Proposition of Selection Operation in a Genetic Algorithm for a Job Shop Rescheduling Problem

Hitoshi Iima

Kyoto Institute of Technology, Matsugasaki, Sakyo-ku, Kyoto 606–8585, Japan,
`iima@si.dj.kit.ac.jp`

Abstract. This paper deals with a two-objective rescheduling problem in a job shop for alteration of due date. One objective of this problem is to minimize the total tardiness, and the other is to minimize the difference of schedule. A genetic algorithm is proposed, and a new selection operation is particularly introduced to obtain the Pareto optimal solutions in the problem. At every generation in the proposed method, two solutions are picked up as the parents. While one of them is picked up from the population, the other is picked up from the archive solution set. Then, two solutions are selected from these parents and four children generated by means of the crossover and the mutation operation. The candidates selected are not only solutions close to the Pareto-optimal front but also solutions with a smaller value of the total tardiness, because the initial solutions are around the solution in which the total tardiness is zero. For this purpose, the solution space is ranked on the basis of the archive solutions. It is confirmed from the computational result that the proposed method outperforms other methods.

1 Introduction

Although many researchers have so far proposed various methods for solving scheduling problems in manufacturing systems, most studies deal with the determination of a schedule for conventional problems in which the condition is given in advance. However, in real manufacturing systems, alteration of problem condition such as change of due date and addition of job often obliges to revise a schedule worked out previously. Vieira et al. [1] presented definition appropriate for most applications of such rescheduling manufacturing systems, and reviewed various methods to solve the rescheduling problems. Moreover, genetic algorithms (GAs) [2] were also applied to some rescheduling problems [3–5]. The GA is an appropriate method to solve rescheduling problems, because diversity of population in the GA is useful in tracking change of problem condition [6].

The aim of most of these methods is to optimize only an original objective function, say the total tardiness. Consequently, the schedule obtained by such a method may be very different from that before the alteration. The difference of schedule incurs time and costs in re-preparing the processing for the case where the problem condition is altered after preparation of processing. As studies considering the schedule difference, Watatani and Fujii [7] defined a problem in which the objective function is a weighted sum of the makespan and the schedule difference, and applied the simulated annealing method to this problem. Abumaizar and Svestka [8] considered a problem for a breakdown of machine, and obtained a schedule with a small difference by rescheduling only the operations affected by the breakdown of machine.

This paper deals with a job shop rescheduling problem of minimizing both the total tardiness and the schedule difference in the case where the due dates of some jobs are altered, and a GA is proposed for obtaining the Pareto optimal solutions. Although a GA with a selection operation was applied to this problem [9] as a preliminary study, it is ineffective in instances with many jobs. In addition, the weight between the objective functions must be given in advance. In order to overcome these disadvantages, a new selection operation is proposed in this paper. In this operation, the solution space is ranked to select not only solutions close to the Pareto-optimal front but also solutions with a smaller value of the total tardiness.

The rest of the paper is organized as follows. Section 2 describes the rescheduling problem as well as the conventional scheduling problem before the alteration. Next, Section 3 presents the GA proposed for the rescheduling problem, and describes the new selection operation. The computational results can be found in Section 4, and the effectiveness of the proposed GA is investigated. Finally, Section 5 concludes the paper.

2 Problem Statement

2.1 Conventional Problem

At the beginning, a conventional problem P^* before alteration is described. In P^*, a set of I kinds of jobs J_i $(i = 1, 2, \cdots, I)$ is processed by using K machines M_k $(k = 1, 2, \cdots, K)$. A machine can process at most one job at a time. A job J_i should be completed by the due date D_i^*. Moreover, J_i consists of K operations O_{ij} $(j = 1, 2, \cdots, K)$. An operation O_{ij} is executed on $\mathrm{M}_{R(i,j)}$ $(R(i,j) \in \{1, 2, \cdots, K\})$, which is given in advance, and its processing time is given as PT_{ij}. No preemption of operation is allowed. There exists a precedence constraint between operations belonging to a job, and the operations must be executed in the order of j. This constraint is often called the technological constraint. The total number of operations is denoted as Q $(Q = IK)$.

The problem P^* is to determine the completion time c_{ij}^* of O_{ij} in such a way that the total tardiness F_1^* should be minimized. The objective function F_1^* is formulated as

$$F_1^* = \sum_{i=1}^{I} \max(c_{iK}^* - D_i^*,\ 0). \tag{1}$$

By applying a GA to P^*, the population at the final generation and the best solution (schedule) S^* can be obtained.

