MOGA: Multi-Objective Genetic Algorithms

Tadahiko MURATA and Hisao ISHIBUCHI
Department of Industrial Engineering, University of Osaka Prefecture
Gakuen-cho 1-1, Sakai, Osaka 593, JAPAN
Phone: +81-722-52-1161 Fax: +81-722-59-3340
e-mail: isys@center.osakafu-u.ac.jp

ABSTRACT

In this paper, we propose a framework of genetic algorithms to search for Pareto optimal solutions (i.e., non-dominated solutions) of multi-objective optimization problems. Our approach differs from single-objective genetic algorithms in its selection procedure and elite preserve strategy. The selection procedure in our genetic algorithms selects individuals for a crossover operation based on a weighted sum of multiple objective functions. The characteristic feature of the selection procedure is that the weights attached to the multiple objective functions are not constant but randomly specified for each selection. The elite preserve strategy in our genetic algorithms 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.

I. INTRODUCTION

Genetic Algorithms have been mainly applied to single-objective optimization problems. Many real-world problems, however, have multiple objective functions. These objective functions should be combined into a scalar fitness function in order to be handled by a single-objective genetic algorithm. If we assign a constant weight to each of multiple objective functions for combining them, the direction of search in the genetic algorithm is fixed 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.

Since Schaffer's work [10], a few studies have been attempted for applying genetic algorithms to multi-objective optimization problems. Schaffer proposed the Vector Evaluated Genetic Algorithm (VEGA) for finding Pareto optimal solutions of multi-objective optimization problems. In the VEGA, a population is divided into disjoint subpopulations that are governed by different objective functions. Although Schaffer reported some successful results, the VEGA seems to be able to find only extreme solutions on the Pareto front. Fig.2 shows the search directions in the VEGA. As shown in Fig.2, the VEGA may find extreme solutions because the search directions are parallel to the axes of the objective space. Schaffer suggested two approaches to improve the VEGA. 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.

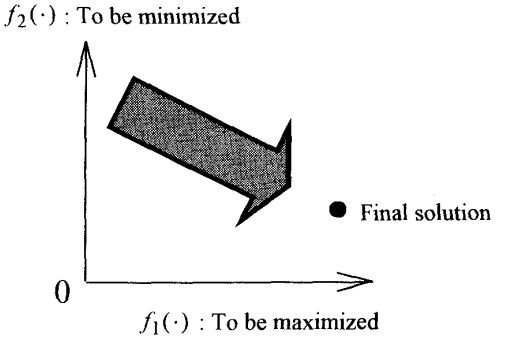

Fig.1 Direction of the search in GA with a combined fitness function

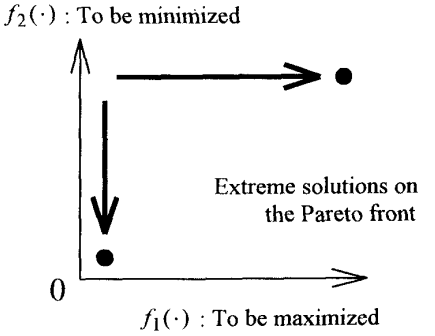

Fig.2 Directions of the search in the VEGA

The aim of multi-objective optimization problems is 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

0-7803-2759-4/95/$4.00

approach is to show the set of Pareto optimal solutions to the decision maker. Then one of the Pareto optimal solutions can be chosen depending on the preference. To find out all the Pareto optimal solutions by genetic algorithms, the variety of individuals should be kept in each generation. Recently Horn *et al.* [5] proposed the Niched Pareto Genetic Algorithm by incorporating the concept of Pareto domination in the selection procedure and applying a niching pressure to spread the population out along the Pareto front.

In this paper, we propose a Multi-Objective Genetic Algorithm (MOGA) with various search directions as shown in Fig.3. In the selection procedure, our MOGA uses a weighted sum of multiple objective functions to combine them into a scalar fitness function. The characteristic feature of our MOGA is that the weights attached to the multiple objective functions are not constant but randomly specified for each selection. Therefore the direction of the search in the MOGA is not fixed. A tentative set of Pareto optimal solutions is preserved in the execution of the MOGA. A certain number of individuals in this set are inherited to the next generation as elite individuals.

