Pergamon

Computers ind. Engng Vol. 30, No. 4, pp. 957–968, 1996
Copyright © 1996 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
S0360-8352(96)00045-9
0360-8352/96 $15.00 + 0.00

MULTI-OBJECTIVE GENETIC ALGORITHM AND ITS APPLICATIONS TO FLOWSHOP SCHEDULING

TADAHIKO MURATA, HISAO ISHIBUCHI and HIDEO TANAKA

Department of Industrial Engineering, Osaka Prefecture University, Gakuen-cho 1-1, Sakai, Osaka 593, Japan

Abstract—In this paper, we propose a multi-objective genetic algorithm and apply it to flowshop scheduling. The characteristic features of our algorithm are its selection procedure and elite preserve strategy. The selection procedure in our multi-objective genetic algorithm selects individuals for a crossover operation based on a weighted sum of multiple objective functions with variable weights. The elite preserve strategy in our algorithm uses multiple elite solutions instead of a single elite solution. That is, a certain number of individuals are selected from a tentative set of Pareto optimal solutions and inherited to the next generation as elite individuals. In order to show that our approach can handle multi-objective optimization problems with concave Pareto fronts, we apply the proposed genetic algorithm to a two-objective function optimization problem with a concave Pareto front. Last, the performance of our multi-objective genetic algorithm is examined by applying it to the flowshop scheduling problem with two objectives: to minimize the makespan and to minimize the total tardiness. We also apply our algorithm to the flowshop scheduling problem with three objectives: to minimize the makespan, to minimize the total tardiness, and to minimize the total flowtime. Copyright © 1996 Elsevier Science Ltd

1. INTRODUCTION

Flowshop scheduling problems are one of the most well known problems in the area of scheduling. The objective of minimizing the makespan is often employed as a criterion of flowshop scheduling since Johnson's work [1]. Various heuristic approaches (e.g., Dannenbring [2], Nawaz *et al.* [3], Osman and Potts [4] and Widmer and Hertz [5]) as well as optimization techniques (e.g., Ignall and Schrage [6] and Lomnicki [7]) have been proposed for minimizing the makespan. While these studies treated a single objective, many real-world problems involve multiple objectives. Recently several researchers have tackled multi-objective flowshop scheduling problems. For example, Ho and Chan [8] proposed a heuristic method for flowshop scheduling with bicriteria, Gangadharan *et al.* [9] proposed a simulated annealing heuristic for flowshop scheduling with bicriteria, and Morizawa *et al.* [10] proposed a complex random sampling method for multi-objective problems.

Genetic algorithms have been mainly applied to single-objective optimization problems. When we apply a single-objective genetic algorithm to a multi-objective optimization problem, multiple objective functions should be combined into a scalar fitness function. If we assign a constant weight to each of the multiple objective functions for combining them, the direction of search in the genetic algorithm is constant in the multi-dimensional objective space as shown in Fig. 1. In Fig. 1, $f_1(\cdot)$ is an objective function to be maximized and $f_2(\cdot)$ is to be minimized. The closed circle in Fig. 1 represents the final solution by the single-objective genetic algorithm.

Some studies have been attempted for designing multi-objective genetic algorithms since Schaffer's work [11]. Schaffer proposed the Vector Evaluated Genetic Algorithm (VEGA) for finding Pareto optimal solutions of multi-objective optimization problems. In Schaffer's VEGA, a population was divided into disjoint subpopulations, then each subpopulation was governed by its own objective function. Although Schaffer [11] reported some successful results, his approach seems to be able to find only extreme solutions on Pareto fronts as shown in Fig. 2 because its search directions are parallel to the axes of the objective space. Schaffer suggested two approaches to improve his approach in his paper [11]. One is to provide a heuristic selection preference for non-dominated individuals in each generation. The other is to crossbreed among the "species" by adding some mate selection.

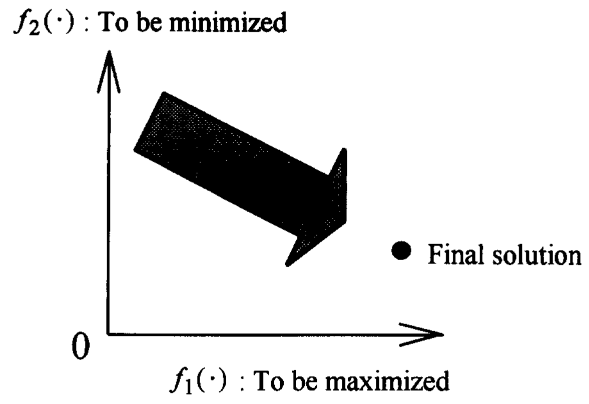

$f_2(\cdot)$: To be minimized
Final solution
0
$f_1(\cdot)$: To be maximized

Fig. 1. Direction of the search by GA with a combined fitness function.

The point of multi-objective optimization problems is how to find all possible tradeoffs among multiple objective functions that are usually conflicting. Since it is difficult to choose a single solution for a multi-objective optimization problem without iterative interaction with the decision maker, one general approach is to show the set of Pareto optimal solutions to the decision maker. Then the decision maker can choose any one of the Pareto optimal solutions. To find out all the Pareto optimal solutions by genetic algorithms, the variety of individuals should be kept in each generation. Recently Gen *et al.* [12] proposed a genetic algorithm for solving a bicriteria transportation problem, Tamaki *et al.* [13] proposed a genetic algorithm for scheduling problems with multi-criteria, and Horn *et al.* [14] proposed the Niched Pareto Genetic Algorithm by incorporating the concept of Pareto domination in the selection procedure and applying a niching pressure to spend the population out along Pareto fronts.