2.2 Rescheduling Problem

Consider the situation that the production has been prepared on the basis of the best schedule S^* obtained for the conventional problem P*. After the preparation, some due dates are altered. The due date of J_i after the alteration is denoted as D_i. Since S^* may not be optimal for D_i no longer, it should be revised. In this situation, the second objective function is defined as the magnitude of difference between S^* and a schedule S revised.

The rescheduling problem P coped with in this paper is to determine the completion time c_{ij} of O_{ij} in such a way that both the total tardiness F_1 and the schedule difference F_2 should be minimized. The objective functions in P are formulated as

$$\min \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} \displaystyle\sum_{i=1}^{I} \max(c_{iK} - D_i,\ 0) \\ \displaystyle\sum_{i=1}^{I}\sum_{j=1}^{K} |a_{ij} \cap a^*_{ij}| \end{pmatrix} \tag{2}$$

where a_{ij} and a^*_{ij} are the set of operations executed before O_{ij} on $\mathrm{M}_{R(i,j)}$ in S and that after O_{ij} on $\mathrm{M}_{R(i,j)}$ in S^*, respectively. Moreover, the symbol $|\ \ |$ means the number of elements. The schedule difference F_2 is given on the basis of Watatani's definition [7]. If there exists an operation belonging to both a_{ij} and a^*_{ij}, it means that the processing order of the operation and O_{ij} is reversed.

3 Application of Genetic Algorithm

GAs should be appropriately designed for respective optimization problems. In this section, a GA is designed for solving the rescheduling problem P. The population size and the final generation in this GA are denoted as N and G, respectively.

3.1 Individual Description

So far, many GAs have been proposed for solving job shop scheduling problems [10–12]. Recently Gen et al. [13], Ono et al. [14] and Shi et al. [15] proposed similar individual descriptions in which job numbers are sequenced, and obtained good solutions with less computational consumption.

The individual description proposed by Shi et al. is used in the proposed GA. This individual description is designed by utilizing the precedence constraint between operations effectively. The genotype is expressed by sequencing job numbers $\{i\}$ $(i = 1, 2, \cdots, I)$ K times. The length of the sequence is equal to the total number Q of operations. The genotype represents basically a production order, and the schedule of operation corresponding to a gene is determined in turn according to the decoding procedure. The operation is uniquely determined from the precedence constraint. The completion time c_{ij} of each operation O_{ij} is calculated from the genotype in the following way.

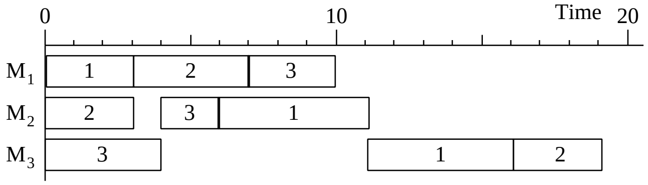


Fig. 1. Example of decoding genotype {213311223}

Table 1. Example of instance data (I=3,K=3)

	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
$R(i,j)$	1	2	3	2	1	3	3	2	1
PT_{ij}	3	5	5	3	4	3	4	2	3

Step 1 Set $m_k \leftarrow 0$ ($\forall k = 1, 2, \cdots, K$). The variable m_k represents the completion time of the operation to be executed last on the machine M_k.

Step 2 Set $j(i) \leftarrow 1$ ($\forall i = 1, 2, \cdots, I$). The variable $j(i)$ represents the operation number to be executed next in the job J_i.

Step 3 Set $\ell \leftarrow 1$. The variable ℓ represents the locus.

Step 4 The completion time of operation $O_{i_\ell j(i_\ell)}$ corresponding to the gene i_ℓ at the ℓ-th locus is given by

$$c_{i_\ell j(i_\ell)} = \max(m_{R(i_\ell, j(i_\ell))}, c_{i_\ell\ j(i_\ell)-1}) + PT_{i_\ell j(i_\ell)}. \tag{3}$$

Note that c_{i0}=0.

Step 5 Set $m_{R(i_\ell, j(i_\ell))} \leftarrow c_{i_\ell j(i_\ell)}$.

Step 6 If $j(i_\ell) < K$, set $j(i_\ell) \leftarrow j(i_\ell) + 1$.

Step 7 If $\ell = Q$, terminate this procedure. If not, set $\ell \leftarrow \ell + 1$ and return to Step 4.

