Cellular Genetic Local Search for Multi-Objective Optimization

Tadahiko Murata
Department of Industrial and
Information Systems Engineering,
Ashikaga Institute of Technology
Ashikaga, Tochigi 326-8558, Japan
murata@ashitech.ac.jp
+81-284-62-0605

Hisao Ishibuchi
Department of Industrial Engineering
College of Engineering
Osaka Prefecture University
Sakai, Osaka 599-8531, Japan
hisaoi@ie.osakafu-u.ac.jp
+81-722-54-9350

Mitsuo Gen
Department of Industrial and
Information Systems Engineering,
Ashikaga Institute of Technology
Ashikaga, Tochigi 326-8558, Japan
gen@ashitech.ac.jp
+81-284-62-0605

Abstract

In this paper, we show how cellular structures can be combined with multi-objective genetic local search (MOGLS) algorithms for improving their search ability to find Pareto-optimal solutions of multi-objective optimization problems. We propose two ideas for implementing a cellular MOGLS algorithm: assignment of a different local search direction to each cell, and relocation of individuals based on their objective values. In our cellular MOGLS algorithm, every individual in each population exists in a cell of a spatially structured space (e.g., two-dimensional grid-world) where each cell has a different local search direction. Such a local search direction corresponds to weights in a scalar fitness function defined by the weighted sum of multiple objectives. The selection of parents for generating a new individual in a cell is performed within the neighborhood of that cell based on its local search direction. A local search procedure is applied to new individuals generated by genetic operations for maximizing the fitness function. It should be noted that each cell has its own local search direction, which is used in the selection as well as in the local search. Newly generated individuals are relocated into cells according to their locations in the multi-dimensional objective space.

1 INTRODUCTION

Genetic algorithms have been successfully applied to various optimization problems (Goldberg 1989). The extension of GAs to multi-objective optimization was proposed in several manners (Schaffer 1985, Kursawe 1991, Horn *et al.* 1994, Fonseca & Fleming 1995, Murata & Ishibuchi 1995, Zitzler & Thiele 1999). 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.

Many hybrid algorithms of GAs and neighborhood search (e.g., local search, simulated annealing, and tabu search) were proposed for single-objective optimization problems to improve the search ability of GAs, and their high performance was reported in the literature. While we can expect significant improvement of the performance of multi-objective GAs by such hybridization, multi-objective hybrid GAs had not been proposed until Ishibuchi & Murata (1998) hybridized their multi-objective genetic algorithm (MOGA: Murata & Ishibuchi 1995) with a local search procedure. Main issues in such hybridization for multi-objective optimization problems are the specification of local search directions and the balance between the genetic global search and the local search. In the multi-objective genetic local search (MOGLS) algorithm of Ishibuchi & Murata (1998), the weighted sum of multiple objectives is used as a fitness function. The fitness function is used in the local search as well as in the selection. For the search of various Pareto-optimal solutions in large areas of the objective space, a different weight vector is randomly specified whenever a pair of parent solutions is selected for generating a new solution. The randomly specified weight vector is also used as the local search direction for the new solution. A local search procedure is applied to the new solution for iteratively improving its fitness value (i.e., the weighted sum of multiple objectives). This means that each new solution has its own local search direction that is randomly specified for the selection of their parents. A more sophisticated specification method was proposed in Murata *et al.*(1999) where the local search direction of each new solution was defined according to its location in the multi-dimensional objective space. In these MOGLS algorithms, the balance between the genetic global search and the local search can be controlled by restricting the number of solutions examined in a single iteration of the local search procedure. That is, all the neighboring solutions are not examined in the local search procedure for preventing it from spending almost all the available computation time. In this paper, we combine these MOGLS algorithms with cellular structures.

The concept of cellular genetic algorithms was proposed by Whitley (1993). 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 proposed by Whitley, such algorithms had already been proposed by Manderik and Spiessens (1989). A similar concept was also studied in evolutionary ecology in the framework of "structured demes" (Wilson 1977, Dugatkin and M. Mesterton-Gibbons 1996). The effect of spatial structures on the evolution of cooperative behavior has also been examined in many studies (e.g., Nowak & May 1992, Wilson *et al.* 1992, Oliphant 1994, Grim 1996, and Ishibuchi *et al.* 2000) where each individual was located in a cell of single-dimensional or two-dimensional grid-worlds.