In this paper, we propose a multi-objective genetic algorithm with various search directions as shown in Fig. 3. There are two characteristic features of our algorithm. One is its selection procedure. In the selection procedure, our multi-objective genetic algorithm uses a weighted sum of multiple objective functions to combine them into a scalar fitness function. The weights attached to the multiple objective functions are not constant but randomly specified for each selection. Therefore the direction of the search in our multi-objective genetic algorithm is not constant. The other characteristic feature of our algorithm is its elite preserve strategy. A tentative set of Pareto optimal solutions is preserved in the execution of our multi-objective genetic algorithm. A certain number of individuals in this set are inherited to the next generation as elite individuals. In order to show that our approach can handle multi-objective optimization problems with concave Pareto fronts, we apply our proposed genetic algorithm to a two-objective function optimization problem with concave Pareto fronts. The performance of our multi-objective genetic algorithm is examined by applying it to the flowshop scheduling problem with two objectives: to minimize the makespan and to minimize the total tardiness. We also apply our algorithm to the flowshop scheduling

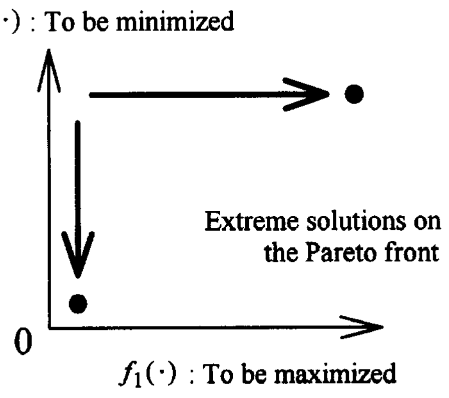

$f_2(\cdot)$: To be minimized
Extreme solutions on the Pareto front
0
$f_1(\cdot)$: To be maximized

Fig. 2. Directions of the search by Schaffer's VEGA.

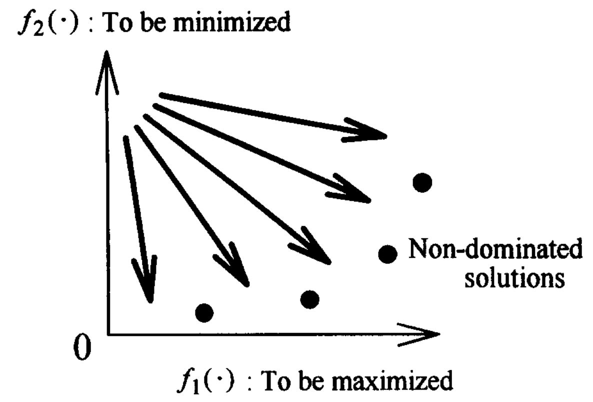

$f_2(\cdot)$: To be minimized
0
$f_1(\cdot)$: To be maximized
Non-dominated solutions

Fig. 3. Directions of the search by our multi-objective genetic algorithm.

problem with three objectives: to minimize the makespan, to minimize the total tardiness, and to minimize the total flowtime.

2. GENETIC OPERATIONS IN MULTI-OBJECTIVE GENETIC ALGORITHM

2.1. Selection procedure

One of the simplest methods to combine multiple objective functions into a scalar fitness solution is the following weighted sum approach (we assume that all the objective functions should be maximized):

$$f(\mathbf{x}) = w_1 \cdot f_1(\mathbf{x}) + \cdots + w_i \cdot f_i(\mathbf{x}) + \cdots + w_n \cdot f_n(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is a string (i.e., solution), $f(\mathbf{x})$ is a combined fitness function, $f_i(\mathbf{x})$ is the i-th objective function, w_i is a constant weight for $f_i(\mathbf{x})$, and n is the number of the objective functions.

If we use the weighted sum in (1) with the constant weights w_i's, the search direction in genetic algorithms is also constant as shown in Fig. 1. Therefore we propose an idea to use variable weights in the selection procedure. The aim of this idea is to realize various search directions in Fig. 3 to search for Pareto optimal solutions. For example, we can specify variable weights as follows for a two-objective optimization problem:

$$f(\mathbf{x}) = w_1 \cdot f_1(\mathbf{x}) + w_2 \cdot f_2(\mathbf{x}), \tag{2}$$

$$\begin{cases} w_1 = (i-1)/(N_{\text{selection}} - 1), & i = 1, 2, \ldots, N_{\text{selection}}, \\ w_2 = 1 - w_1, \end{cases} \tag{3}$$

where $N_{\text{selection}}$ is the number of selections in each generation. The weighting scheme in (2)–(3) means that different weights are used for selecting each pair of parent strings in our multi-objective genetic algorithm. Because $N_{\text{selection}}$ pairs of parent strings are to be selected in each generation, $N_{\text{selection}}$ pairs of different weights are specified in (3). In (3), the values of w_1 and w_2 are evenly distributed over the closed interval [0, 1]. In computer simulations in this paper, we used the weighting scheme in (2)–(3) for two-objective problems.

