

Performance Evaluation of Solution-Based GA and Rule-Based GA for Scheduling Problems

Tadahiko Murata and Mitsuo Gen

Department of Industrial and Information Systems Engineering, Ashikaga Institute of Technology
Ashikaga 326-8558, Japan
murata@ashitech.ac.jp

Abstract

In this paper, we compare the performance of a solution-based schedule generation method and a rule-based schedule generation method, both of which are genetic-algorithm-based search algorithms. In the rule-based algorithm, a job permutation is determined by dispatching rules for scheduling problems. Therefore we call this algorithm a rule-based GA in this paper. We employ common dispatching rules such as SPT (Shortest Processing Time), EDD (Earliest Due Date), MST (Minimum Slack Time). We also employ a random rule, which select a job randomly from the remaining jobs. We employ a genetic algorithm to combine these dispatching rules to generate a schedule. On the other hand, a job permutation is directly treated as an individual in the solution-based schedule generation method using genetic algorithms. We apply the both genetic algorithms to flowshop scheduling problems with multiple machines in order to optimize an objective function. We employ one of three objectives in each experiment.

1 Introduction

Recently, genetic algorithms are successfully applied to a lot of optimization problems [1]. Manufacturing optimization has become a major application field for genetic algorithms. Recent report shows the surprisingly wide range of manufacturing problems covered by researchers using genetic algorithms [2].

In many studies, job permutations are directly coded as individuals in genetic algorithms [3], in which the feasibility of solutions is guaranteed by the application of

various specially designed genetic operators. We call this search method a solution-based genetic algorithm in this paper. On the other hand, we employed the sequence of dispatching rules as an individual in a rule-based genetic algorithm in [4]. The Shortest Processing Time (SPT) rule is a well-known dispatching rule, which can yield the optimum job permutation for minimizing the total flowtime in flowshop scheduling problems with a single machine. The Earliest Due Date (EDD) rule is also known as the optimum dispatching rule, by which the optimum job permutation can be obtained for minimizing the maximum tardiness in one-machine scheduling problems. The Minimum Slack Time (MST) rule is often employed as a dispatching rule to minimize the total tardiness. We combine the above three dispatching rules to generate a solution (i.e., a job permutation) using a genetic algorithm. Thus a sequence of dispatching rules is an individual in genetic algorithms. Since a set of solutions (i.e., job permutations) generated from sequences of the above three dispatching rules does not correspond to the universal set of job permutations. That is, any sequences of dispatching rules may not generate the optimal job permutation for scheduling problems. To cope with this difficulty, we introduced a simple heuristic dispatching rule. That is a random job selection (RND) rule.

In this paper, we compare the performance of these two genetic algorithms: one is the solution-based genetic algorithm and the other is the rule-based genetic algorithm. We apply these genetic algorithms to 10-job and 5-machine flowshop scheduling problems, 20-job and 10-machine flowshop scheduling problems, and 50-job and 10-machine flowshop scheduling problems. Furthermore we employ one of three functions to be optimized in genetic algorithms.

That is the makespan, the total flowtime, and the maximum tardiness. By computer simulations, we show the relation between the algorithms and the objectives.

2 Scheduling Problems

The research field of scheduling problems has a lot of kinds of problems. In this paper, we investigate problems of flowshop scheduling. Since Johnson show the optimal method for minimizing the makespan for the flowshop scheduling problems with two machines [5], a lot of researchers have tried to constructing an optimization method for scheduling problems. It is difficult, however, to find the optimal solution of a flowshop scheduling problem involving many jobs and machines (e.g., 100 jobs and 10 machines). Many approaches can be classified into two categories: optimization algorithms for the exact solution and heuristic algorithms for near optimal solutions. We try to find near optimum solutions using genetic algorithms in this paper.