In this paper, we extend our MOGA (Murata & Ishibuchi, 1995) and MOGLS (Ishibuchi & Murata, 1998) as shown in Figure 1 by combining these algorithms with cellular structures and immigration procedures. First we extend these algorithms by assigning every individual in each population to a cell in a spatially structured space (e.g., two-dimensional grid-world). We add the abbreviation "C-" to show "Cellular-" for the extended algorithms. Furthermore we extend the cellular algorithms by introducing a relocation procedure (i.e., a kind of immigration). 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). The extended algorithms, which are based on the cellular structure and the immigration procedure, are referred to as Cellular Immigrative ("CI-") algorithms in this paper.

2 MULTI-OBJECTIVE OPTIMIZATION

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

$$\text{Maximize} \quad f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x}),\ ...,\ f_n(\boldsymbol{x}), \qquad (1)$$

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

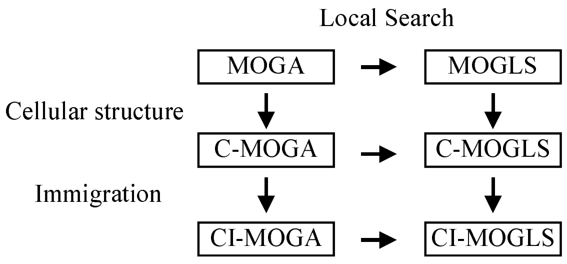

Figure 1: Extensions of the MOGA and the MOGLS in this paper by introducing cellular structures and immigration procedures.

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

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 MOGA AND MOGLS

In this section, we explain our MOGA and MOGLS, which will be extended in later sections of this paper. Since the MOGA can be viewed as a special case of the MOGLS (i.e., MOGLS with no local search procedure), we only show the MOGLS algorithm in detail.

In the MOGLS algorithm of Ishibuchi & Murata (1998), a local search procedure is applied to each of new solutions (i.e., new individuals) generated by genetic operations for iteratively improving their fitness values. The weighted sum of the n objectives is used as a fitness function:

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

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

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

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

This fitness function is utilized when a pair of parent solutions are selected for generating a new solution by crossover and mutation. A local search procedure is applied to the newly generated solution to maximize its fitness value. One characteristic feature of our MOGLS algorithm 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. Another characteristic feature of the MOGLS algorithm is not to examine all neighborhood solutions of a current solution in the local search. Only a small number of neighborhood solutions are examined to prevent the local search procedure from spending almost all the available computation time in the MOGLS algorithm. The neighborhood structure for the local search is defined in the solution space. The neighboring solutions of a current solution are defined by its small modifications (e.g., exchange of two jobs in a schedule corresponding to the current solution).

3.1 SELECTION OPERATION

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

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

where $random_i$ are nonnegative random real numbers. 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 LOCAL SEARCH PROCEDURE

As we have already mentioned, a local search procedure is applied to each new solution generated by the genetic operations (i.e., selection, crossover, and mutation) for maximizing its fitness function $f(\boldsymbol{x})$ in (3). Our MOGLS algorithm employs the following local search procedure with the limited examination of neighborhood solutions for iteratively improving the fitness function $f(\boldsymbol{x})$ of a solution $\boldsymbol{x}$ in a current population:

Step 1) Let $\boldsymbol{x}$ be the initial solution of the local search.

Step 2) Randomly select a neighborhood solution $\boldsymbol{y}$ of the current solution $\boldsymbol{x}$.

Step 3) If $f(\boldsymbol{x}) < f(\boldsymbol{y})$, i.e., if $\boldsymbol{y}$ is a better solution than $\boldsymbol{x}$, replace the current solution $\boldsymbol{x}$ with $\boldsymbol{y}$ and return to Step 2.

Step 4) If randomly selected k neighborhood solutions of the current solution $\boldsymbol{x}$ have been already examined (i.e., if there is no better solution among the examined k neighborhood solutions), stop the local search procedure for the current solution $\boldsymbol{x}$. Otherwise, return to Step 2.