In general, we can randomly determine the value of each weight. For a multi-objective optimization problem with n objective functions ($n \geqslant 2$), we can assign a random real number to each weight as follows when each pair of parent strings are selected for a crossover operation.

$$w_i = \frac{rnd_i}{\sum_{j=1}^{n} rnd_j}, \quad i = 1, 2, \ldots, n, \tag{4}$$

where rnd_i and rnd_j are non-negative random integers (or non-negative random real numbers). From (4), we can see that n random real numbers are generated for the weights w_i's to calculate the weighted sum in (1) when each pair of parent strings are selected. This procedure is iterated $N_{\text{selection}}$ times in each generation for selecting $N_{\text{selection}}$ pairs of parent strings for a crossover operation. In computer simulations for a three-objective problem, we used the weighting scheme in (1) and (4).

The weighted sum $f(\mathbf{x})$ in (1) and (2) is used for determining the selection probability of each string in our multi-objective genetic algorithm. Because the weights w_i's are not constant but variable, the selection probability of each string is also variable even in a single generation. This realizes various search directions in our multi-objective genetic algorithm.

2.2. *Elite preserve strategy*

In multi-objective optimization problems, a solution with the best value of each objective can be regarded as an elite individual. Therefore we have n elite individuals for an n-objective problem. It is natural to think that such solutions are to be preserved to the next generation in genetic algorithms.

During the execution of our multi-objective genetic algorithm, a tentative set of Pareto optimal solutions is stored and updated at every generation. We also preserve a certain number of individuals randomly selected from the tentative set of Pareto optimal solutions in addition to the n elite individuals with respect to n objectives. That is, multiple Pareto optimal solutions are used as elite individuals in our multi-objective genetic algorithm. This elite preserve strategy has an effect on keeping the variety in each population in our multi-objective genetic algorithm.

In computer simulation, three solutions were selected as elite individuals for a two-objective flowshop scheduling problem: two elite individuals with respect to two objectives and a randomly selected individual from the tentative set of Pareto optimal solution. As a variant, we can design a multi-objective genetic algorithm by randomly selecting a certain number of elite individuals from the tentative set of Pareto optimal solutions. In computer simulation for a three-objective flowshop scheduling problem, three solutions which were randomly selected from the tentative set of Pareto optimal solutions were inherited to the next generation.

2.3. *Algorithm*

The following genetic operations are employed to generate and handle a population (i.e., a set of strings) in our multi-objective genetic algorithm (see Fig. 4).

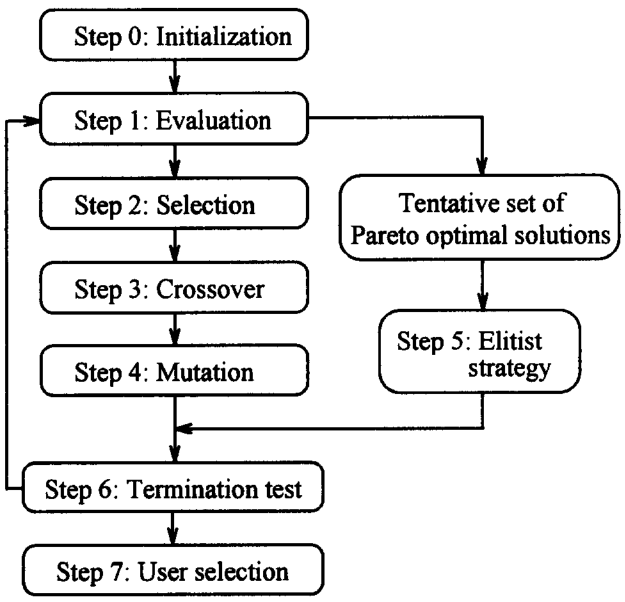


Fig. 4. Outline of our multi-objective genetic algorithm.

Step 0 (Initialization): Generate an initial population containing N_{pop} strings where N_{pop} is the number of strings in each population.
Step 1 (Evaluation): Calculate the values of the objective functions for the generated strings. Update a tentative set of Pareto optimal solutions.
Step 2 (Selection): Calculate the fitness value $f(\mathbf{x})$ of each string by using the weights defined as (3) or (4). Select a pair of strings from the current population according to the following selection probability. The selection probability $P(\mathbf{x})$ of a string $\mathbf{x}$ in a population Ψ is specified as

$$P(\mathbf{x}) = \frac{f(\mathbf{x}) - f_{\min}(\Psi)}{\sum_{\mathbf{x}\in\Psi} \{f(\mathbf{x}) - f_{\min}(\Psi)\}}, \tag{5}$$

where

$$f_{\min}(\Psi) = \min\{f(\mathbf{x})|\mathbf{x}\in\Psi\}. \tag{6}$$

This step is repeated $N_{\text{selection}}$ times to produce N_{pop} offspring by the crossover operation in Step 3. The values of the weights w_i's in (3) or (4) are specified at each of the $N_{\text{selection}}$ iterations.
Step 3 (Crossover): For each selected pair, apply a crossover operation to generate an offspring with the crossover probability P_c. N_{pop} strings should be generated by the crossover operation ($N_{\text{selection}} = N_{\text{pop}}$ if a single offspring is generated from a pair of parent strings in the crossover operation).
Step 4 (Mutation): For each string generated by the crossover operation, apply a mutation operation with a prespecified mutation probability P_m.
Step 5 (Elitist strategy): Randomly remove N_{elite} strings from the N_{pop} strings generated by the above operations, and add the same number of strings from a tentative set of Pareto optimal solutions to the current population.
Step 6: (Termination test): If a prespecified stopping condition is not satisfied, return to Step 1.
Step 7 (User selection): The multi-objective genetic algorithm shows the final set of Pareto optimal solutions to the decision maker. A single solution (i.e., the final solution) is selected by the decision maker's preference.