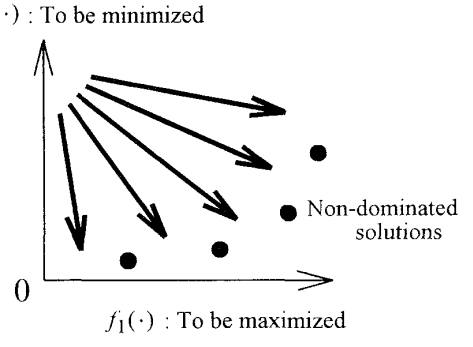

Fig.3 Directions of the search in the Multi-Objective Genetic Algorithm (MOGA)

II. GENETIC OPERATIONS IN MULTI-OBJECTIVE GENETIC ALGORITHM

A. Selection Procedure

One simple way to combine multiple objective functions into a scalar fitness function is the following weighted sum approach (we assume that all the objective functions should be maximized):

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

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

If we use constant weights w_i's in (1), the search direction in genetic algorithms is fixed as shown in Fig.1. Therefore we propose a selection procedure with random weights to search for Pareto optimal solutions by utilizing various search directions as shown in Fig.3.

When a pair of strings are selected for a crossover operation, we assign a random real number to each weight as follows,

$$w_i = \frac{random_i(\cdot)}{\sum_{j=1}^{n} random_j(\cdot)}, \quad i = 1,2,\cdots,n, \tag{2}$$

where $random_j(\cdot)$ is a non-negative random number. From (2), we can see that w_i is a real number in the closed interval [0, 1]. Next pair of strings are selected with different weight values newly given by (2), and so on.

B. Elite Preserve Strategy

During the execution of the MOGA, a tentative set of Pareto optimal solutions is stored and updated at every generation. A certain number (say, N_{elite}) of individuals are randomly selected from the set at each generation. Those solutions are used as elite individuals in our MOGA. This elite preserve strategy has an effect in keeping the variety of each population in our MOGA.

C. Algorithm

The block diagram of the proposed MOGA is shown in Fig.4 and it is described below.

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 of each string using the random weights w_i's in (2). Select a pair of strings from the current population according to the following selection probability. The selection probability $P(\boldsymbol{x})$ of a string $\boldsymbol{x}$ in a population Ψ is specified as

$$P(\boldsymbol{x}) = \frac{f(\boldsymbol{x}) - f_{\min}(\Psi)}{\sum_{\boldsymbol{x} \in \Psi} \{f(\boldsymbol{x}) - f_{\min}(\Psi)\}}, \tag{3}$$

where

$$f_{\min}(\Psi) = \min\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \Psi\}. \tag{4}$$

This step is repeated for selecting $N_{pop}/2$ pairs of strings from the current populations.

Step 3 (Crossover): For each selected pair, apply a crossover operation to generate two new strings. N_{pop} new strings are generated by the crossover operation.

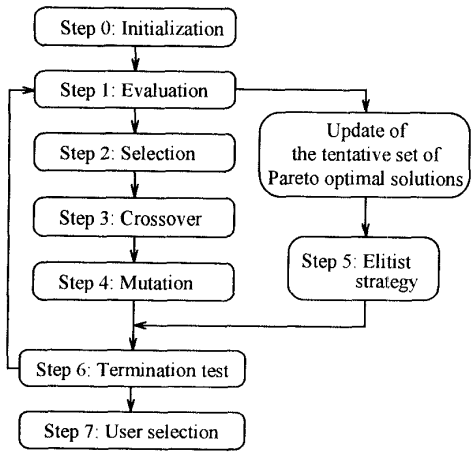

Fig.4 The block diagram of the Multi-Objective Genetic Algorithm (MOGA)

Step 4 (Mutation): For each bit value of the strings generated by the crossover operation, apply a mutation operation with a prespecified mutation probability.

Step 5 (Elitist strategy): Randomly remove N_{elite} strings from the set of the N_{pop} strings generated by the previous operations, and replace them with N_{elite} strings randomly selected from a tentative set of Pareto optimal solutions.

Step 6 (Termination test): If a prespecified stopping condition is not satisfied, return to Step 1.

Step 7 (User selection): The MOGA shows the final set of Pareto optimal solutions to the decision maker. The best solution is then selected according to the decision maker's preference.

The genetic operations such as crossover and mutation are decided depending on the feature of the problem at hand.

