Specification of Genetic Search Directions in Cellular Multi-objective Genetic Algorithms

Tadahiko Murata [1], Hisao Ishibuchi [2] and Mitsuo Gen [1]

[1] Department of Industrial and Information Systems Engineering,
Ashikaga Institute of Technology,
268 Omae-cho, Ashikaga 326-8558, Japan
{murata, gen}@ashitech.ac.jp
[2] Department of Industrial Engineering, Osaka Prefecture University,
1-1 Gakuen-cho, Sakai, Osaka 599-8531, Japan
hisaoi@ie.osakafu-u.ac.jp

Abstract. When we try to implement a multi-objective genetic algorithm (MOGA) with variable weights for finding a set of Pareto optimal solutions, one difficulty lies in determining appropriate search directions for genetic search. In our MOGA, a weight value for each objective in a scalar fitness function was randomly specified. Based on the fitness function with the randomly specified weight values, a pair of parent solutions are selected for generating a new solution by genetic operations. In order to find a variety of Pareto optimal solutions of a multi-objective optimization problem, weight vectors should be distributed uniformly on the Pareto optimal surface. In this paper, we propose a proportional weight specification method for our MOGA and its variants. We apply the proposed weight specification method to our MOGA and a cellular MOGA for examining its effect on their search ability.

1 Introduction

Genetic algorithms have been successfully applied to various optimization problems [1]. The extension of GAs to multi-objective optimization was proposed in several manners (for example, Schaffer [2], Kursawe [3], Horn *et al*. [4], Fonseca & Fleming [5], Murata & Ishibuchi [6], Zitzler & Thiele [7]). The aim of these algorithms is to find a set of Pareto-optimal solutions of a multi-objective optimization problem. Another issue in multi-objective optimization is to select a single final solution from Pareto-optimal solutions. Many studies on multi-objective GAs did not address this issue because the selection totally depends on the decision maker's preference. In this paper, we also concentrate our attention on the search for finding a set of Pareto-optimal solutions.

In this paper, we try to improve the search ability of our multi-objective genetic algorithm (MOGA) in [6] and its variants (i.e., extensions of our MOGA). Fig. 1 shows some extended algorithms in our previous studies [8-10]. By hybridizing our MOGA with a local search procedure, we have already extended it to a multi-objective genetic local search algorithm (MOGLS [8]). We have also extended our MOGA to a cellular multi-objective genetic algorithm (C-MOGA [9]) by introducing a cellular structure. We have employed a local search procedure and a cellular structure in a cellular MOGLS [10]. Furthermore we have extended the cellular algorithms by introducing a relocation procedure (i.e., a kind of immigration) in [10]. Each individual is relocated to a cell at every generation based on the values of multiple objectives (i.e., the location in the multi-dimensional objective space). Those extended algorithms, which are based on the cellular structure and the immigration procedure, are referred to as Cellular Immigrative ("CI-") algorithms in Fig. 1.

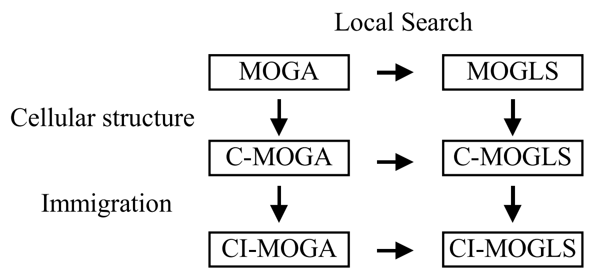

Fig. 1. Extensions of our MOGA by introducing local search, cellular structures and immigration procedures

When we try to implement our MOGA and its variants with variable weights, one difficulty lies in determining appropriate search directions for genetic search. In those algorithms, the weight value for each objective in a scalar fitness function is randomly specified. Based on the scalar fitness function with the randomly specified weight values, a pair of parent solutions are selected for generating a new solution by genetic operations. In order to find a variety of Pareto optimal solutions of a multi-objective optimization problem, a proportional weight specification method is more desirable than the random specification method. In this paper, we propose a proportional weight specification method for our MOGA and its variants. We apply the proposed weight specification method to our MOGA and a cellular MOGA for examining its effect on their search ability.

