Genetic Algorithm for Job Shop Scheduling Problem: A Case Study

***Abstract*—The job-shop scheduling (JSS) is a schedule planning for low volume systems with many variations in requirements. In job-shop scheduling problem (JSSP), there are k operations and n jobs to be processed on m machines with a certain objective function to be minimized. Due to complexity of transferring work in process product, this research add transfer time variable from one machine to another for each different operation. Performance measures are mean flow time and make span. In this paper we used genetic algorithm (GA) with some modifications to deal with problem of job shop scheduling. The result than is compared with dispatching rules such as longest processing time, shortest processing time and first come first serve. The numerical example showed that GA result can outperform the other three methods.**

***Index Terms*—Job shop, scheduling, genetic algorithm, dispatching rules.**

I. INTRODUCTION

As the market becomes more competitive,the company ability to deliver their goods in the right time becomes necessity. A high variety of product with low quantity comprise between 50% and 75% of all manufactured components,thereby making schedule optimization an indispensable step in the overall manufacturing process. Thus,scheduling optimization holds the main role to catch up with market demand. In general,the scheduling can be described as the allocation of jobs over time when limited resources are available,where a number of objectives should be optimized,and several constraints must be satisfied. A job is determined by a predefined set of operations,and the result of a scheduling algorithm is a schedule that contains the start times and allocation of resources to each operation. The *n* x *m* classical JSP involves *n* jobs and *m* machines. Each job is to be processed on each machine in a predefined sequence and each machine processes only one job at a time.

In practice,the shop-floor setup typically consists of multiple copies of the most critical machines so that bottlenecks due to long operations or busy machines can be reduced. The job shop scheduling problem is one of the hardest combinatorial optimization problem which is belongs to the class of NP-hard problems. Since classical optimization methods (branch and bound method,dynamic programming) can be used only for small scale problem,more complex tasks must be solved by heuristic methods such assimulated annealing, tabu search,evolutionary algorithm, particle swarm and genetic algorithm.

This paper focuses on developing algorithm to solve job shop scheduling problem. The algorithm is designed by considering machine availability constraint and the transfer time between operations. Next, machine availability constraint is described. The machine availability constraint is used to calculate realistic makespan for company that has breaking period during processing time.

II. PROBLEM DEFINITION

This research is focusing on investigating machine scheduling problems in manufacturing and service environments where jobs represent activities and machines represent resources, and each machine can only process one job at a time. In this paper, we will focus on the low volume system also known as job-shop. We will use the real case data gained from manufacturing company.

To simplify the explanation we will use the following notations throughout the paper:

j= job (*j*=1,2,….,*n*)
i= machine (i=1,2,.....,m) tt= transfer time
P= processing time W= waiting time
C= completion time job

In this type of environment, where the products are made to order, the job-shop scheduling problem (JSSP) can be described as follow:

1) Job sets
   $J = \{J_1, J_2, \ldots\ldots, J_j\} \mid j = 1,2,3,\ldots\ldots,n$
2) Machine sets
   $M = \{M_1, M_2, \ldots\ldots, M_i\} \mid i = 1,2,3,\ldots\ldots,m$
3) Operations
   $O = \{O_1, O_2, \ldots\ldots, O_o\} \mid o = 1,2,3,\ldots\ldots,k$
4) Processing time for each operation
   $P_{ij} = \{P_{11}, P_{12}, \ldots\ldots\ldots, P_{ij}\}, \mid i = 1,2,3,\ldots\ldots,n\ ; j = 1,2,3,\ldots\ldots,m$
5) Transfer time
   $t_{ij} = \{t_{11}, t_{12}, \ldots\ldots\ldots, t_{ij}\} \mid i = 1,2,3,\ldots\ldots,n\ ; j = 1,2,3,\ldots\ldots,m$

Furthermore, each job should be processed through the machines in a particular order or also known as technological constraint. The maximal time required for all operations to complete their processes is called makespan while the average time required for all operations is called mean flow time. In this paper, our intention is to minimize two objectives, makespan value and mean flow time.