This procedure is terminated when no better solution is found among k neighborhood solutions that are randomly selected from the neighborhood of the current solution. We can adjust the balance between the genetic global search and the local search through the value of k. If we specify k as $k = 0$, our MOGLS algorithm is identical to the MOGA because no local search is executed in our MOGLS algorithm.

3.3 ELITIST STRATEGY

Our MOGLS algorithm separately stores two different sets of solutions: a current population and a tentative set of nondominated solutions. After the local search procedure is applied to each solution, the current population is replaced with the improved solutions by the local search procedure. The tentative set of nondominated solutions is also updated by the improved solutions. That is, if a solution obtained by the local search procedure is not dominated by any other solutions in the improved 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 MOGLS algorithm.

From the tentative set of nondominated solutions, a few solutions are randomly selected and added to the current population before the local search procedure is applied (see Figure 2). That is, the local search procedure is applied to the selected nondominated solutions as well as the generated solutions by the genetic operations. The direction of the local search (i.e., weight values) for each nondominated solution is determined by the fitness function used in the selection of its parent solutions. The randomly selected nondominated solutions may be viewed as elite solutions because they are added to the current population with no genetic operations.

3.4 MOGLS 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 Figure 2). Using these notations, our MOGLS algorithm can be written as follows.

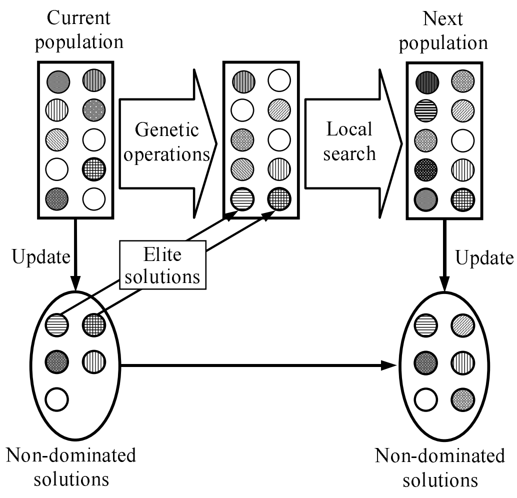

Figure 2: Illustration of our MOGLS algorithm.

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 $P(\boldsymbol{x})$, select a pair of parent solutions from the current population Ψ.

$$P(\boldsymbol{x}) = \frac{f(\boldsymbol{x}) - f_{\min}(\Psi)}{\sum_{\boldsymbol{x} \in \Psi} \{f(\boldsymbol{x}) - f_{\min}(\Psi)\}}, \qquad (7)$$

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) Local Search: Apply the local search procedure in Section 3.2 to each of the N_{pop} solutions in the current population. The local search direction for each solution is specified by the weight values used in the selection of its parent solutions. The current population is replaced with the N_{pop} solutions improved by the local search procedure.

Step 6) Termination Test: If a prespecified stopping condition is satisfied, end the algorithm. Otherwise, return to Step 1.

4 CELLULAR ALGORITHMS

4.1 CELLS WITH WEIGHT VECTORS

In cellular algorithms, each individual resides in a cell in a spatially structured space (e.g., two-dimensional grid-world). For utilizing a cellular structure in our MOGLS algorithm, we assign a different weight vector to each cell. That is, each cell has its own weight vector, which is used for generating a new individual for that cell in the selection and the local search. For our n-objective optimization problem, cells are structured in an n-dimensional weight space. Figure 3 shows an example of structured cells for a two-dimensional optimization problem where the fitness function $f(\boldsymbol{x})$ is defined by two weights w_1 and w_2 as $f(\boldsymbol{x}) = w_1 f_1(\boldsymbol{x}) + w_2 f_2(\boldsymbol{x})$. In this figure, the population size is ten because an individual exists in each cell. As shown in Figure 3, the location of each cell corresponds to its weight vector. Weight vectors are uniformly specified according to the population size. For example, weight vectors for 11 individuals (i.e., for 11 cells) are (1.0, 0.0), (0.9, 0.1), ..., (0.0, 1.0).