In this paper, we employed the two-point crossover and the shift mutation shown in Fig. 5 for flowshop scheduling. These genetic operators were found to be good operators for a single-objective genetic algorithm for flowshop scheduling in our previous work [15].

3. SIMULATION RESULTS FOR A NUMERICAL EXAMPLE

If multi-objective optimization problems have concave Pareto fronts, weighted sum approaches with constant weights tend to fail to find entire Pareto fronts (i.e., fail to find all the Pareto optimal solutions). Our approach, however, can handle multi-objective optimization problems with concave Pareto fronts. This is shown by the following example.

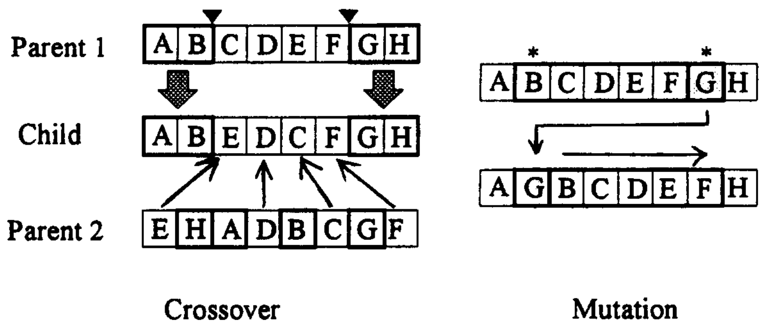


Fig. 5. Genetic operators.

We applied our multi-objective genetic algorithm to a test problem with the following two objectives to be minimized:

$$\text{Minimize} f_1(\mathbf{x}) = 2\sqrt{x_1}, \tag{7}$$

$$\text{Minimize} f_2(\mathbf{x}) = x_1(1 - x_2) + 5, \tag{8}$$

subject to

$$\begin{cases} 1 \leqslant x_1 \leqslant 4, \\ 1 \leqslant x_2 \leqslant 2. \end{cases} \tag{9}$$

Substituting equation (7) into equation (8), we obtained the relation between $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ as follows:

$$f_2(\mathbf{x}) = \frac{1 - x_2}{4} \cdot \{f_1(\mathbf{x})\}^2 + 5. \tag{10}$$

When $x_2 = 2$, equation (10) gives the Pareto front of this problem. This Pareto front forms the concave shape in the objective space as shown in Fig. 6.

In our multi-objective genetic algorithm, the fitness function $f(\mathbf{x})$ was specified as

$$f(\mathbf{x}) = -w_1 \cdot f_1(\mathbf{x}) - w_2 \cdot f_2(\mathbf{x}). \tag{11}$$

Because $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ should be minimized, the negative sign " $-$ " is attached to each weight in (11).

In our multi-objective genetic algorithm, the number of solutions in each population (i.e., population size) was specified as $N_{\text{pop}} = 100$, two point crossover was employed with the crossover probability 0.9, the mutation probability was specified as $P_m = 0.01$, and the number of elite individuals was specified as $N_{\text{elite}} = 5$ (five elite solutions were randomly chosen from a tentative set of Pareto optimal solutions). In Fig. 6, ■ denotes the final solutions by our multi-objective genetic algorithm after 20 generations. From Fig. 6, we can observe that our multi-objective genetic algorithm can find many solutions on the concave Pareto front.

We also applied a single-objective genetic algorithm where the weights w_1 and w_2 were fixed as follows:

$$w_1{:}w_2 = 100{:}1,\ 50{:}1,\ 20{:}1,\ 15{:}1,\ 10{:}1,\ 5{:}1,\ 2{:}1,$$
$$1{:}1,\ 1{:}2,\ 1{:}5,\ 1{:}10,\ 1{:}15,\ 1{:}20,\ 1{:}50,\ 1{:}100.$$

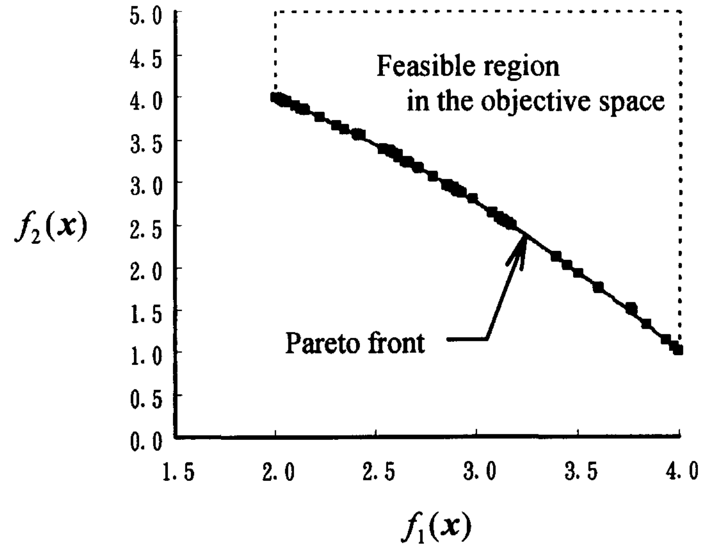


Fig. 6. Simulation result for the concave Pareto front problem.

The single-objective genetic algorithm with these 15 different weight values found only two Pareto solutions: $(f_1, f_2) = (2, 4)$ and $(f_1, f_2) = (4, 1)$. From these simulation results, we can see that the single-objective genetic algorithm with constant weights cannot find the concave Pareto front even if various weight values were employed.