We first briefly describe n -job and m -machine flowshop scheduling problems. General assumptions of flowshop scheduling problems can be written as follows (for details, see Daniels and Chambers [6]): Jobs are to be processed on multiple stages sequentially. There is one machine at each stage. Machines are available continuously. A job is processed on one machine at a time without preemption, and a machine processes no more than one job at a time. In this paper, we assume that n jobs are processed in the same order on m machines. This means that the aim of our flowshop scheduling is sequencing n jobs to optimize an objective function. Let the processing time and the completion time of job j on machine i be $t_P(i, j)$ and $t_C(i, j)$, respectively. The sequence of n jobs is denoted by an n -dimensional vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$, where x_k represents the k -th processing job. The completion time of each job corresponding to the sequence \mathbf{x} can be calculated as

$$t_C(1, x_1) = t_P(1, x_1), \quad (1)$$

$$t_C(i, x_1) = t_C(i-1, x_1) + t_P(i, x_1), \quad \text{for } i = 2, 3, \dots, m, \quad (2)$$

$$t_C(1, x_k) = t_C(1, x_{k-1}) + t_P(1, x_k), \quad \text{for } k = 2, 3, \dots, n, \quad (3)$$

$$t_C(i, x_k) = \max\{t_C(i-1, x_k), t_C(i, x_{k-1})\} + t_P(i, x_k), \quad \text{for } i = 2, 3, \dots, m; k = 2, 3, \dots, n. \quad (4)$$

Flowshop scheduling problems are to determine the sequence \mathbf{x} of n jobs based on a specific criterion. One of

the makespan, the total flowtime, and the maximum tardiness is often used as a scheduling criterion. Each of these criteria can be defined as follows in flowshop scheduling:

$$\text{Makespan: } f_1(\mathbf{x}) = t_C(m, x_n), \quad (5)$$

$$\text{Total flowtime: } f_2(\mathbf{x}) = \sum_{k=1}^n t_C(m, x_k), \quad (6)$$

Maximum tardiness:

$$f_3(\mathbf{x}) = \max\{\max\{t_C(m, x_k) - d(x_k), 0\} | k = 1, 2, \dots, n\}, \quad (7)$$

where $d(x_k)$ is the due date of the k -th processing job.

3 Schedule-Based Algorithm

The job sequence can be employed directly as an individual in genetic algorithms. We apply a solution-based genetic algorithm with a crossover and a mutation in Figure 1 and Figure 2. Figures show the individuals for 10-job problems. In Figure 1, two positions in Parent 1 are randomly selected and the jobs in the head and the tail parts are inherited according to the location in Parent 1. The remaining jobs are reordered in their appearance in Parent 2, and are placed in the middle part of the offspring. In Figure 2, the two positions are selected to shift jobs as a block.

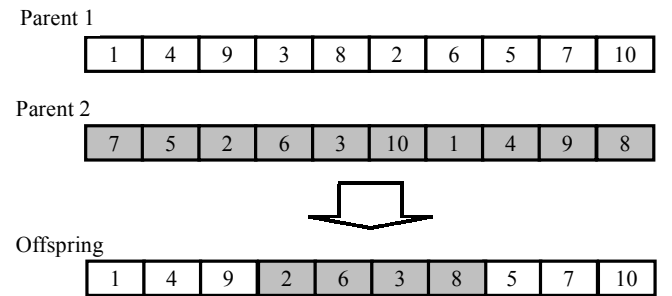


Figure 1 Crossover for Solution-Based GA.

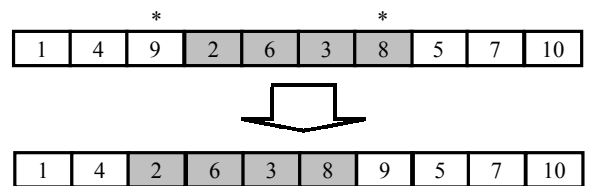


Figure 2 Mutation for Solution-Based GA.