The decoding procedure is explained by using an illustrative example. Figure 1 depicts the solution corresponding to genotype {213311223} in the instance shown by Table 1. At the beginning, the schedule of O_{21} corresponding to the first gene {2} is determined. Since $R(2,1)$=2 and PT_{21}=3, O_{21} is started at zero on M_2 and is completed at three. Next, the schedule of O_{11} and O_{31}, which correspond to the subsequence {13}, is determined in a similar way. Next, the fourth gene {3} is picked up. Since it is the second time this kind of gene is picked up ($j(3)$=2), the second operation O_{32} of J_3 corresponds to the gene. Since m_2=3 and c_{31}=4, O_{32} is started at four. Finally, the schedule of the remaining operations is determined in a similar way.

3.2 Initial Population

The initial population in GAs is generated randomly for optimization problems in general. Therefore the initial population is far from the Pareto optimal solutions,

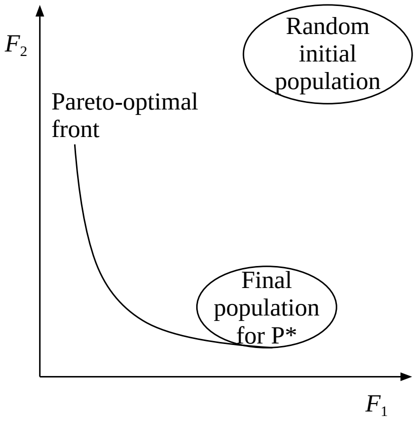


Fig. 2. Initial population of GA for a two-objective problem

and the population approaches them gradually by the exploration of GA, as shown in Fig. 2. On the other hand, the final population and the best schedule S^* obtained by applying a GA to the conventional problem P^* can be utilized as the initial population for P. Since the schedule difference F_2 is zero for $S = S^*$, one of the Pareto optimal solutions is already obtained for P. Moreover, the other solutions in the final population are also closer to S^* than the random initial population, as shown in Fig. 2. Therefore, the final population is used as the initial population in the proposed GA. However, the diversity for the final population is lost, because a GA has been performed for P^*. Thus, diverse initial solutions are generated by changing some solutions in the final population. These solutions changed are the non-dominated solutions in the final population in order to obtain a good solution set. The detail procedure to generate the initial population is as follows.

Step 1 Find the n' non-dominated solutions $y_1, y_2, \cdots, y_{n'}$ for P from the final population obtained for P^*, and add these solutions to the initial population for P. Set $n \leftarrow 1$.

Step 2 Generate a solution by shifting a gene selected randomly in y_n just before another gene selected randomly, and add the solution to the initial population.

Step 3 If the N initial solutions are generated, terminate this procedure.

Step 4 Set $n \leftarrow n + 1$. If $n > n'$, set $n \leftarrow 1$.

Step 5 Return to Step 2.

Rescheduling may be invoked during the actual processing of jobs. In this case, the schedule of operations completed at the time is unchanged. The schedule of operations which are being executed at the time is also unchanged. The genes associated with their operations are removed from the initial population, and the operations of GA are applied to the remaining genes.

$$\begin{array}{lcl} \mathrm{pa}^1 : \ 112321332 & \rightarrow & \mathrm{ch}^1 : \ 131221323 \\ \mathrm{pa}^2 : \ 321213123 & \rightarrow & \mathrm{ch}^2 : \ 213213132 \end{array}$$

Fig. 3. Example of SPX

3.3 Crossover and Mutation Operation

As the crossover operation, the set partition crossover (SPX) proposed by Shi et al. [15] is used. SPX is effective from the viewpoint of the computation time. The procedure of SPX is explained as follows. In the following explanation, pa^1_ℓ and pa^2_ℓ are defined as the gene at the ℓ-th locus in two parents pa^1 and pa^2, respectively, and ch^1_ℓ and ch^2_ℓ are defined as the gene at the ℓ-th locus in two children ch^1 and ch^2, respectively.

Step 1 Separate the set of all genes into two subsets SU_a and SU_b, which are nonempty and exclusive.
Step 2 Set $\ell \leftarrow 1$, $\ell_1 \leftarrow 1$ and $\ell_2 \leftarrow 1$.
Step 3 If $pa^1_\ell \in SU_a$, set $ch^1_{\ell_1} \leftarrow pa^1_\ell$ and $\ell_1 \leftarrow \ell_1 + 1$. Go to Step 5.
Step 4 Set $ch^2_{\ell_2} \leftarrow pa^1_\ell$ and $\ell_2 \leftarrow \ell_2 + 1$.
Step 5 If $pa^2_\ell \in SU_b$, set $ch^1_{\ell_1} \leftarrow pa^2_\ell$ and $\ell_1 \leftarrow \ell_1 + 1$. Go to Step 7.
Step 6 Set $ch^2_{\ell_2} \leftarrow pa^2_\ell$ and $\ell_2 \leftarrow \ell_2 + 1$.
Step 7 If $\ell = Q$, terminate this procedure. If not, set $\ell \leftarrow \ell + 1$ and return to Step 3.