The concept of cellular genetic algorithms was proposed by Whitley [11]. In cellular genetic algorithms, each individual (i.e. a chromosome) resides in a cell of a spatially structured space. Genetic operations for generating new individuals are locally performed in the neighborhood of each cell. While the term "cellular genetic algorithm" was introduced by Whitley, such algorithms had already been proposed by Manderik & Spiessens [12]. A similar concept was also studied in evolutionary ecology in the framework of "structured demes" (Wilson [13], Dugatkin & Mesterton-Gibbons [14]). The effect of spatial structures on the evolution of cooperative behavior has also been examined in many studies (e.g., Nowak & May [15], Wilson *et al.* [16], Oliphant [17], Grim [18], and Ishibuchi *et al.* [19]) where each individual was located in a cell of single-dimensional or two-dimensional grid-worlds. The concept for generating a grid world on an n-objective space is also employed in the Pareto Archived Evolution Strategy (PAES) [20], where each individual is located in a grid on the objective space. The PAES employs a grid world in order to avoid introducing a niche size in the algorithm.

2 Multi-objective Optimization

Let us consider the following multi-objective optimization problem with *n* objectives:

$$\text{Maximize} \quad f_1(\mathbf{x}),\ f_2(\mathbf{x}),\ \ldots,\ f_n(\mathbf{x}), \tag{1}$$

where $f_1(\cdot)$, $f_2(\cdot)$, ..., $f_n(\cdot)$ are *n* objectives. When the following inequalities hold between two solutions **x** and **y**, the solution **y** is said to dominate the solution **x**:

$$\forall i : f_i(\mathbf{x}) \le f_i(\mathbf{y}) \text{ and } \exists j : f_j(\mathbf{x}) < f_j(\mathbf{y}). \tag{2}$$

If a solution is not dominated by any other solutions of the multi-objective optimization problem, that solution is said to be a Pareto-optimal solution. The task of multi-objective algorithms in this paper is not to select a single final solution but to find all Pareto-optimal solutions of the multi-objective optimization problem in (1). When we use heuristic search algorithms such as taboo search, simulated annealing, and genetic algorithms for finding Pareto-optimal solutions, we usually can not confirm the optimality of obtained solutions. We only know that each of the obtained solutions is not dominated by any other solutions examined during the execution of those algorithms. Therefore obtained solutions by heuristic algorithms are referred to as “nondominated” solutions. For a large-scale multi-objective optimization problem, it is impossible to find all Pareto-optimal solutions. Thus our task is to find many near-optimal nondominated solutions in a practically acceptable computational time. The performance of different multi-objective algorithms is compared based on several quality measures of obtained nondominated solutions.

3 Multi-objective Genetic Algorithms (MOGA)

In this section, we explain our MOGA [6], which is the basic algorithm of the C-MOGA (See Fig. 1). In our MOGA, the weighted sum of the *n* objectives is used as a fitness function:

$$f(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_n f_n(\mathbf{x}), \tag{3}$$

where $w_1, \ldots, w_n$ are nonnegative weights for the *n* objectives, which satisfy the following relations:

$$w_i \ge 0 \text{ for } i = 1, 2, \ldots, n, \tag{4}$$

$$w_1 + w_2 + \cdots + w_n = 1. \tag{5}$$

This fitness function is utilized when a pair of parent solutions are selected for generating a new solution by crossover and mutation. One characteristic feature of our

MOGA is to randomly specify weight values whenever a pair of parent solutions are selected. That is, each selection (i.e., the selection of two parents) is performed based on a different weight vector. This means that each of newly generated solutions by the genetic operations has its own weight vector. The other characteristic feature of our MOGA is preserving all nondominated solutions which are obtained during the execution of the algorithm. We describe these characteristic features in the following subsections.

3.1 Selection Operation

When a pair of parent solutions are to be selected from a current population in a selection operation for generating an offspring by genetic operations, first the n weight values ($w_1, w_2, \ldots, w_n$) are randomly specified as follows:

$$w_i = random_i / (random_1 + \cdots + random_n)\,,\ i = 1, 2, \ldots, n\,, \tag{6}$$

where $random_i$ is a nonnegative random real number. For example, when N pairs of parent solutions are selected for generating a new population, N different weight vectors are specified by (6). This means that N search directions are utilized in a single generation. In other words, each selection (i.e., the selection of two parents) is governed by a different fitness function.