4 Rule-Based Algorithm

In [4], we employed the sequence of dispatching rules, such as Shortest Processing Time (SPT) rule, Earliest Due

Date (EDD) rule, and Minimum Slack Time (MST) rule, as an individual in the rule-based genetic algorithm. Originally these dispatching rules are developed for single-machine scheduling problems. Therefore, we should extend these dispatching rules for multiple-machine scheduling problems.

(1) Shortest Processing Time (SPT)

This rule determines a job permutation in increasing order of job processing time. That is, the job with the minimum processing time is firstly selected from all jobs as the first job (i.e., x_1). Next, the job with the minimum processing time in the other jobs except the job selected at first. This procedure is repeated until all jobs are allocated in a job sequence. Since the job completion time varies according to both the completion time of preceding job and the completion time of the job in the preceding machine (see Equation (4)), we select a job with the minimum completion time from the set of remaining jobs as the k -th job in a job sequence:

$$x_k = j \text{ with } \min_{j \in R}(t_C(m, j)), \quad (8)$$

$$t_C(m, j) = \max\{t_C(m-1, j), t_C(m, x_{k-1})\} + t_P(m, j), \quad (9)$$

where R is the set of remaining jobs.

(2) Earliest Due Date (EDD)

This rule determines a job permutation in the increasing order of job due date. Since job due dates are not concerned with the number of machines in scheduling problems, we employ the original EDD rule for single-machine scheduling problems.

(3) Minimum Slack Time (MST)

In single-machine scheduling problems the slack time of the k -th job is determined as follows:

$$Slack(x_k) = d(x_k) - t_C(1, x_{k-1}) - t_P(1, x_k). \quad (10)$$

Since $t_C(1, x_k) = t_C(1, x_{k-1}) + t_P(1, x_k)$ (see Equation (3)) is the completion time of the k -th job in a single-machine scheduling problem, we define the slack time of the k -th job for multiple-machine scheduling problems as follows:

$$Slack(x_k) = d(x_k) - t_C(m, x_k). \quad (11)$$

We select a job with the minimum slack time from the set of remaining jobs as the k -th job in a job sequence:

$$x_k = j \text{ with } \min_{j \in R}(Slack(j)) = \min_{j \in R}(d(j) - t_C(m, j)). \quad (12)$$

We employ the sequence of these three rules as an individual (i.e., solution) in a rule-based algorithm.

When we employ the above rules as gene in each individual, the number of the first job in job permutations generated from rule sequences is the number of rules at most. That is, when we employ the above three rules for rule sequences, we can select a job as the first job in job permutations by one of SPT, EDD, MST rules. Therefore we can have only three different jobs processed in the beginning. This limitation may cause unreachableness to the optimum job permutation for scheduling problems. To avoid this limitation, we introduced the simple heuristic dispatching rule [4]. That is, a job is randomly selected from the set of remaining jobs.

(4) Random (RND)

The k -th job can be determined as follows:

$$x_k = \text{Random}(j \in R), \quad (13)$$

where $\text{Random}(j \in R)$ means a job number in the set of remaining jobs (i.e., R), which is randomly selected.

We also employ the above three rules and RND rule as gene in each solution for the rule-based GA.

We employ the following crossover and mutation operators in Figure 3 and Figure 4. Figures show the individuals for 10-job problems. Since the last job can be determined automatically, only 9 rules are required for sequencing 10 jobs. In Figure 3, some rules in Parent 1 are randomly selected and they are replaced by the jobs in the same location in Parent 2. In Figure 4, the selected rules are changed to the other rules randomly. Rules are selected by the mutation probability.

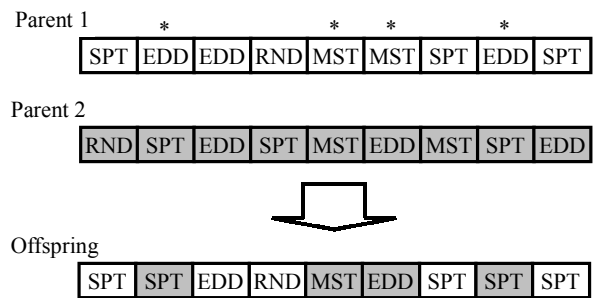


Figure 3 Crossover for Rule-Based GA.