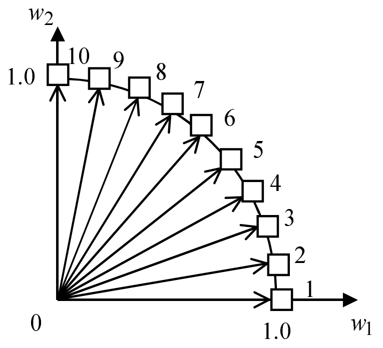

Figure 3: Location of each cell in the weight space.

4.2 DEFINITION OF NEIGHBORHOOD

We can intuitively define the 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, the neighborhood of each cell is defined by its nearest

k_{neighbor} cells (including that cell with zero distance).

4.3 SELECTION AND LOCAL SEARCH

In our MOGLS algorithm described in Section 3, each solution has its own weight vector, which was used for selecting its parent solutions. In our cellular multi-objective genetic local search (C-MOGLS) algorithms proposed in this paper, however, each cell has its own weight vector. As shown in Figure 3, the weight vector assigned to each cell corresponds to its location. For generating a new individual for a cell by the genetic operations, we use its weight vector in the fitness function. The same weight vector (i.e., the same fitness function) is also used for improving the newly generated individual by the local search procedure. The local search is applied to the individual generated by the genetic operation in the same manner as in our MOGLS algorithm in Section 3. The improved individual is the resident in the same cell at the next generation.

Two parents for generating a new individual in a cell are selected from its k_{neighbor} neighbors (including that cell). 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.

5 IMMIGRATION PROCEDURE

Furthermore we extend the C-MOGLS algorithm to a cellular immigrative multi-objective genetic local search (CI-MOGL) algorithm by relocating newly generated individuals based on their locations in the multi-dimensional objective space. That is, we immigrate each individual to an appropriate cell according to its multiple objective values. This procedure is applied to individuals in the current population before the local search. This is to assign an appropriate local search direction to each individual generated by the genetic operations.

Let us illustrate the necessity of this immigration procedure using Figure 4. Let us assume that two individuals indicated large open circles in this figure are selected for generating a new individual for a cell with the depicted weight vector: $\boldsymbol{w} = (0.1, 0.9)$. This weight vector is also used as the local search direction for a new individual generated by the genetic operations from these parents. When a new individual is generated around the parents (e.g., A in Figure 4), the weight vector $\boldsymbol{w} = (0.1, 0.9)$ is appropriate as the local search direction for the new individual. On the contrary, when a new individual is far from the parents (e.g., B in Figure 4), the weight vector $\boldsymbol{w} = (0.1, 0.9)$ is not appropriate as the local search direction. As we can see from Figure 4, an appropriate local search direction may be related to the location of the initial solution for the local search procedure in the multi-dimensional objective space. For example, a weight vector $\boldsymbol{w} = (0.9, 0.1)$ seems to be much more appropriate for the solution B than $\boldsymbol{w} = (0.1, 0.9)$ in Figure 4.

Based on the above discussion, we can relocate each individual generated by the genetic operations to an appropriate cell. This relocation procedure is based on the location of each individual in the objective space. For relocating all individuals in the current population before the local search procedure, we number each individual according to its location in the objective space. For the simplicity of explanation of the immigration procedure, we assume that our multi-objective optimization problem has two objectives $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$. The following procedure is used for numbering the solutions.

[Numbering Procedure]
Let $P_{\boldsymbol{x}}$ be the location of the solution $\boldsymbol{x}$ in the two-dimensional objective space, i.e., $P_{\boldsymbol{x}} = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$. Calculate the angle $\theta(\boldsymbol{x})$ between the $f_1(\boldsymbol{x})$ axis of the objective space and the line $OP_{\boldsymbol{x}}$ where O is the origin of the objective space. The angle $\theta(\boldsymbol{x})$ is obtained from the following relation:

$$\tan\theta(\boldsymbol{x}) = \frac{f_2(\boldsymbol{x})}{f_1(\boldsymbol{x})}. \qquad (8)$$

As shown in the left figure in Figure 5, we number the individuals in the current population from 1 to N_{pop} in an increasing order of $\theta(\boldsymbol{x})$ where N_{pop} is the population size. Each individual is immigrated to the cell with the same number (compare Figure 5 with Figure 3). It should be noted that each cell has its own number assigned in a similar mechanism as shown in Figure 3.