III. SIMULATION RESULTS

In this section, we first compare the proposed MOGA with the VEGA by Schaffer [10] and the Niched Pareto GA by Horn *et al.* [5] on the test problem used in [5]. Then we demonstrate the effectiveness of our MOGA by computer simulations on a flowshop scheduling problem and a fuzzy rule selection problem.

A. A Simple Test Problem

First we show the simulation results by the three genetic algorithms (VEGA, Niched Pareto GA and MOGA) for a simple test problem [5] called "unitation versus pairs". This problem has two objectives to be maximized: unitation and complementary adjacent pairs. Unitation $Unit(\boldsymbol{x})$ is simply the number of 1's in the fixed length bit string $\boldsymbol{x}$ (*e.g.*, $Unit(11010100) = 4$). Pairs $Prs(\boldsymbol{x})$ is the number of pairs of complementary adjacent bits (*i.e.*, 01 or 10). For example, $Prs(11010100) = 5$.

When our MOGA was applied to this problem, we used the fitness function as

$$f(\boldsymbol{x}) = w_{Unit} \cdot Unit(\boldsymbol{x}) + w_{Prs} \cdot Prs(\boldsymbol{x}), \qquad (5)$$

where w_{Unit} and w_{Prs} are randomly specified non-negative weights for $Unit(\cdot)$ and $Prs(\cdot)$ respectively.

Pareto optimal solutions of this problem with 12-bit strings are shown in Fig.5 [5] where "P" indicates a Pareto optimal solution and "-" denotes a feasible solution dominated by the Pareto optimal solution(s).

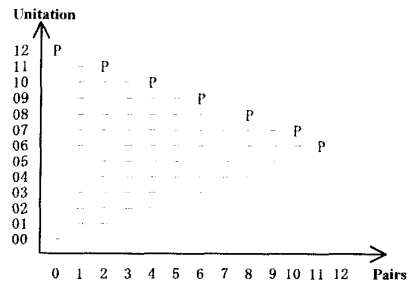

Fig.5 Feasible (-) and Pareto (P) solutions [5]

In all the three algorithms, the one-point crossover was employed with the crossover probability 0.9. The number of individuals in each population (*i.e.*, population size) was specified as $N_{pop} = 100$. Fig.6 ~ Fig.8 show the final solutions by the three algorithms after 100 generations (*i.e.*, solutions in the 100-th generation). In Fig.6 ~ Fig.8, each numeral plotted in the 2-dimensional objective space shows the number of obtained solutions (*i.e.*, individuals) at the corresponding coordinate. Simulation results in Fig.6 and Fig.8 are obtained by our simulation, and Fig.7 is quoted from Horn *et al.* [5]. Pareto optimal solutions in these figures are indicated by encircling the corresponding numerals (*e.g.*, ②).

From these figures, we can observe the characteristic of each approach. Many individuals in the VEGA were driven to the two extreme solutions, *i.e.*, there were 38 and 21 individuals at (Pairs, Unitation) = (0, 12) and (11, 6) respectively (see Fig.6). Two Pareto optimal solutions (Pairs, Unitation) = (4, 10) and (6, 9) were not included in the final generation obtained by the VEGA (see also Fig.6). The Niched Pareto GA succeeded in maintaining equal size subpopulations, but one Pareto optimal solution (Pairs, Unitation) = (11, 6) was not included in the final generation (see Fig.7). Our MOGA could find all the Pareto optimal solutions (see Fig.8).

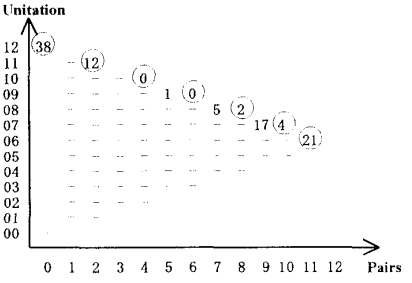

Fig.6 Simulation result by the VEGA

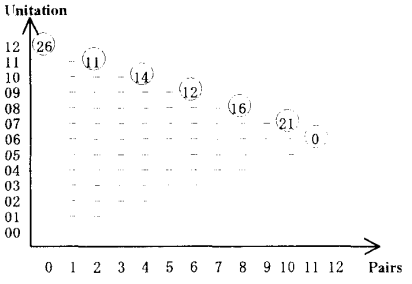

Fig.7 Simulation result by the Niched Pareto GA [5]