4. SIMULATION RESULTS FOR FLOWSHOP SCHEDULING

In this section, we demonstrate the effectiveness of our multi-objective genetic algorithm by computer simulations on a flowshop scheduling problem with two objectives: to minimize the makespan and to minimize the total tardiness. We also apply our algorithm to the flowshop scheduling problem with three objectives: to minimize the makespan, to minimize the total tardiness, and to minimize the total flowtime.

Flowshop scheduling problems are one of the most well-known problems in the area of scheduling. General assumptions of the flowshop scheduling problems can be written as follows (see Dudek *et al.* [16]). Jobs are to be processed on multiple stages sequentially. There is one machine on 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 our flowshop scheduling is the n-job sequencing problem. In this paper, the sequence of the n jobs is denoted by a string $\mathbf{x} = (x_1, \ldots, x_k, \ldots, x_n)$ where x_k is the k-th job to be processed on the m machines.

4.1. Two-objective flowshop scheduling problem

In computer simulations, we employed two objectives as scheduling criteria; to minimize the makespan (i.e., the completion time of the last job) and to minimize the total tardiness (i.e., the sum of the tardiness for the duedate of each job). We specified the due date of each job by the following procedure:

Step 1: Randomly generate a permutation of the n jobs.
Step 2: Calculate the completion time C_j of each job, $j = 1, 2, \ldots, n$.
Step 3: Add a random integer rnd_j in the closed interval $[-100, 100]$ to each C_j. That is, the duedate d_j of the j-th job is $d_j = C_j + rnd_j$.

It is known that there is no correlation between the two objectives: the makespan and the total tardiness. In our multi-objective genetic algorithm, the fitness function $f(\mathbf{x})$ can be written as

$$f(\mathbf{x}) = -W_{\text{Makespan}} \cdot \mathit{Makespan}(\mathbf{x}) - W_{\text{Tardiness}} \cdot \mathit{Tardiness}(\mathbf{x}), \tag{12}$$

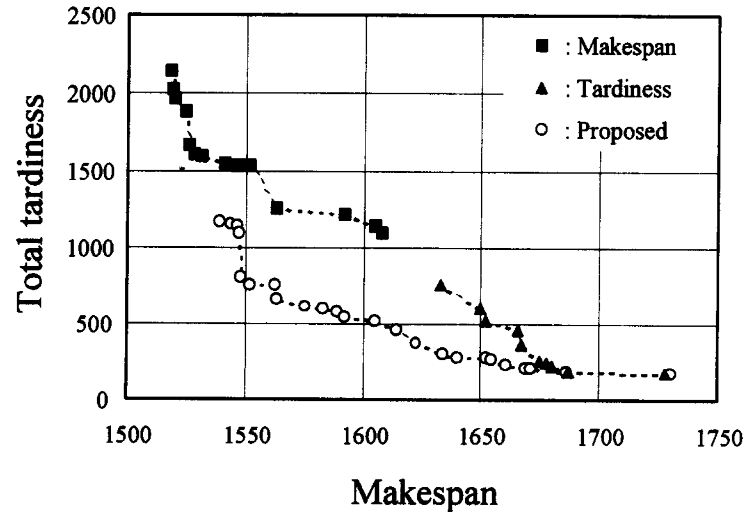


Fig. 7. Comparison of our multi-objective genetic algorithm with two trials of the single-objective genetic algorithm.

where *Makespan*(**x**) is the makespan when the n jobs are processed in the order of **x**, *Tardiness*(**x**) is the total tardiness, and W_{Makespan} and $W_{\text{Tardiness}}$ are non-negative variable weights for *Makespan*(**x**) and *Tardiness*(**x**), respectively. In computer simulations, W_{Makespan} and $W_{\text{Tardiness}}$ were specified in the same manner as (3). Because *Makespan*(**x**) and *Tardiness*(**x**) should be minimized, the negative sign " – " is attached to each weight in (12).

As a test problem, we generated a flowshop scheduling problem with 20 jobs and 10 machines by randomly specifying the processing time of each job at each machine as an integer in the closed interval [1, 99]. In computer simulation, we used the following parameter specifications.

Population size:	$N_{\text{pop}} = 10$,
Crossover probability:	$P_c = 1.0$,
Mutation probability:	$P_m = 1.0$/string.

Non-dominated solutions obtained by the proposed algorithm are shown by ○ in Fig. 7 where the horizontal and vertical axes are the makespan and the total tardiness, respectively. In Fig. 7, non-dominated solutions obtained by the two trials of a single-objective genetic algorithm are shown by ■ (obtained by the genetic algorithm for minimizing the makespan) and ▲ (obtained by the genetic algorithm for minimizing the total tardiness). In the single-objective genetic algorithm, the fitness function was equivalent to the value of its objective function with the negative sign " – ". A tentative set of Pareto optimal solutions was stored and updated in the single-objective genetic algorithm in the same manner as in the multi-objective genetic algorithm. In order to compare the proposed algorithm with the two trials of the single-objective genetic algorithm under the same computation load, we specified the number of evaluations of the fitness function as 100,000 in our multi-objective genetic algorithm and as 50,000 in each trial of the single-objective genetic algorithm. Therefore 100,000 solutions were evaluated in each approach.

From Fig. 7, we can see that the set of the non-dominated solutions obtained by our multi-objective genetic algorithm (○) is superior to the set of the non-dominated solutions obtained by the single-objective genetic algorithm (■ and ▲). This is because many solutions denoted by ■ and ▲ are dominated by solutions denoted by ○. This demonstrates the high performance of our multi-objective genetic algorithm.