Figure 3 depicts an example of SPX, where $SU_a = \{1, 2\}$ and $SU_b = \{3\}$. As shown in this figure, SPX preserves the order of genes in each subset.

In the mutation operation, two genes are selected randomly, and each of them is shifted just before another gene selected randomly.

3.4 Flow of GA

Most of selection operations for multiobjective optimization problems are based on the multi-objective GA (MOGA) [16] proposed by Fonseca and Fleming. In MOGA the rank of solution is given on the basis of the number of solutions which dominate itself, and solutions close to the Pareto-optimal front tend to be selected. In a GA with such a selection operation, solutions out of edge of incumbent population are not actively explored, and the exploration progresses in the direction of arrow (a) in Fig. 4. Even if this GA is applied to P, it is hard to obtain Pareto optimal solutions with a smaller F_1 and a larger F_2 because the initial solutions are around S^*. In order to obtain all the Pareto optimal solutions, the exploration must progress in the direction of not only arrow (a) but also arrow (b).

In this paper, a new selection operation, the selection by area ranking (SAR), is used to obtain diverse Pareto optimal solutions. In SAR, solutions are selected by using the solution space ranked on the basis of the archive solution set X, which is the non-dominated solution set in the solutions explored by the incumbent generation. The solution space is ranked as follows.

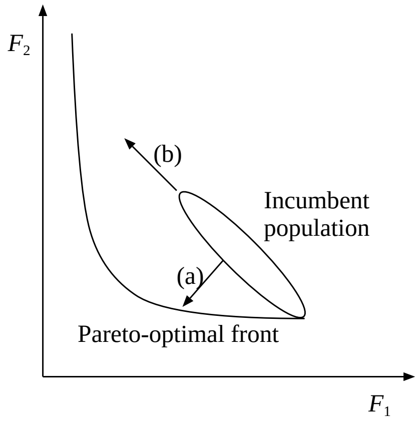


Fig. 4. Direction of exploration

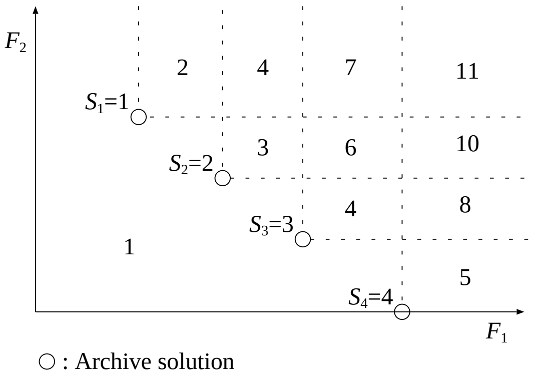


Fig. 5. Example of area ranking

Step 1 Find the area in which all the solutions are not dominated by any solution of X, and set the rank of the area to one.
Step 2 Number all the solutions of X in the order of F_1, and set the score S_m of m-th solution to m ($S_m = m$).
Step 3 For the area in which the solution x is dominated by at least one archive solution of X, and set the rank of the area to $\sum_{m' \in X_{\mathrm{sub}}} S_{m'} + 1$, where X_{sub} is the set of the archive solution numbers dominating x.

A solution with a small rank is close to the Pareto-optimal front, or has a small F_1. An example of area ranking is shown in Fig. 5 for $|X| = 4$.

At every generation in the proposed GA, two solutions are picked up as the parents. While one of them is picked up randomly from the population, the other is picked up randomly from X in order to explore an area around X intensively.

The former solution is removed from the population. Next, four children are generated from the parents by means of the crossover and the mutation operation. Next, the two solutions with the smallest ranks are selected from the two parents and the four children, and then are added to the population. If the candidates for the second solution selected are plural, the solution with the minimum value of F_1 is selected from these candidates. In this selection, the population size increases, because only a single solution is picked up as a parent from the population. In order to prevent this, another solution selected randomly is removed from the population before the selection.

The flow of the proposed GA is as follows.

Step 1 Generate the initial population, and set $g \leftarrow 1$. The variable g represents the generation.
Step 2 Determine the archive solution set X from the initial population.
Step 3 As the parents, pick up a solution x_1 randomly from the population and a solution x_2 randomly from X. Remove x_1 from the population.
Step 4 Generate two solutions x_3 and x_4 by means of the crossover operation. The crossover operation is applied at every generation.
Step 5 Generate a solution x_5 from x_1 by means of the mutation operation. Similarly, generate a solution x_6 from x_2 by means of the mutation operation. The mutation operation is also applied at every generation.
Step 6 Remove a solution selected randomly from the population.
Step 7 Select two solutions from the solutions $\{x_1, x_2, \cdots, x_6\}$ by means of SAR, and add them to the population.
Step 8 Update X from $X \cup \{x_3, x_4, x_5, x_6\}$.
Step 9 If $g = G$, terminate this algorithm and output X as the answer. If not, set $g \leftarrow g + 1$ and return to Step 3.

