.31 hours.

REFERENCES

[1]. David Applegate and William Cook (1991). A Computational Study of the Job Shop Scheduling Problem, ORSA Journal on Computing, Vol. 3, No.2, Spring 1991.
[2]. Brucker P. (2001). Scheduling algorithms, Springler-Verlag, Berlin, ISBN 10: 3540415106.
[3]. Anant Singh Jain and Sheik Meeran (1998). A State-of-the-art Review of Job-Shop Scheduling Techniques. Department of Applied Physics, Electronic and Mechanical Engineering University of Dundee, Dundee, Scotland, UK, DD1 4HN.
[4]. Lestan Z, Brezocnik M, Buchmeister B, Brezovnik S and Balic J (2009). Solving the Job-Shop Scheduling Problem with a Simple Genetic Algorithm. ISSN 1726-4529.
[5]. Wallace J. Hopp and Mark L. Spearman (2000). Factory Physics: Foundations of Manufacturing Management, second edition, 2000. 698pp. ISBN 0-256-24795-1.
[6]. D. Shiffman, 2012. The Nature of Code. Publisher: D. Shiffman, 2012. ISBN 0985930802, 9780985930806.