(1) $C_{\max} = \max\left\{C_j\right\}$

(2)

When minimizing the makespan, at least one of the optimal solutions is a semi-active (no operation can started earlier without violating the technological constraints. For this reason, every time when makespan is optimized, a schedule can be described by the processing orders of operations on the machines. Some assumptions used in this research are:

1) The jobs are independent and consist of strictly ordered operation sequences.
2) No priorities are assigned to any job or operation.
3) Job pre-emption is not allowed.
4) A given operation can be performed by one or more non-identical machines (called alternative machines).
5) The setup times are independent of the operation sequence and are included in the processing times.
6) The transfer time between operations will be occurred whenever there is a machine changes for each job.

The completion time of each job will be follows the equation (3). The transferring time will be included based on the total transfer time between operation perform in each job.

(3)

$$c_j = tt_j + \sum_{i \in I} P_{ij} + \sum_{i \in I} W_{ij}$$

Waiting time will be appears if the job arrive in the machine that still perform another job. Waiting time calculates by subtract the completion time of previous operation with the next operation in certain machine. The equation can be written as:

$$W_{ij} = \left| C_j - M_i \right|$$

$$C_{ij} \geq C_{hi} + P_{ij}$$

$$C_{ij} - C_{ia} \geq P_{ij} \sqrt{C_{ia} - C_{ij}} \geq P_{ia} \text{ if } M_j = M_a$$

$$C_{ji} \geq 0, j, a \in M$$

(4)

(5)

(6)

(7)

Constraint (5) guarantee the operation can only be started after previous operation in the same machine is done. Constraint (6) required to make sure that each machine can only process a job in a time.

III. PROPOSED METHOD

As the JSSP is an NP-Hard problem,the proposed formulation is not applicable to find optimal solutions. Hence,the genetic algorithm (GA) is used to obtain the result of the objectives. Each chromosome/individual of the GA represents a permutation of the work station. The chromosomes with the heuristic cross over and mutation operators are developed trough some repetitions. The genetic algorithm (GA) is a stochastic search technique that mimics the mechanisms of the Darwinian evolution based on the concept of the survival of the fittest [11]. The basic component of a GA is the solution representation, popularly known as the chromosome or individual, which represents a complete solution of a problem. The proposed GA generates a set of permutation as representation solution, where the individual is a result permutation of the work station to be arranged.

A. Initial Solution

The initial solution will be done randomly as a set of job permutation in each machine. The total gen for each chromosome will be equal to the number of operations performs to finish the products.

	M1			**M2**					**Mn**	
1	4	2	3	6	5	9	10	11	10	11
2	8	1	7	6	9	5	10	11	11	12
3	2	3	6	8	11	5	10	9	14	15
.				.	.	.	.	.	.	.
.				.	.	.	.	.	.	.
o	4	2	3	6	5	9	10	11	11	12

Fig. 1. Solution representation

B. Selection

The function of a selection operator is to form a mating consisting of the above-average chromosomes of the population. The mating pool will be used by the crossover and mutation operators with the expectation for generating good offspring chromosomes. The roulette wheel tournament is applied here for this purpose. It picks up two chromosomes from the population and stores a copy of the best chromosomes (based on objective values) in the mating pool. The process is repeated until the size of the mating pool equals that of the population.

C. Cross Over

Using a random procedure, two point cross over is performs. For each couple of parents with single line encoded chromosomes, a random integer is generated to choose the two cross over point. The next step is to swap the range between parent 1 and 2 based on the cross over point. To make sure the feasibility of the solution,this research adopted order cross over.