The archive solution set X has no size limitation, because the size remains relatively small for this problem.

4 Computational Result

The proposed GA is applied to forty instances in order to evaluate its effectiveness.

4.1 Instance Data

The forty instances for the conventional problem P* are given by revising the benchmark instances la01–la40 [17]. Of the instance data, the machine number $R(i, j)$ and the processing time PT_{ij} of operation O_{ij} are the same as those of benchmark instance. The due date D_i^* of each job J_i is given by the total processing time $\sum_{j=1}^{K} PT_{ij}$ times a constant d. The constant d for each instance is given in such a way that the total tardiness F_1^* obtained by applying a GA to this instance is less than one hundred. The constant d, the number I of jobs,

Table 2. Instance data

Instance	d	I	K	Q	Instance	d	I	K	Q
1	2.10	10	5	50	21	1.85	15	10	150
2	2.00				22	2.00			
3	2.10				23	1.80			
4	2.20				24	1.85			
5	2.10				25	1.90			
6	3.00	15	5	75	26	2.30	20	10	200
7	2.90				27	2.30			
8	2.80				28	2.30			
9	3.00				29	2.40			
10	2.90				30	2.40			
11	3.40	20	5	100	31	3.20	30	10	300
12	3.40				32	3.10			
13	3.30				33	3.30			
14	3.80				34	3.17			
15	3.70				35	3.35			
16	1.50	10	10	100	36	1.65	15	15	225
17	1.60				37	1.65			
18	1.55				38	1.63			
19	1.55				39	1.60			
20	1.50				40	1.59			

the number K of machines and the total number Q of operations are shown in Table 2.

A GA has been applied to these instances, and the final population and the best schedule S^* have been obtained. In order to generate the instances for the rescheduling problem P, the due date is altered for a few jobs on the basis of S^*. These jobs are selected randomly from the jobs such that the tardiness is zero in S^*. The number of the jobs selected is nearly ten percent of I. The due date D_i of each of the jobs $\{\mathrm{J}_i\}$ is altered to $0.9c_i^*$, where c_i^* is the completion time for S^*. Hence, the jobs become tardy for $S = S^*$.

4.2 Comparison among Procedures to Pick Up the Parents

In the proposed GA (called PGA), one of parents is picked up from the population, and the other is picked up from the archive solution set X. In order to evaluate the effectiveness of this procedure, PGA is compared with the following methods.

· GA-P : Both parents are picked up from the population.
· GA-X : Both parents are picked up from X.

In these GAs, there are two parameters: the population size N and the final generation G. The values of these parameters are decided through a preliminary calculation. The value of N is set to 500, and the value of G is set as shown in

Table 3. Final generation G

Instance	PGA,GA-P,GA-X,GA-M	SPEA2
1–10	10000	30
11–20	50000	40
21–30	100000	50
31–40	150000	60

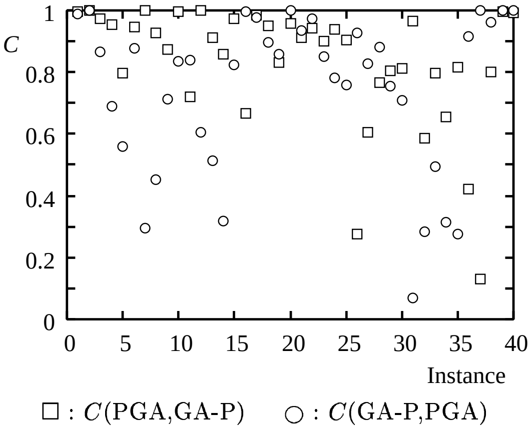


Fig. 6. Comparison between PGA and GA-P

Table 3. Each GA is performed one hundred times with various random seeds for an instance.

The GAs are evaluated by the coverage metric [18] which means relation of domination between the solution sets obtained by two methods. The coverage metric of Method 1 (M1) to Method 2 (M2) is defined by

$$C(\mathrm{M1},\mathrm{M2}) = \frac{|\{x_2 \in X_2; \exists x_1 \in X_1 : x_1 \succeq x_2\}|}{|X_2|} \tag{4}$$

where X_1 and X_2 are the solution sets obtained by M1 and M2, respectively. Moreover, $x_1 \succeq x_2$ means that Solution x_1 dominates Solution x_2 or has the same objective values as x_2.