3.2 Elitist Strategy

Our MOGA separately stores two different sets of solutions: a current population and a tentative set of nondominated solutions. After genetic operations are applied to the current population, it is replaced with newly generated solutions. At the same time, the tentative set of nondominated solutions is updated. That is, if a newly generated solution is not dominated by any other solutions in the current population and the tentative set of nondominated solutions, this solution is added to the tentative set. Then all solutions dominated by the added one are removed from the tentative set. In this manner, the tentative set of nondominated solutions is updated at every generation in our MOGA.

From the tentative set of nondominated solutions, a few solutions are randomly selected and added to the current population (see Fig. 2). The randomly selected nondominated solutions may be viewed as elite solutions because they are added to the current population with no modification.

When a multi-objective optimization problem has a non-convex Pareto front, weighted sum approaches with constant weights fail to find its entire Pareto solutions. This is because those algorithms try to find a single optimal solution with respect to the fixed weights by their single trial. Our approach remedies such a difficulty by using variable weights and storing the tentative set of nondominated solutions. This set is updated by examining the Pareto optimality of every solution generated by genetic operations during the execution of the algorithm. It was shown in [6, 8] that our approach found nondominated solutions on a non-convex Pareto front of a two-objective continuous optimization problem. In [6, 8], our approach was also applied to two-objective flowshop scheduling problems with non-convex Pareto fronts.

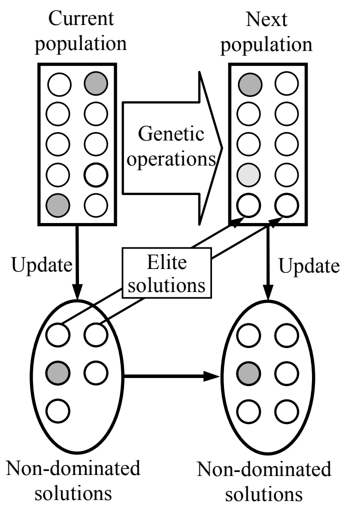

Fig. 2. Illustration of the elitist strategy in the MOGA

3.3 Algorithm

Let us denote the population size by N_{pop} . We also denote the number of nondominated solutions added to the current population by N_{elite} (i.e., N_{elite} is the number of elite solutions, see Fig. 2). Using these notations, our MOGA can be written as follows.

Step 0) **Initialization:** Randomly generate an initial population of N_{pop} solutions.

Step 1) **Evaluation:** Calculate the values of the n objectives for each solution in the current population. Then update the tentative set of nondominated solutions.

Step 2) **Selection:** Repeat the following procedures to select ($N_{\text{pop}} - N_{\text{elite}}$) pairs of parent solutions.

a) Randomly specify the weight values w_1 , $w_2,\ldots,w_n$ in the fitness function (3) by (6).

b) According to the following selection probability $P(\mathbf{x})$, select a pair of parent solutions from the current population Ψ .

$$P(\mathbf{x}) = \frac{f(\mathbf{x}) - f_{\min}(\Psi)}{\sum\limits_{\mathbf{y}\in\Psi}\{f(\mathbf{y}) - f_{\min}(\Psi)\}}\,, \tag{7}$$

where $f_{\min}(\Psi)$ is the minimum fitness value in the current population Ψ .

Step 3) **Crossover and Mutation:** Apply a crossover operator to each of the selected ($N_{\text{pop}} - N_{\text{elite}}$) pairs of parent solutions. A new solution is generated from each pair of parent solutions. Then apply a mutation operator to the generated new solutions.

Step 4) **Elitist Strategy:** Randomly select N_{elite} solutions from the tentative set of nondominated solutions, and add the selected N_{elite} solutions to the ($N_{\text{pop}} - N_{\text{elite}}$) solutions generated in Step 3 to construct a population of N_{pop} solutions.

Step 5) **Termination Test:** If a prespecified stopping condition is satisfied, end the algorithm. Otherwise, return to Step 1.

4 Cellular Algorithms

4.1 Relation between Cell Location and Weight Vector

In cellular algorithms, each individual (i.e. a solution) resides in a cell in a spatially structured space (e.g., two-dimensional grid-world). For utilizing a cellular structure in our MOGA, we assign a different weight vector to each cell. For our *n*-objective optimization problem, cells are structured in an *n*-objective weight space. Fig. 3 shows an example of structured cells for a two-dimensional optimization problem where the two weights w_1 and w_2 are used for the calculation of the fitness function $f(\mathbf{x})$ as $f(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x})$. In this figure, the population size is eleven because an individual exists in each cell. As shown in Fig. 3, the location of each cell corresponds to its weight vector. In order to allocate cells on uniformly distributed weight vectors, we generate weight vectors systematically (not randomly). For example, weight vectors in Fig. 3 are (1.0, 0.0), (0.9, 0.1), ..., (0.0, 1.0).