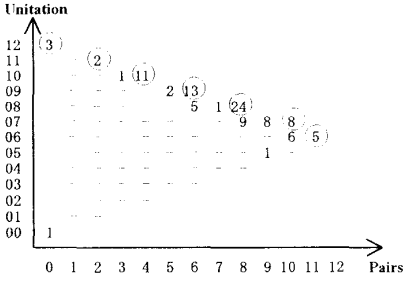

Fig.8 Simulation result by the MOGA

B. A Flowshop Scheduling Problem

Flowshop scheduling problems are quite well-known in the area of scheduling. Recently, several heuristic approaches based on iterative improvement procedures were applied to the flowshop scheduling. Many researchers applied genetic algorithms to scheduling problems (for example, see Fox and McMahon [3], Glass *et al.* [4], Ishibuchi *et al.* [6], Manderick *et al.* [9] and Syswerda [11]).

General assumptions of the flowshop scheduling problems can be described as follows (see Dudek *et al.* [1]). Jobs (work orders) 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 our flowshop scheduling is the n-job and m-machine sequencing problem. Therefore, our purpose is to determine the sequence of the n jobs. This sequence is denoted by a string $\boldsymbol{x}$ in genetic algorithms.

We employed two objectives as scheduling criteria: to minimize the makespan (*i.e.*, the completion time of all jobs) and to minimize the sum of tardiness for the duedate of each job. It is known that there is no correlation between these two objectives. In our MOGA, the fitness function $f(\boldsymbol{x})$ can be written as

$$f(\boldsymbol{x}) = -w_{Makespan} \cdot Makespan(\boldsymbol{x}) - w_{Tardiness} \cdot Tardiness(\boldsymbol{x}), \tag{6}$$

where $Makespan(\boldsymbol{x})$ is the makespan when the n jobs are processed in the order of $\boldsymbol{x}$, $Tardiness(\boldsymbol{x})$ is the sum of tardiness, and $w_{Makespan}$ and $w_{Tardiness}$ are randomly specified non-negative weights for $Makespan(\boldsymbol{x})$ and $Tardiness(\boldsymbol{x})$ respectively. Because $Makespan(\boldsymbol{x})$ and $Tardiness(\boldsymbol{x})$ should be minimized, the negative sign " − " is attached to each weight in (6).

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]. We employed the two-point crossover and the shift change mutation as in Fig.9. In computer simulations, we used the following parameter specifications.

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

In this problem, we assign the mutation probability to each individual. Therefore, $P_m = 1.0$ means that each individual is mutated once a time.

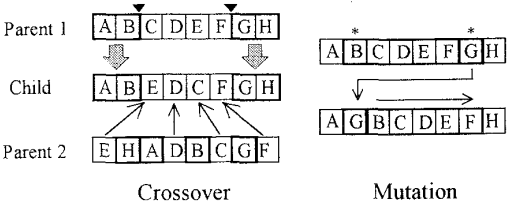

Fig.9 Genetic operations

We applied the VEGA and the MOGA to this flowshop scheduling problem. As a stopping condition, we used the total number of evaluations of sequences (*i.e.*, solutions). When 50,000 solutions were evaluated in each algorithm, the algorithm was terminated. The simulation results by these algorithms are shown in Fig.10 where the horizontal and vertical axes are the makespan and the sum of tardiness respectively. In Fig.10, ■ and ○ show the obtained solutions by the VEGA and the MOGA respectively.

From Fig.10, we can see that better solutions were obtained by the MOGA because many solutions obtained by the VEGA are dominated by the MOGA solutions.

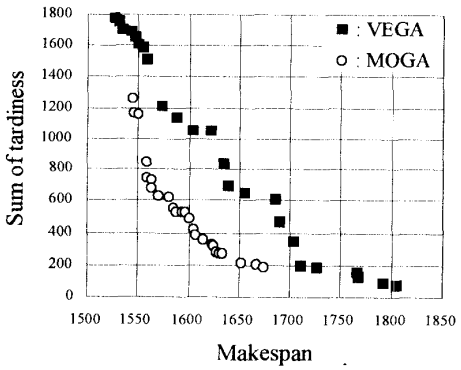

Fig.10 Simulation results to the flowshop problem

C. A Fuzzy Rule Selection Problem

Next, we show the simulation results by the VEGA and the MOGA for a fuzzy rule selection problem in Ishibuchi *et al.* [7, 8]. The fuzzy rule selection problem is to select a small number of fuzzy if-then rules from the possible rules to construct a compact classification system. A set of the selected rules (*i.e.*, an individual) is denoted by a string $\boldsymbol{x}$. This problem has the following two objectives:

(i) To maximize the number of correctly classified patterns by the selected fuzzy if-then rules.
(ii) To minimize the number of the selected fuzzy if-then rules.