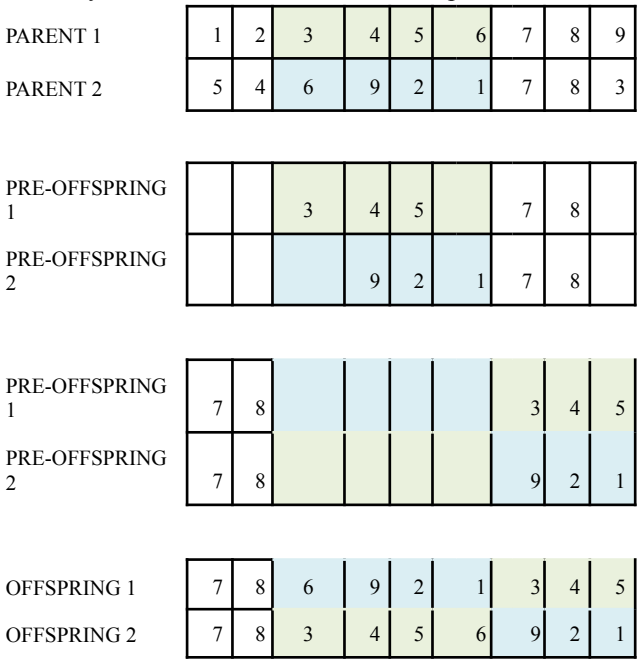


Fig. 2. Two point order cross over

D. Mutation

The mutation operator is applied to create the different new chromosomes and to prevent the population from stagnating in their local optimal solution with a predefined mutation probability. For this research, the mutation is done using swapping mutation. Using this random procedure, the two random integers are generated as the replacing genes.

E. Stopping Criteria

The maximum number of generation (G) is selected as the stopping criteria. In this process from one generation to the next generation, the cross over and mutation is repeated until the maximum number of generation is satisfied.

IV. EXPERIMENTAL DESIGN

The effect of many different parameters on the performance characteristic in a condensed set of experiments can be examined by using the concept of design experiment proposed by Montgomery. Once the parameters affecting a process that can be controlled have been determined, the levels at which these parameters should be

varied must be determined. Determining what levels of a variable to test requires an in-depth understanding of the process, including the minimum, maximum, and current value of the parameter.

TABLE I: PARAMETERS SETTING

Factor	Parameter Setting
Population Size	45
Cross Over Probability	0.55
Mutation Probability	0.13

If the difference between the minimum and maximum value of a parameter is large, the values being tested can be further apart or more values can be tested. If the range of a parameter is small, then less value can be tested or the values tested can be closer together. Some tuning parameter used in the proposed GA will be determined by using DOE method. In GA, there are three parameters considered, population size, crossover probability and mutation probability.

V. RESULT AND DISCUSSION

The maximum number of generation (G) is selected as the stopping criteria. In this process from one generation to the next generation, the cross over and mutation is repeated until the maximum number of generation is satisfied. The proposed algorithm is coded in Matlab. The results are compared with the arrangement of job based on company data using longest processing time (LPT) rule. There are many simple dispatching rules (priority rule) for scheduling. This priority rules are developed to obtain a good schedules for number of different objective in different situation. They are designed for sequencing many jobs without many effort and time. As the benchmark, another dispatching rule is also performs to solved the problem. Shortest processing time (SPT) and first come first serve (FCFS) rule is chosen as the benchmark rule.

A. Data Set

In this section,the instance is solved using the proposed approach to evaluate the effectiveness of the proposed approach. These instances are based on real case problem in molding manufacturing company. We handled the medium instance with 15 jobs and 15 machines. Their optimal solution is unknown. The only company scheduling data is based on LPT rule that they adapt.