As shown in Fig. 3, we can easily generate uniform weight vectors on the two-dimensional weight space. In order to generate uniformly distributed weight vectors for multi-objective optimization problems with three or more objectives, we propose a weight specification method on an *n*-dimensional grid world. Let us consider weight vectors satisfying the following conditions.

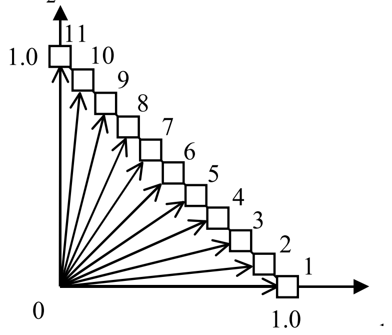

Fig. 3. Location of each cell in the two-dimensional weight space

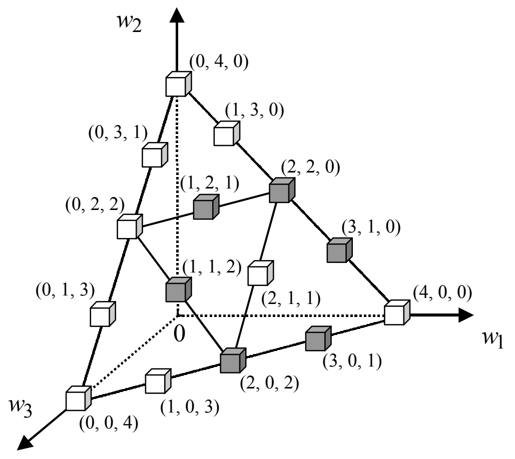

Fig. 4. Location of each cell in the three-dimensional weight space by the proposed method

$$w_1 + w_2 + \cdots + w_n = d \ , \tag{8}$$

$$w_i \in \{0,1,2,...,d\} \, . \tag{9}$$

These conditions show that weight vectors are generated by combining *n* non-negative integers with the sum of *d*. In our cellular algorithm, a cell is located on every weight vector satisfying the above conditions. Thus the number of cells (i.e., the population size) is equal to the total number of weight vectors satisfying the above conditions. This means that the population size is determined by *d*. For example, when we specify *d* as $d = 10$ in (8) for the case of two-objective problems, we will have eleven weight vectors (10, 0), (9, 1), ..., (0, 10). Each of these weight vectors has the same direction as the corresponding weight vector in Fig. 3.

This weight specification method is easily extended to the case with three or more objectives. For example, Fig. 4 shows an example of the three-objective case where *d* is specified as $d = 4$. From Fig. 4, we can observe that the value of *d* can be considered as the number of partitions of the edge between two extreme points (e.g., (0,4,0) and (4,0,0)). By this weight specification method, we can uniformly distribute cells on the *n*-dimensional space. We can calculate the number of cells generated for *n*-objective problems as follows:

$$N_2(d) = d + 1 \approx O(d) \, , \tag{10}$$

$$N_3(d) = \textstyle\sum_{i=0}^{d} N_2(i) = \sum_{i=0}^{d} (i+1) = (i+1)(i+2)/2 \approx O(d^2) \, , \tag{11}$$

$$N_4(d) = \textstyle\sum_{i=0}^{d} N_3(i) = \sum_{i=0}^{d} (i+1)(i+2)/2 \approx O(d^3) \, , \tag{12}$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$N_n(d) = \textstyle\sum_{i=0}^{d} N_{n-1}(i) \approx O(d^{n-1}) \, , \tag{13}$$

where $N_j(d)$, $j = 2,...,n$ are the number of generated cells for *j*-objective problems.

We can see from the above equations that the number of cells can be calculated recursively. We also see that the order of the number of cells is d^{n-1} for *n*-objective problems. Since the number of cells is determined from the value of *d*, the population size of our cellular algorithm can be specified by *d*. In other words, we should specify the value of *d* according to an appropriate population size.