The effectiveness of the elite preserve strategy is demonstrated in Fig. 8. In Fig. 8, "no elite", "2 elite" amd "3 elite" mean that no elite individual, two elite individuals and three elite individuals

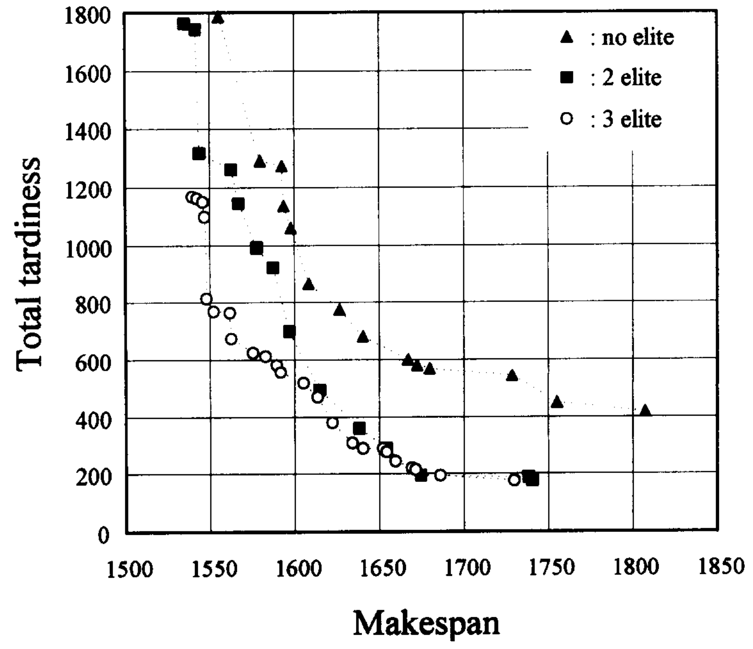
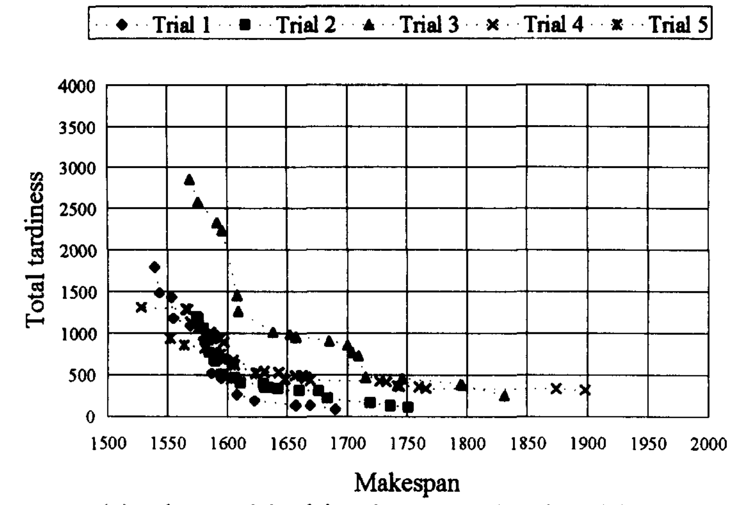
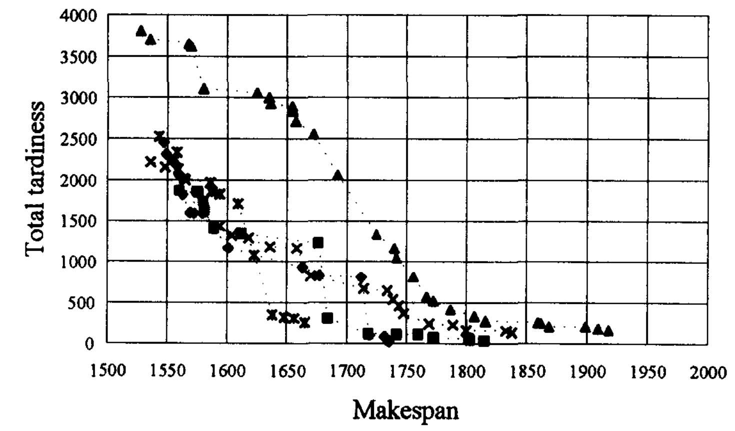


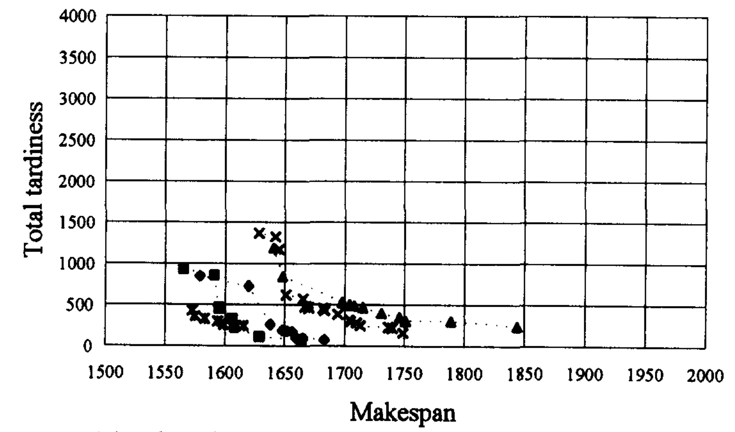
Fig. 8. Effect of the number of elite individuals.



(a) The multi-objective genetic algorithm



(b) Schaffer's VEGA



(c) The single-objective genetic algorithm

Fig. 9. Comparison of our multi-objective genetic algorithm with Schaffer's VEGA and the single-objective genetic algorithm.