Figure 6 shows the average coverage metric between PGA and GA-P. As shown in this figure, PGA is better than GA-P, because C(PGA,GA-P) > C(GA-P,PGA) in many instances. In particular, PGA is much better in Instances 1–10 and 31–35. On the other hand, PGA is worse in Instances 36–40.

Figure 7 shows the average coverage metric between PGA and GA-X. It is found from this figure that PGA is better than GA-X. Since the parents are picked up from X in GA-X, the population in fact is not used for the exploration. The number of solutions of X is small in early generations. Therefore, the same parents are picked up in the early generations, and similar children tend to be

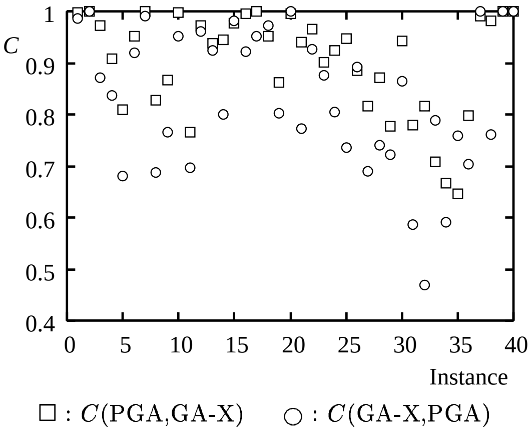


Fig. 7. Comparison between PGA and GA-X

generated from them. For this reason, solutions in the solution space can not be explored sufficiently, and a good solution set is not obtained by GA-X.

4.3 Comparison with Other Methods

In order to evaluate the effectiveness of PGA, it is compared with the following methods.

· GA with the minimal generation gap for rescheduling problems (GA-M) [9]

Two solutions are selected from two parents and four children in this GA as well as PGA. The first solution selected is one with the smallest value of the total tardiness F_1 in order to obtain Pareto optimal solutions with a smaller value of F_1. The second solution selected is the closest one to the archive solution x' with the smallest value of F_1 in X. The distance d_n between x' and each of six solutions x_n $(n = 1, 2, \cdots, 6)$ is defined by using the weight w as

$$d_n = \sqrt{(F_{1n} - F_1')^2 + w^2(F_{2n} - F_2')^2} \quad (n = 1, 2, \cdots, 6) \tag{5}$$

where F_{pn} and F_p' $(p = 1, 2)$ are the objective value F_p for x_n and x', respectively. The value of w must be given in advance, and is set to ten in consideration to the balance between F_1 and F_2.

· Strength Pareto evolutionary algorithm 2 (SPEA2) [19]

SPEA2 is a powerful MOGA-based method for conventional multiobjective problems. In this method solutions close to the Pareto-optimal front tend to be selected. Moreover, the distance between two solutions is calculated, and solutions far from others tend to be selected. SPEA2 is used as one of typical methods in the literature. SPEA2 includes no factor to explore actively an area of smaller F_1. It may be not fair to compare SPEA2 and PGA in this sense. The area of smaller F_1 can be explored by adding a weighted F_1 to the original

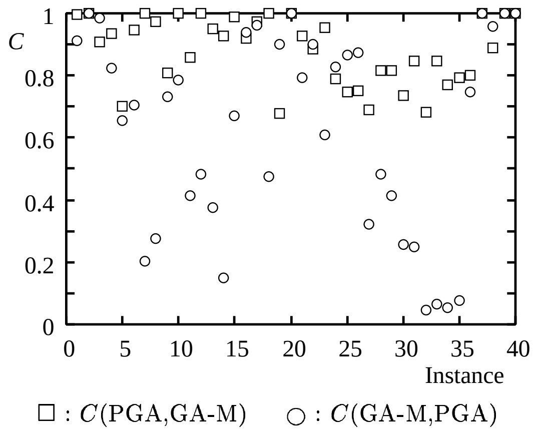


Fig. 8. Comparison between PGA and GA-M

fitness function. The new fitness function ff^{S} in SPEA2 is given by

$$ff^{\mathrm{S}} = ff + w^{\mathrm{S}} F_1 \tag{6}$$

where ff is the original fitness function. The parameter w^{S} is the weight and its value is set to ten through a preliminary calculation. Note that a solution with a lower fitness value is better in SPEA2. By using ff^{S}, solutions with a smaller F_1 tend to be selected.