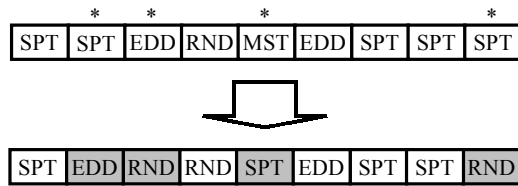


Figure 4 Mutation for Rule-Based GA.

5 Computer Simulation

We apply genetic algorithms to 10-job and 5-machine problems, 20-job and 10-machine problems, and 50-job and 10-machine problems. The processing time of each job at each machine was randomly specified as an integer in the closed interval $[1, 99]$, and the due date of each job is specified randomly. We generate 10 problems for each of three kinds of test problems.

First we apply the solution-based GA and the rule-based GA to the 10-job and 5-machine problems. Table 1 shows that average results obtained by the solution-based GA and the rule-based GA for the problems to minimize the makespan, or the total flowtime, or the maximum tardiness. Each underlined value in Table 1 shows the best results in the same objective. Since all objective values should be minimized, the performance of the solution-based GA is better than that of the rule-based GA for the 10-job and 5-machine problems with any objective.

We apply the genetic algorithms to the 20-job and 10-machine problems. Table 2 shows average results obtained by the algorithms. We can also observe that the solution-based GA has the better performance than the rule-based GA over all the objectives.

Finally, we apply the algorithms to the 50-job and 10-machine problems. Table 3 shows average results obtained by the algorithms. We can observe that the rule-based GA has the better results in the problem with the maximum tardiness.

Table 1 10-job and 5-machine problems.

GA	Makespan	T. Flow	M. Tard
Solution	<u>709.78</u>	<u>4534.28</u>	<u>10.90</u>
Rule	752.54	4636.37	11.26

Table 2 20-job and 10-machine problems.

GA	Makespan	T. Flow	M. Tard
Solution	<u>1544.63</u>	<u>20712.5</u>	<u>21.58</u>
Rule	1738.49	22310.1	34.81

Table 3 50-job and 10-machine problems.

GA	Makespan	T. Flow	M. Tard
Solution	<u>3126.37</u>	<u>90534.4</u>	217.61
Rule	3590.50	103732.6	<u>190.64</u>

6 Conclusion

In this paper, we compared the performance of the solution-based GA and the rule-based GA for flowshop scheduling problems. Almost all the problems, the solution-based GA has an advantage over the rule-based GA. This is because the solution-GA can search all the feasible solution directly. The rule-based GA, however, could find the better results for the problem with the maximum tardiness as an objective function.

References

1. D.E.Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA, Addison-Wesley, 1989.
2. C. Dimopoulos and A.M.S. Zalzal, "Evolutionary computation for manufacturing optimization: Recent developments," Research Report no.716, Dept. of Automatic Control & Systems Engineering, University of Sheffield.
3. T. Murata, H. Ishibuchi and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Computers and Industrial Engineering Journal*, Vol.30, No.4, pp.1061-1071, 1996.
4. T. Murata and M. Gen, "Combinations of Dispatching Rules for Scheduling Problems Using Genetic Algorithms," *Proc. of the Second Asia-Pacific Conference on Industrial Engineering and Management Systems* (Kanazawa, Japan, Oct. 30-31, 1999), pp.315-318, 1999.
5. S.M.Johnson: "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, Vol.1, No.1, pp.61-68, 1954.
6. R.L.Daniels, and R.J.Chambers: "Multiobjective flow-shop scheduling," *Naval Research Logistics*, Vol.37, pp.981-995, 1990.