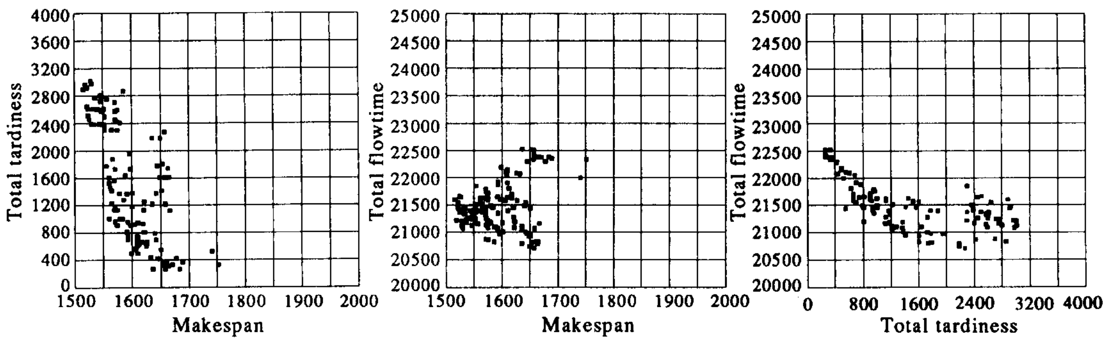
were preserved to the next population in the multi-objective genetic algorithm, respectively. In the "2 elite" algorithm, only the elite individuals with respect to the two objective functions were preserved. On the other hand, an individual randomly selected from a tentative set of Pareto optimal solutions was preserved in addition to such two elite individuals in the "3 elite" algorithm.

Table 1. Average CPU time used by each algorithm for finding the Pareto-optimal solutions in Fig. 9

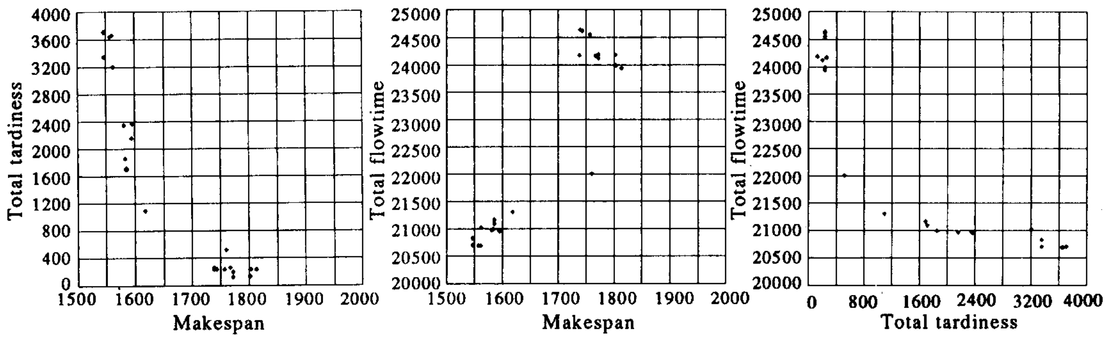
Algorithm	Proposed algorithm	Schaffer's VEGA	SOGA
CPU time (s)	3.08	3.00	2.48

From Fig. 8, we can see that the "3 elite" method found better solutions than the "2 elite" and "no elite" methods. This means that our elite preserve strategy had an effect on the performance of the multi-objective genetic algorithm.

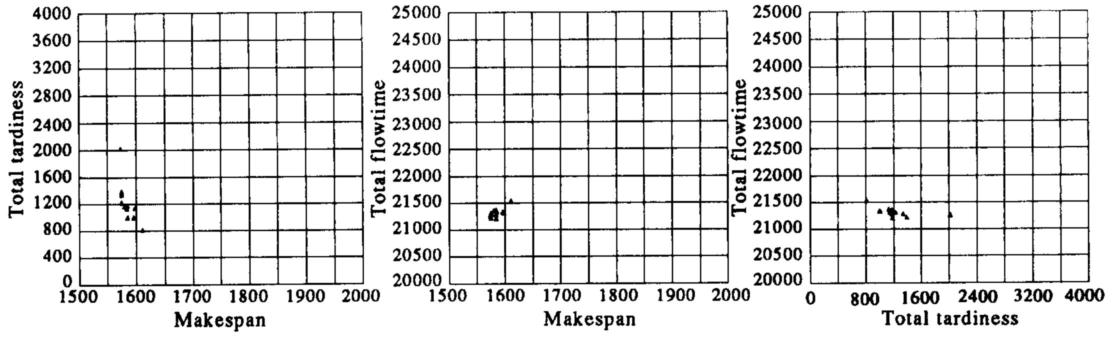
We also applied Schaffer's VEGA [11] and another version of the single-objective genetic algorithm to the same flowshop scheduling problem. In the single-objective genetic algorithm, we used the weights $W_{\text{Makespan}} = 5$ and $W_{\text{Tardiness}} = 2$ to calculate the fitness value. A tentative set of Pareto optimal solutions was stored and updated in the single-objective genetic algorithm in the



(a) The multi-objective genetic algorithm



(b) Schaffer's VEGA



(c) The single–objective genetic algorithm

Fig. 10. Comparison of our multi-objective genetic algorithm with Schaffer's VEGA and the single-objective genetic algorithm.