The population size N in GA-M and SPEA2 is as large as that in PGA. The final generation G is given in such a way that the computation time is the same, and is shown in Table 3. SPEA2 is a generational GA, and the distance between solutions in the population is calculated at every generation. Therefore, G in SPEA2 is smaller than those in the other GAs.

Figure 8 depicts the average coverage metric between PGA and GA-M. It is found from this figure that PGA is better than GA-M. In particular, PGA is much better than GA-M for Instances 11–15 and 26–35 with $I \geq 20$. Figure 9 depicts the average coverage metric between PGA and SPEA2. It is found from this figure that PGA is better than SPEA2 for almost all instances. Since the value of G is relatively small in SPEA2, the search process may not converge still for the value given. PGA is more applicable than SPEA2, because a solution set should be obtained in short time for rescheduling problems.

Next, typical solution sets obtained by PGA, GA-M and SPEA2 are shown in Fig. 10 for Instance 8. These are results in a single trial, and each computation time is about 0.7 second by using a 2.4 GHz, Pentium IV PC, running under LINUX. As shown in this figure, the solutions obtained by PGA are closer to the Pareto-optimal front, and are distributed widely. A decision maker can confirm the total tardiness and the schedule difference of each schedule by the figure, and determine a desirable schedule in accordance with the situation at that time. The solutions obtained by GA-M and SPEA2 in $F_1 > 90$ are the same as those by

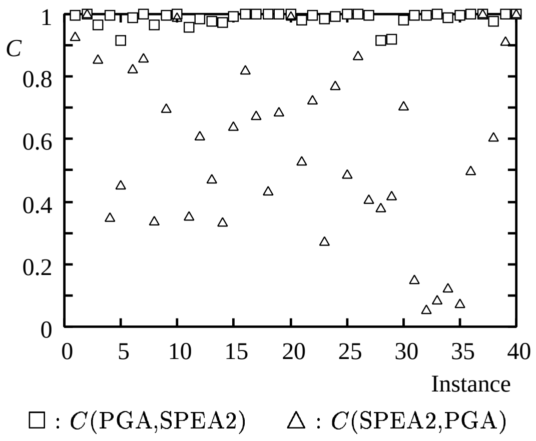


Fig. 9. Comparison between PGA and SPEA2

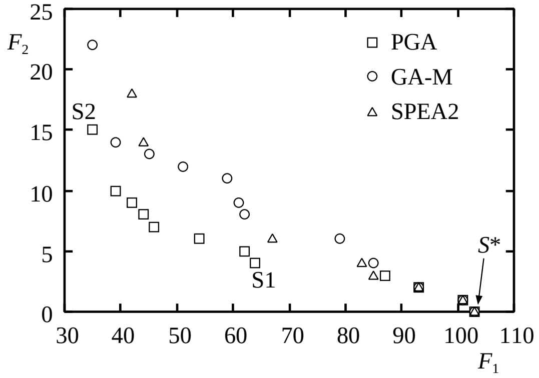
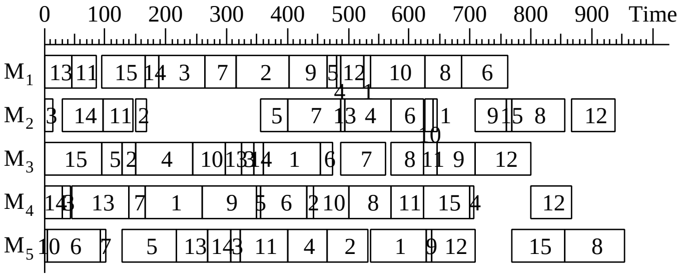


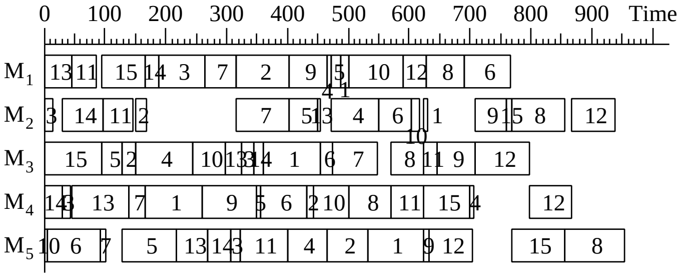
Fig. 10. Solution set obtained for Instance 8

PGA. Although the solutions obtained by GA-M in $F_1 < 90$ are distributed widely, they are far from the Pareto-optimal front. In SPEA2 few solutions are obtained in $F_1 < 80$.