By combining these two objectives, we have the following fitness function in the MOGA:

$$f(\boldsymbol{x}) = w_{NCP} \cdot NCP(\boldsymbol{x}) - w_{\boldsymbol{x}} \cdot |\boldsymbol{x}| \tag{7}$$

where $NCP(\boldsymbol{x})$ is the number of correctly classified patterns by the selected fuzzy if-then rules in $\boldsymbol{x}$, $|\boldsymbol{x}|$ is the number of the selected fuzzy if-then rules in $\boldsymbol{x}$, and w_{NCP} and $w_{\boldsymbol{x}}$ are randomly specified non-negative weights for $NCP(\boldsymbol{x})$ and $|\boldsymbol{x}|$, respectively. Because $|\boldsymbol{x}|$ should be minimized, the negative sign " − " is attached to the weight $w_{\boldsymbol{x}}$.

We applied the VEGA and the MOGA to the rule selection problem for the well-known iris data [2]. The classification problem of the iris data is a three-class problem with four attributes. In each class, 50 patterns are given (total 150 patterns). In this work, we employed six antecedent fuzzy sets: "small", "medium small", "medium", "medium large", "large" and "don't care". Therefore we have 6^4 fuzzy if-then rules as candidate rules because the six antecedent fuzzy sets are possible for each of the four attributes. Each individual $\boldsymbol{x}$ has 6^4 bits corresponding to the candidate rules, and each bit assumes 0, 1 or -1. Here, "0" means that the corresponding rule could not be generated from the given data, "1" means that the rule is selected, and "-1" indicates that the rule is not selected.

The uniform crossover was employed with the crossover probability 1.0, and the mutation was biased as $Pm(1 \rightarrow -1) = 0.1$ and $Pm(-1 \rightarrow 1) = 0.001$. These operations were used in both the VEGA and the MOGA in the same manner. Simulation results to the fuzzy rule selection problem are shown in Fig.11 where ■ and ○ denote the obtained solutions by the VEGA and the MOGA respectively.

From Fig.11, we can see that better solutions were obtained by the MOGA because many solutions obtained by the VEGA are dominated by the MOGA solutions.

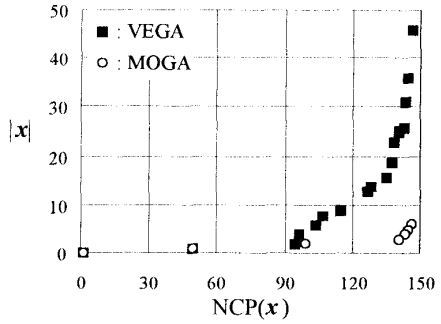

Fig.11 Simulation results to the rule selection problem

IV. DISCUSSION

If multi-objective optimization problems have concave Pareto fronts, weighted sum approaches tend to fail to find entire Pareto fronts (*i.e.*, 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.

We applied the MOGA to a test problem with the following two objectives to be minimized:

$$\text{Minimize } f_1(\boldsymbol{x}) = 2\sqrt{x_1}\,, \tag{8}$$

$$\text{Minimize } f_2(\boldsymbol{x}) = x_1(1 - x_2) + 5\,, \tag{9}$$

subject to

$$\begin{cases} 1 \le x_1 \le 4, \\ 1 \le x_2 \le 2. \end{cases} \tag{10}$$

Substituting Eq.(8) into Eq.(9), we obtained the relation between $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ as follows:

$$f_2(\boldsymbol{x}) = \frac{1 - x_2}{4} \cdot \{f_1(\boldsymbol{x})\}^2 + 5\,. \tag{11}$$

When $x_2 = 2$, Eq.(11) gives the Pareto front of this problem. This Pareto front forms the concave shape in the objective space as shown in Fig.12.