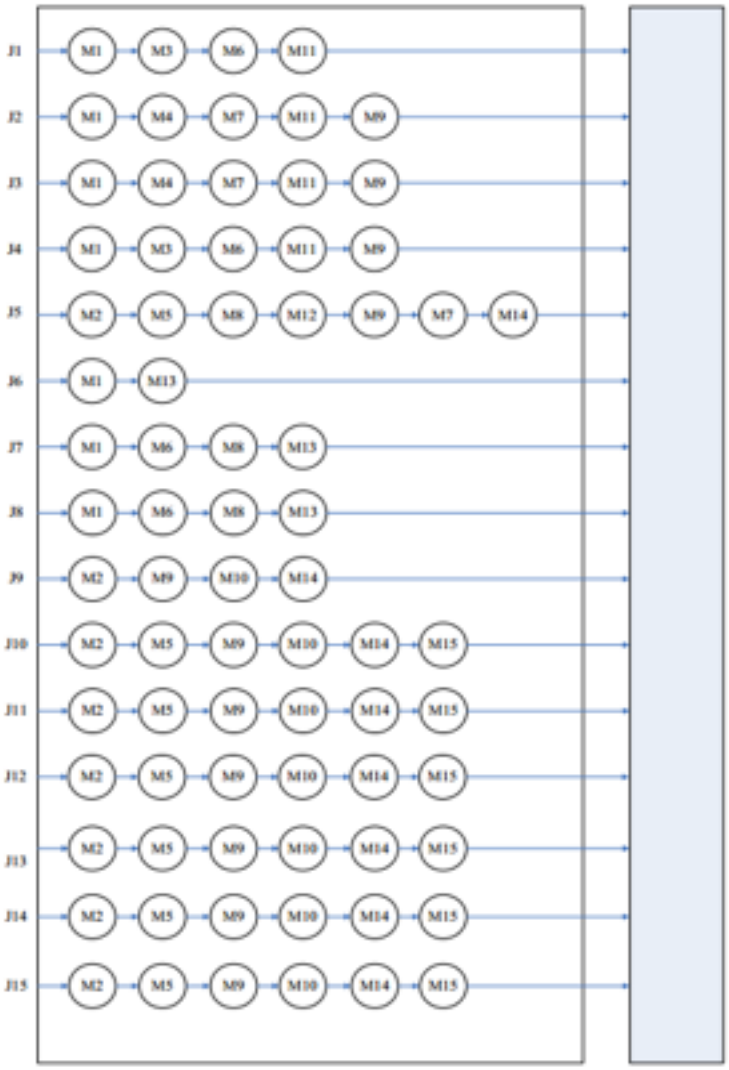


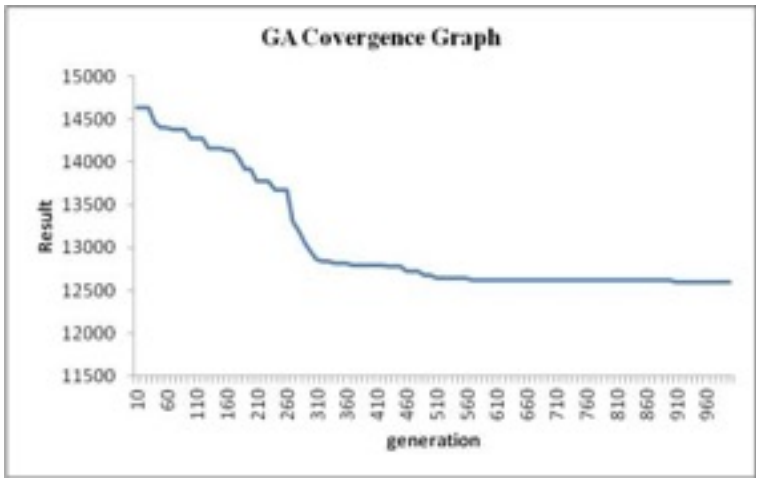
Fig. 3. Job sequence in each machine

B. Computational Study

TABLE II: GENETIC ALGORITHM RESULT

	Mean Flow Time	Makespan
Best	6155	17600
Average	6900.6	18027.41
Standard Deviation	528.8	634.211

It is stated before that the performance of a numerical optimizer may depend upon its parameter settings. The performance of the proposed GA may vary with its parameter values, like population size, cross over probability and mutation probability. Therefore, in order to analyze average performance of the GA, each of the instances is solved 20 times with different sets of such GA parameter values as mention in previous section. The population size is fixed 45 in different runs. The crossover probability is set to 55% and the mutation probability is set to 13%. We set the maximal generation in 2000 iterations. Then, the performance of the GA is evaluated in terms of standard deviation in objective values over 20 runs and average number of objective function required in obtaining the best solution of each run. The average objective values, the standard deviations, and the best result are given in Table II, respectively.