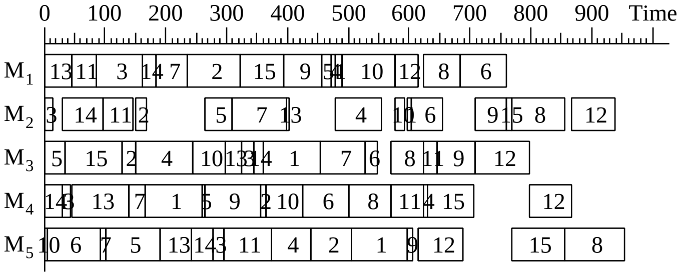
Finally, Fig. 11 depicts the schedules for S^*, S1 and S2 in Fig. 10. In Instance 8 the due date of J_1 becomes earlier. In order to process J_1 earlier, the processing order $\{\mathrm{J}_5\mathrm{J}_4\mathrm{J}_{12}\mathrm{J}_1\mathrm{J}_{10}\}$ on M_1 in S^* is changed to $\{\mathrm{J}_4\mathrm{J}_5\mathrm{J}_1\mathrm{J}_{10}\mathrm{J}_{12}\}$ in S1. Consequently, J_1 on M_5 and M_2 is also processed earlier, and the completion time c_1 of J_1 becomes earlier. While S1 is similar to S^*, S2 is relatively different from S^*. In order to complete J_1 much earlier, the idle time at about time 100 on M_1 and M_5 in S1 is lost in S2.



(a) S^*



(b) S1



(c) S2

Fig. 11. Schedules obtained for Instance 8

5 Conclusion

This paper has dealt with a two-objective rescheduling problem after alteration of due date in a job shop. The aim of this problem is to minimize the schedule difference as well as the total tardiness. A GA has been proposed for obtaining the Pareto optimal solutions in the problem. In particular, a new selection operation by the area ranking has been proposed. In this operation the candidates selected are not only solutions close to the Pareto-optimal front but also solutions with a smaller value of the total tardiness. It is concluded from the computational result that solutions obtained by the proposed GA are closer to the Pareto-optimal front than those by other GAs.

References

1. Vieira, G.E., Herrmann, J.W., Lin, E.: Rescheduling manufacturing systems: A framework of strategies, policies and methods. Journal of Scheduling, **6** (2003) 39–62
2. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison–Wesley (1989)
3. Lin, S., Goodman, E.D., Punch, W.F.III: A genetic algorithm approach to dynamic job shop scheduling problems. Proceedings of 7th International Conference on Genetic Algorithms (1997) 481–488
4. Bierwirth, C., Mattfeld, D.C.: Production scheduling and rescheduling with genetic algorithms. IEEE Transactions on Evolutionary Computation **7** (1999) 1–17
5. Branke, J., Mattfeld, D.C.: Anticipation in dynamic optimization: The scheduling case. Proceedings of Parallel Problem Solving from Nature VI (2000) 253–262
6. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Norwell (2002)
7. Watatani, Y., Fujii, S.: A study on rescheduling policy in production system. JAPAN/USA Symposium on Flexible Automation **2** (1992) 1147–1150
8. Abumaizar, R.J., Svestka, J.A.: Rescheduling job shops under random disruptions. International Journal of Production Research **35** (1997) 2065–2082
9. Iima, H., Nakase R., Sannomiya, N.: Genetic algorithm approach to a multiobjective rescheduling problem in a job shop, Proceedings of 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (2003) 422–437
10. Davis, L.: Job shop scheduling with genetic algorithms. Proceedings of First International Conference on Genetic Algorithms (1985) 136–140
11. Nakano, R.: Conventional genetic algorithm for job shop problems. Proceedings of Fourth International Conference on Genetic Algorithms (1991) 474–479
12. Fang, H.L., Ross, P., Corne, D.: A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. Proceedings of Fifth International Conference on Genetic Algorithms (1993) 375–382
13. Gen, M., Cheng, R.: Genetic Algorithms & Engineering Design. John Wiley & Sons, Inc. (1997)
14. Ono, I., Yamamura, M., Kobayashi, S.: A genetic algorithm for job-shop scheduling problems using job-based order crossover. Proceedings of 1996 IEEE International Conference on Evolutionary Computation (1996) 547–552
15. Shi, G., Iima, H., Sannomiya, N.: A new encoding scheme for solving job shop problems by genetic algorithm. Proceedings of 35th IEEE Conference on Decision and Control (1996) 4395–4400
16. Fonseca C.C., Fleming P.J.: Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. Proceedings of 5th International Conference on Genetic Algorithms (1993) 416–423
17. Lawrence S.: Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement). Graduate School of Industrial Administration, Carnegie-Mellon University (1984)
18. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation **3** (1999) 257–271
19. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the performance of the strength Pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Communication Networks Lab, Swiss Federal Institute of Technology (2001)