In the proposed MOGA, the fitness function $f(\boldsymbol{x})$ was specified as

$$f(\boldsymbol{x}) = -w_1 \cdot f_1(\boldsymbol{x}) - w_2 \cdot f_2(\boldsymbol{x}) . \quad (12)$$

Because $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ should be minimized, the negative sign " $-$ " is attached to each weight in (12).

In the MOGA, the number of population was specified as $N_{pop} = 100$, two point crossover was employed with the crossover probability 0.9, mutation rate was $P_m = 0.01$, and the number of elite individuals was $N_{elite} = 5$. In Fig.12, ■ denotes the final solutions by the MOGA after 20 generations. From Fig.12, we can observe that the MOGA can find many solutions on the concave Pareto front.

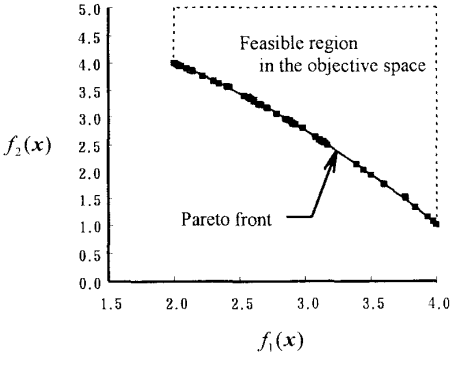

Fig.12 Simulation result to the concave Pareto front problem

We also applied a single objective genetic algorithm (SOGA) where weights w_1 and w_2 were fixed as follows:

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

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.

V. CONCLUSION

In this paper, we proposed a framework of genetic algorithms for multi-objective optimization problems. Our approach has two characteristic features. Firstly, 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 varies at the selection of each pair of strings. Secondly, multiple elite individuals selected from a tentative set of Pareto optimal solutions are inherited to the next generation. By computer simulations, we demonstrated that the proposed MOGA (Multi-Objective Genetic Algorithm) could find better solutions than the VEGA (Vector Evaluated Genetic Algorithm) by Schaffer [10].

REFERENCES

[1] R.A.Dudek, S.S.Panwalkar and M.L.Smith, "The lessons of flowshop scheduling search", *Operations Research*, Vol.**47**(1), pp.65-74, 1992.

[2] R.A.Fisher, "The use of multiple measurements in taxonomic problems", *Annals of Eugenics*, Vol.**7**, pp.179-188, 1936.

[3] B.R.Fox and M.B.McMahon, "Genetic operations for sequencing problems", in G.J.E.Rawlins (ed.) *Foundations of Genetic Algorithms* (Morgan Kaufmann Publishers, San Mateo), pp.284-300, 1991.

[4] C.A.Glass, C.N.Potts and P.Shade, "Genetic algorithms and neighborhood search for scheduling unrelated parallel machines", *Preprint series, No.OR47*, University of Southampton, 1992.

[5] J.Horn, N.Nafpliotis and D.E.Goldberg, "A Niched Pareto Genetic Algorithm for multiobjective optimization", *Proc. of 1st ICEC*, pp. 82-87, 1994.

[6] H.Ishibuchi, N.Yamamoto, T.Murata and H.Tanaka, "Genetic Algorithms and Neighborhood Search Algorithms for Fuzzy Flowshop Scheduling Problems", *Fuzzy Sets and Systems*, Vol.**67**, pp.81-100, 1994.

[7] H.Ishibuchi, K.Nozaki, N.Yamamoto and H.Tanaka, "Construction of fuzzy classification system with rectangular fuzzy rules using genetic algorithms", *Fuzzy Sets and Systems*, Vol.**65**, pp.237-253, 1994.

[8] H.Ishibuchi, T.Murata and I.B.Turksen, "Selecting linguistic classification rules by two-objective genetic algorithms", *Proc. of IEEE-SMC'95* (in press).

[9] B.Manderick and P.Spiessens, "How to select genetic operators for combinatorial optimization problems by analyzing their fitness landscape", in J.M.Zurada *et al.* (eds.) *Computational Intelligence Imitating Life* (IEEE press, NY), pp.170-181, 1994.

[10] J.D.Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms", *Proc. of 1st ICGA*, pp. 93-100, 1985.

[11] G.Syswerda, "Scheduling optimization using genetic algorithms", in L.Davis (ed.) *Handbook of Genetic Algorithms* (Van Nostrand Reinhold, New York), pp.332-349, 1991.