It is observed in Table II that the proposed GA could obtain the good solution for in every run (out of its 20 runs). However, the standard deviation is still considered large. Figure 4 shows the convergence point of the running problem. As stated above, such unavoidable variation in the performance of a metaheuristic is quite common. Since the objective of using the metaheuristic is to improve the company rule result, thus, we compare the result with the company actual result which performs using longest processing time approach.

C. Discussions

The optimum objective value obtained by GA for this problem is represent in Table III row 2, while the result for the actual scheduling is present in row 3. In this real case problem with 15 jobs, it is observed that the proposed GA could get a better result compared to the company actual schedule.

TABLE III: COMPARISON WITH DISPATCHING RULES

	Mean Flow Time	Makespan
Genetic Algorithm	6155	17600
Actual Condition-Longest processing time (LPT)	12840	20220
Shortest processing time (SPT)	7560	18020
First come first serve	8760	18080

Compared with the actual solution performed by LPT rule, the proposed GA produces the better solution. It also can be seen that GA outperforms the others procedure in term of solution quality. GA result can obtain 13% improvement of the makespan and 52% of mean flow time. Compare to the other two rules, GA have average gap for makespan is 3% and mean flow time is 33%. The average relative percentage gap obtained by dividing the difference between GA solution and dispatching rule solution by the best known solution based on LPT values.

VI. CONCLUSION

In this paper,we proposed genetic algorithm (GA) for solving the job shop scheduling problem,which asks for an arrangement of a sequence of job in certain machine. The proposed GA is investigated on a real case

problem,in which found to successful obtain the good solution value for the instances in every run (out of its 20 runs) compare to dispatching rules result.

REFERENCES

[1] D. J. Hoitomt, P. B. Luh and K. R. Pattipati, “A practical approach to job shop scheduling problems,” IEEE Trans. on Robotics and Automation, vol. 9, pp. 1-13, February 1993.

[2] T. Bäeck, D. B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, New York, Bristol: Oxford University Press, Institute of Physics, 1997, ch. F1.5.

[3] A. El-Bouri, N. Azizi, and S. Zolfaghari,“A comparative study of a new heuristic based on adaptive memory programming and simulated annealing: the case of job shop scheduling,” European Journal of Operational Research, vol.177, pp. 1894-1910, March 2007.

[4] J. P. Watson, A. Howe and L. Whitley, “Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem,” Computers & Operations Research, vol. 33, pp. 2623-2644, September 2006.

[5] S. Esquivel, S. Ferrero, R. Gallard, C. Salto, H. Alfonso, and M. Schtz, “Enhanced evolutionary algorithms for single and multiobjective optimization in the job shop scheduling problem,” Knowledge-Based Systems, vol.15, pp. 13-25, January 2002.

[6] Xia and Z. Wu, “A hybrid particle swarm optimization approach for the job-shop scheduling problem,” International Journal Advance Manufacturing Technology, vol. 29, pp. 360-366, April 2006.

[7] I. Moon, S. Lee,and H. Bae, “Genetic algorithms for job shop scheduling problems with alternative routings,” International Journal of Production Research, vol. 46, pp. 2695-2705, March 2008.

[8] R. Qing and Y. Wang, “A new hybrid genetic algorithm for job shop scheduling problem,” Computer &Operations Research, vol. 39, pp. 2291-2299, October 2012.

[9] S. French, Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop,New York,USA: John Willey & Sons Inc,1982.

[10] M. T. Jensen and T. K. Hansen, “Robust solutions to job shop problems,” in Proc. of Congress on Evolutionary Computation, Washington DC, 1999, pp 1138-1144.

[11] D. E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, New York, Addison Wesley, 1989, pp. 41