4.2 Definition of Neighborhood

We can arbitrary define a neighborhood structure among cells. That is, we can utilize any distance between cells in the *n*-dimensional space in which cells are structured. For example, the Euclid distance can be used for measuring the distance between cells. In this paper, we use the Manhattan distance. That is, we define the distance between a cell with the weight vector $\mathbf{w} = (w_1, w_2, ..., w_n)$ and another cell with $\mathbf{v} = (v_1, v_2, ..., v_n)$ as follows:

$$Distance(\mathbf{w}, \mathbf{v}) = \sum_{i=1}^{n} | w_i - v_i | . \tag{14}$$

We define the neighborhood of the weight vector **w** as

$$N(\mathbf{w}) = \{\mathbf{v} \mid Distance(\mathbf{w}, \mathbf{v}) \le D\} . \tag{15}$$

For example, when $D = 2$ in Fig. 5, the cell with the weight vector (2,1,1) has six neighboring cells (i.e., shaded cells, (1,2,1), (1,1,2), (2,0,2), (3,0,1), (3,1,0), and (2,2,0) in Fig. 4) and that cell itself in its neighborhood. As shown in this example, the neighborhood of each cell is defined by its nearby cells within the distance *D* including that cell itself.

4.3 Selection

Two parents for generating a new individual in a cell are selected from its neighborhood. When $D = 2$ in Fig. 5, the parent solutions for the cell on (2,1,1) can be selected from that cell and its six neighbors. It is noted that the fitness value of each neighbor is recalculated based on the weight vector assigned to the cell for which a new individual is generated. That is, each individual is differently evaluated by this recalculation procedure of the fitness function in the selection for each cell. This corresponds to the selection procedure of our original MOGA where the selection of each pair of parents was governed by a different weight vector (see Step 2 in Subsection 3.3).

It is noted that the modification of the normalization condition from (5) to (8) has no effect on the selection procedure (Step 2 (a) in Subsection 3.3). Let $\mathbf{w}'$, which satisfies the normalization condition (8), be a weight vector generated by the proposed weight specification method. A weight vector $\mathbf{w} = (w_1, w_2, ..., w_n)$ satisfying the normalization condition (5) can be easily obtained from the relation

$\mathbf{w}' = d \cdot \mathbf{w} = (dw_1, dw_2, ..., dw_n)$. Let us denote our scalar fitness function with the weight vectors $\mathbf{w}$ and $\mathbf{w}'$ by $f(\mathbf{x}, \mathbf{w})$ and $f(\mathbf{x}, \mathbf{w}')$, respectively, for a solution $\mathbf{x}$ in the current population. Since our scalar fitness function in linear with respect to weight values, we have $f(\mathbf{x}, \mathbf{w}') = d \cdot f(\mathbf{x}, \mathbf{w})$ and $f_{\min}(\Psi, \mathbf{w}') = d \cdot f_{\min}(\Psi, \mathbf{w})$ in the selection procedure defined by (7). Thus the same selection probability is obtained from $\mathbf{w}$ and $\mathbf{w}'$ for each solution $\mathbf{x}$ in the current population Ψ. This means that the modification of the normalization condition from (5) to (8) has no effect on the selection procedure.

4.4 Other Genetic Opearations

In the previous Subsections 4.1 to 4.3, we show the characteristic features in our C-MOGA. As for other genetic operations such as crossover, mutation, and elite preserve strategy, we can employ the same operations which can be used in the MOGA. That is, the same crossover and mutation operations can be employed for the C-MOGA. As shown in Fig. 2, some solutions are selected from the tentative set of nondominated solutions, and add them as elite solutions to the current population randomly.

5 Computer Simulations

5.1 Test Problems

We applied the C-MOGA with the proposed weight specification method to flowshop scheduling problems. Flowshop scheduling is one of the most well-known scheduling problems. Since Johnson's work [21], various scheduling criteria have been considered. Among them are makespan, maximum tardiness, total tardiness, and total flowtime. Several researchers extended single-objective flowshop scheduling problems to multi-objective problems (see, for example, Daniels & Chambers [22]).