same manner as in the multi-objective genetic algorithm. As a stopping condition, we used the total number of evaluations of strings (i.e., solutions). When 100,000 solutions were evaluated in each algorithm, the algorithm was terminated. The simulation results by the proposed genetic algorithm, Schaffer's VEGA and the single-objective genetic algorithm are shown in Figs 9(a), (b) and (c), respectively. We applied each algorithm five times to the same flowshop scheduling problem. Each algorithm began to search Pareto-optimal solutions from the same initial generation. From Fig. 9, we can see that better solutions were obtained by the multi-objective genetic algorithm because many solutions obtained by Schaffer's VEGA in Fig. 9(b) are dominated by multi-objective genetic algorithm solutions in Fig. 9(a). The single-objective genetic algorithm could find some better solutions than our multi-objective genetic algorithm, but the single-objective algorithm tended to fail to find large Pareto fronts. The average CPU time used by each algorithm for finding the solutions in Fig. 9 is shown in Table 1. We can see from Table 1 that a similar computation time was spent by each algorithm in computer simulations.

4.2. Three-objective flowshop scheduling problem

We also applied our multi-objective genetic algorithm to a flowshop scheduling problem with three objectives: to minimize the makespan, to minimize the total tardiness and to minimize the total flowtime (i.e., the sum of completion time over all jobs). We used the same parameters as used for the two-objective flowshop scheduling problem. Because it is difficult to show solutions in the three-dimensional objective space, we show the solutions by projecting them on to two-dimensional objective spaces. Figure 10 shows the simulation results obtained by our multi-objective genetic algorithm, Schaffer's VEGA and the single-objective genetic algorithm. In the single-objective genetic algorithm, we used the constant weights $W_{\text{Makespan}} = 5$, $W_{\text{Tardiness}} = 2$ and $W_{\text{Flowtime}} = 1$ to calculate the fitness value. From Fig. 10, we can observe that our multi-objective genetic algorithm could find a better set of solutions.

5. CONCLUSION

In this paper, we proposed a framework of genetic algorithms for multi-objective optimization problems. Our approach has two characteristic features. One is that the weights used for combining multiple objectives into a scalar fitness function are randomly specified for each selection. That is, the weights are not constant but variable in our multi-objective genetic algorithm. The other characteristic feature is that multiple elite individuals selected from a tentative set of Pareto optimal solutions are inherited to the next generation. By computer simulations, we showed that our multi-objective genetic algorithm could find concave Pareto optimal solutions, and we demonstrated that our multi-objective genetic algorithm could find better solutions than the VEGA (Vector Evaluated Genetic Algorithm) by Schaffer [11] and the single-objective genetic algorithm.

Acknowledgement—This work was partially supported by Grant-in-Aid for International Scientific Research (No. 06045047) from the Ministry of Education, Science and Culture in Japan.

REFERENCES

1. S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Res. Logistics Q.* **1(1)**, 61–68 (1954).
2. D. G. Dannenbring. An evaluation of flowshop sequencing heuristics. *Mgmt Sci.* **23**, 1174–1182 (1977).
3. M. Nawaz Jr, E. E. Enscore and I. Ham. A heuristic algorithm for *m*-machine, *n*-job flowshop sequencing problem. *OMEGA* **11**, 91–98 (1983).
4. I. H. Osman and C. N. Potts. Simulated annealing for permutation flow-shop scheduling. *OMEGA* **17(6)**, 551–557 (1989).
5. M. Widmer and A. Hertz. A new heuristic method for the flowshop sequencing problem. *Europ. J. Opnl Res.* **41(2)**, 186–193 (1990).
6. E. Ignall and L. E. Schrage. Application of branch- and bound technique to some flow shop problems. *Ops Res.* **13(3)**, 400–412 (1965).
7. Z. Lomnicki. A branch- and -bound algorithm for the exact solution of the three-machine scheduling program. *Opnl Res. Q.* **16(1)**, 89–107 (1965).
8. J. C. Ho and Y.-L. Chang. A new heuristic for the *n*-job, *m*-machine flowshop problem. *Europ. J. Opnl Res.* **52**, 194–202 (1991).

9. R. Gangadharan and C. Rajendran. A simulated annealing heuristic for scheduling in a flowshop with bicriteria. *Proc. 16th Int. Conf. on Computers Ind. Engng* 345–348, 7–9 Mar. (1994).
10. K. Morizawa, T. Ono, H. Nagasawa and N. Nishiyama. An interactive approach for searching a preferred schedule. *J. Japan Ind. Mgmt Assoc.* **44(4),** 277–283 (1993, in Japanese).
11. J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. *Proc. 1st ICGA*, pp. 93–100 (1985).
12. M. Gen, K. Ida, E. Kono and Y. Li. Solving bicriteria solid transportation problem by genetic algorithm. *Proc. 16th Int. Conf. on Computers Ind. Engng*, 572–575, 7–9 Mar. (1994).
13. H. Tamaki, M. Mori, M. Araki, Y. Mishima and H. Ogai. Multi-criteria optimization by genetic algorithms: a case of scheduling in hot rolling process. *Proc. APORS'94*, 374–381, 26–29 July (1994).
14. J. Horn, N. Nafpliotis and D. E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. *Proc. 1st CIEC*, pp. 82–87 (1994).
15. T. Murata, H. Ishibuchi and H. Tanaka. Genetic algorithms for flowshop scheduling problems. *Computers ind. Engng* (to appear).
16. R. A. Dudek, S. S. Panwalkar and M. L. Smith. The lessons of flowshop scheduling research. *Ops Res.* **40(1),** 7–13 (1992).