In this paper, we use the makespan and the total tardiness as two scheduling criteria in our two-objective flowshop scheduling problems. The makespan is the maximum completion time of all jobs to be processed. The total tardiness is the total overdue of all jobs. We also employ the total flowtime together with these two criteria in our three-objective flowshop scheduling problems. The total flowtime is the total completion time of all jobs. Let $g_1(\mathbf{x})$, $g_2(\mathbf{x})$, and $g_3(\mathbf{x})$ be the makespan, the total tardiness, and the total flowtime, respectively. Since these scheduling criteria are to be minimized, we specify the three objectives $f_1(\mathbf{x})$, $f_2(\mathbf{x})$ and $f_3(\mathbf{x})$ of our flowshop scheduling as $f_1(\mathbf{x}) = -g_1(\mathbf{x})$, $f_2(\mathbf{x}) = -g_2(\mathbf{x})$ and $f_3(\mathbf{x}) = -g_3(\mathbf{x})$.

Since flowshop scheduling is to find a job permutation that optimizes the given objectives, a sequence of jobs is handled as an individual (i.e., as a string) in our algorithm. As test problems, we generated ten 20-job and 10-machine flowshop scheduling problems with two and three objectives. The processing time of each job on each machine was specified as a random integer in the interval [1, 99], and the duedate of each job was defined randomly. Our task is to find a set of Pareto-optimal solutions of each test problem.

5.2 Quality Measures of Solution Sets

Since multi-objective algorithms find a set of nondominated solutions with respect to multiple objectives (not a single final solution with respect to a single objective), the comparison between different multi-objective algorithms is not easy. For this purpose, we use the following measures for evaluating the quality of a solution set obtained by each algorithm.

1) **The number of obtained nondominated solutions**
   The number of nondominated solutions obtained by each algorithm is a measure to evaluate the variety of the solution set.
2) **The number of solutions that are not dominated by other solution sets**
   For comparing different solution sets with one another, we examine whether each solution is dominated by any other solutions in other sets. If a solution is dominated by another solution, we remove that solution. In this manner, we remove solutions dominated by other solution sets. The number of remaining solutions in each solution set is a measure for evaluating its relative quality with respect to the other solution sets.
3) **Set quality measure proposed by Esbensen**
   Esbensen [23] proposed an evaluation method of the quality of a solution set. Let us denote a solution set by Ω. The best solution $\mathbf{x}^*$ for a given weight vector $\mathbf{w} = (w_1, w_2, ..., w_n)$ can be chosen from Ω for the n-objective optimization problem as follows:

$$\begin{aligned} f(\mathbf{x}^*) &= w_1 f_1(\mathbf{x}^*) + w_2 f_2(\mathbf{x}^*) + \cdots + w_n f_n(\mathbf{x}^*) \\ &= \max\{w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \cdots + w_n f_n(\mathbf{x}) \mid \mathbf{x} \in \Omega\}\,. \end{aligned} \tag{16}$$

   Esbensen [23] proposed an idea of measuring the quality of the solution set Ω by calculating the expected value of $f(\mathbf{x}^*)$ over possible weight vectors. In this paper, we calculate the expected value of $f(\mathbf{x}^*)$ by randomly generating 10,000 weight vectors by (6). That is, the quality of the solution set Ω is calculated as follows:

$$q(\Omega) = \frac{1}{10000} \sum_{i=1}^{10000} \max\{w_1^i f_1(\mathbf{x}) + w_2^i f_2(\mathbf{x}) + \cdots + w_n^i f_n(\mathbf{x}) \mid \mathbf{x} \in \Omega\}\,, \tag{17}$$

   where $q(\Omega)$ is the quality of the solution set Ω and $\mathbf{w}^i = (w_1^i, w_2^i, ..., w_n^i)$, $i = 1, 2, \ldots, 10000$ are randomly specified weight vectors.

5.3 Simulation Results on Two-Objective Flowshop Scheduling Problems

In our computer simulations, we applied three algorithms (i.e., the MOGA with random weights, the MOGA with weights generated by the proposed method, and the C-MOGA) to test problems in order to show the effectiveness of the weight specification method and compare the search ability of the C-MOGA with that of the MOGAs. We employed the following parameter specifications:

Crossover: Two-point order crossover (crossover rate: 0.8),
Mutation: Shift mutation (mutation rate: 0.3),
Number of elite solutions: $N_{\text{elite}} = 3$,
Neighborhood structure for the local search: Shift,
Stopping condition: Examination of 50,000 solutions.

We used the above stopping condition in order to compare the three algorithms under the same computation load. In a single trial of each algorithm, 50,000 solutions were examined. The parameter *d*, which determines the population size, was specified as $d = 100$ for two-objective problems. This means that the population size was 101 (see the equation (10)). The weight vectors of 101 cells were $(w_1, w_2) = (100, 0)$, (99, 1), ..., (0, 100). For the C-MOGA, we specified the value of *D* as $D = 20$. Therefore parent solutions for each cell are selected from neighboring cells within the Manhattan distance 20.

We examined the effect of the introduction of the weight specification method and the cellular structure (i.e., the locally restricted genetic operations) in this experiment. We obtained a set of nondominated solutions by each algorithm. In Table 1, we summarize the average results over 100 trials for each algorithm (i.e. 10 trials for each of 10 test problems). In this table, "*A*" is the number of nondominated solutions obtained by each algorithm, and "*B*" is the number of solutions that are not dominated by other solutions obtained by the other algorithm. The ratio of these two numbers is shown in the column of *B/A*. "*Quality*" is the set quality measure of Esbensen, and "*SD of Q*" shows the standard deviation of the value of *Quality*. In the calculation of "*SD of Q*", we averaged the standard deviation for each of ten test problems.

From Table 1, we can see that most solutions obtained by the MOGA with random weights are dominated by solutions obtained by the MOGA with the proposed method or the C-MOGA. Thus we can conclude that the weight specification method proposed in this paper is effective in the MOGA and the C-MOGA. We can also observe that the average and the standard deviation of the Quality value for the C-MOGA are better than those for the MOGAs.

Next, we examined the specification of the parameter *D* in the C-MOGA. Table 2 shows the average results over 100 trials for each specification of the neighborhood structure (i.e., each specification of *D*) in the C-MOGA. In this table, the C-MOGA with $D = 200$ is the MOGA with the proposed weight specification method. When $D = 200$, all solutions in the current population are considered as neighbors of every cell in the selection procedure. From Table 2, we can observe that the restriction of the genetic operations within some neighboring cells is effective for improving the ability to find good sets of nondominated solutions.

Table 1. Comparison of MOGA with C-MOGA (Two-Objective)

	A	*B*	*B/A*	*Quality*	*SD of Q*
MOGA (random)	14.6	2.8	0.200	-1063.4	48.4
MOGA (proposed)	17.1	10.8	0.648	-967.4	15.2
C-MOGA ($D = 20$)	17.5	9.3	0.536	-963.6	10.6

A: The number of nondominated solutions of the method.
B: The number of nondominated solutions that are not dominated by those obtained by the other method.
Quality: Set quality measure of Esbensen.
SD of Q: Standard deviation of Quality.

Table 2. Effect of the choice of *D* in C-MOGA (Two-Objective)

D	4	10	20	50	200
Quality	-976.4	-968.6	-963.6	-966.0	-967.4
SD of Q	17.9	14.3	10.6	12.9	15.2

Table 3. Comparison of MOGA with C-MOGA (Three-Objective)

	A	*B*	*B/A*	*Quality*	*SD of Q*
MOGA	44.8	8.4	0.190	-8004.7	65.54
C-MOGA	61.2	63.8	0.966	-7850.9	42.13

Table 4. Effect of the choice of *d* in C-MOGA (Three-Objective)

D	10	11	12	13	14
Poplation Size	66	78	81	105	120
# of Generations	758	642	618	477	417
Quality	-7871.0	-7866.7	-7852.9	-7850.9	-7864.7

5.4 Simulation Results on Three-Objective Flowshop Scheduling Problems

We also applied the C-MOGA and the MOGA with random weights to three-objective test problems. We used the same parameter specifications as in the previous subsection except for the population size. Since we defined $d = 13$ for the C-MOGA, the population size was 105 from the equation (11). In order to compare the two algorithms under the same computation load, we specified the population size in the MOGA as 105. Simulation results are summarized in Table 3 and Table 4. From Table 3, we can see that the performance of the C-MOGA is better than that of the MOGA. Table 4 shows the effect of the choice of a value of *d* on the performance of the C-MOGA. The second row shows the population size calculated from the value of *d*. It is noted that each algorithm with a different value of *d* was terminated when the number of examined solutions exceeded the termination condition. The number of generations is shown in the third row of Table 4 for each specification of *d*. From Table 4, we can see that the best quality value was obtained by the C-MOGA with $d = 13$.

6 Conclusion

In this paper, we proposed a weight specification method for the cellular multi-objective genetic algorithm (C-MOGA), which is an extension of a multi-objective genetic algorithm (MOGA) in our former study (Murata & Ishibuchi [6]). In the proposed C-MOGA, each individual is located in a cell with a different weight vector. This weight vector governs the selection operation. The selection is performed in the neighborhood of each cell. The effectiveness of the C-MOGA with the proposed weight specification method was demonstrated by computer simulations on two- and three-objective flowshop scheduling problems.

References

1. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley (1989).
2. Schaffer, J.D.: Multi-objective optimization with vector evaluated genetic algorithms. *Proc. of 1st Int'l Conf. on Genetic Algorithms* (1985) 93-100.
3. Kursawe, F.: A variant of evolution strategies for vector optimization. In H.-P.Schwefel and R.Männer (Eds.), *Parallel Problem Solving from Nature*, Springer-Verlag, Berlin (1991) 193-197.
4. Horn, J., Nafpliotis, N. and Goldberg, D.E.: A niched Pareto genetic algorithm for multi-objective optimization. *Proc. of 1st IEEE Int'l Conf. on Evolutionary Computation* (1994) 82-87.
5. Fonseca, C. M. and Fleming, P. J.: An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary Computation* **3** (1995) 1-16.
6. Murata, T. and Ishibuchi, H.: Multi-objective genetic algorithm and its applications to flowshop scheduling. *International Journal of Computers and Engineering* **30**, 4 (1996) 957-968.
7. Zitzler, E. and Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto Approach. *IEEE Trans. on Evolutionary Computation* **3** (1999) 257-271.
8. Ishibuchi, H. and Murata, T.: A multi-objective genetic local search algorithms and its application to flowshop scheduling. *IEEE Trans. on System, Man, and Cybernetics, Part C* **28** (1998) 392-403.
9. Murata, T. and Gen, M.: Cellular genetic algorithm for multi-objective optimization. *Proc. of 4th Asian Fuzzy System Symposium* (2000) 538-542.
10. Murata, T., Ishibuchi, H., and Gen, M.: Cellular genetic local search for multi-objective optimization. *Proc. of the Genetic and Evolutionary Computation Conference 2000* (2000) 307-314.
11. Whitley, D.: Cellular Genetic Algorithms. *Proc. of 5th Int'l Conf. on Genetic Algorithms* (1993) 658.
12. Manderick, B. and Spiessens, P.: Fine-grained parallel genetic algorithms. *Proc. of 3rd Int'l Conf. on Genetic Algorithms* (1989) 428-433.
13. Wilson, D. S.: Structured demes and the evolution of group-advantageous traits. *The American Naturalist* **111** (1977) 157-185.
14. Dugatkin, L. A. and Mesterton-Gibbons, M.: Cooperation among unrelated individuals: Reciprocal altruism, by-product mutualism and group selection in fishes. *BioSystems* **37** (1996) 19-30.
15. Nowak, M. A. and May, M.: Evolutionary games and spatial chaos. *Nature* **359** (1992) 826-859.
16. Wilson, D. S., Pollock, G. B., and Dugatkin, L. A.: Can altruism evolve in purely viscous populations? *Evolutionary Ecology* **6** (1992) 331-341.
17. Oliphant, M.: Evolving cooperation in the non-iterated Prisoner's Dilemma: The importance of spatial organization. in R. A. Brooks and P. Maes (Eds.), *Artificial Life IV*, MIT Press, Cambridge (1994) 349-352.

18. Grim, P.: Spatialization and greater generosity in the stochastic Prisoner's Dilemma. *BioSystems* **37** (1996) 3-17.
19. Ishibuchi, H., Nakari, T., and Nakashima T.: Evolution of Strategies in Spatial IPD Games with Structured Demes, *Proc. of the Genetic and Evolutionary Computation Conference 2000* (2000).
20. Knowles, J.D., and Corne, D.W.: Approximating the nondominated front using the Pareto Archived Evolution Strategy, *Evolutionary Computation* (MIT Press), **8**, 2 (2000) 149-172.
21. Johnson, S.M.: Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly* **1** (1954) 61-68.
22. Daniels, R.L. and Chambers, R.J.: Multiobjective flow-shop scheduling. *Naval Research Logistics* **37** (1990) 981-995.
23. Esbensen, H.: Defining solution set quality. *Memorandum* (No.UCB/ERL M96/1, Electric Research Laboratory, College of Engineering, Univ. of California, Berkeley, USA, Jan., 1996).