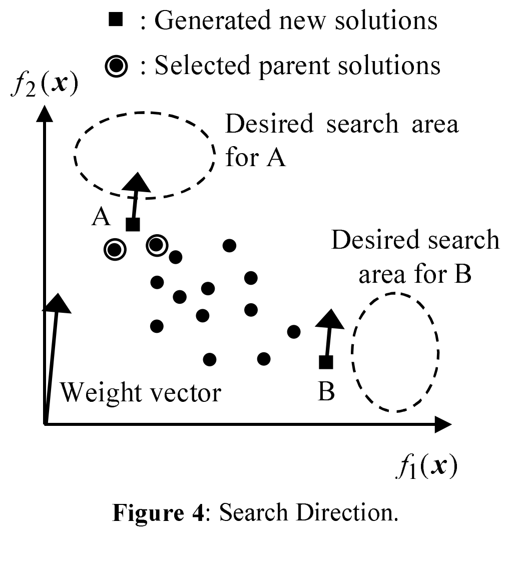
■ : Generated new solutions
◉ : Selected parent solutions
$f_2(\boldsymbol{x})$
Desired search area for A
A
Desired search area for B
Weight vector
B
$f_1(\boldsymbol{x})$

Figure 4: Search Direction.

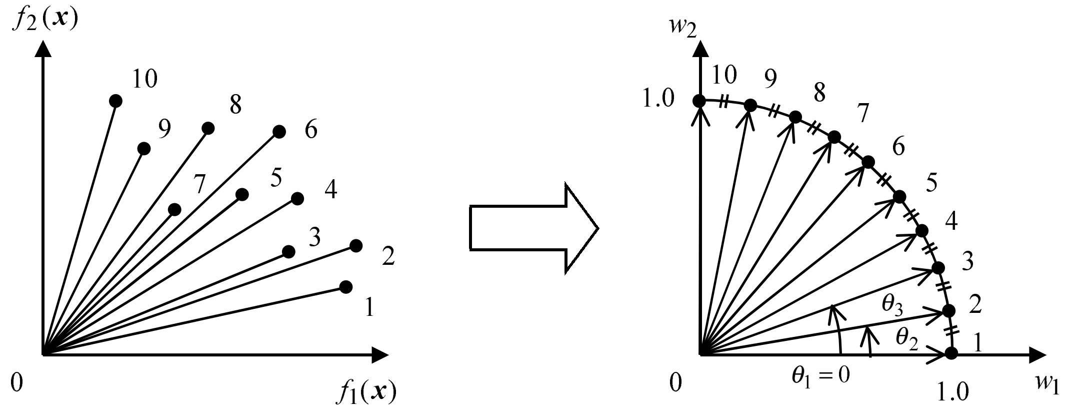

Figure 5: Specification of an appropriate local search direction for each individual.

6 COMPUTER SIMULATIONS

6.1 TEST PROBLEMS

We applied the proposed CI-MOGLS algorithm and its variants to flowshop scheduling problems. Flowshop scheduling is one of the most well-known scheduling problems. Since Johnson's work (1954), 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 1990).

In this paper, we use the makespan and the total tardiness as two scheduling criteria in our 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. Let $g_1(\boldsymbol{x})$ and $g_2(\boldsymbol{x})$ be the makespan and the total tardiness. Since these scheduling criteria are to be minimized, we specify the two objectives $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ of our flowshop scheduling as $f_1(\boldsymbol{x}) = -g_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x}) = -g_2(\boldsymbol{x})$.

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. 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. In our computer simulations, each solution $\boldsymbol{x}$ was represented by a permutation of 20 jobs.

6.2 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 (1996)**
Esbensen (1996) proposed an evaluation method of the quality of a solution set. Let us denote a solution set by Ω. The best solution $\boldsymbol{x}^*$ for a given weight vector $\boldsymbol{w} = (w_1, w_2)$ can be chosen from Ω for the two-objective optimization problem as follows:

$$f(\boldsymbol{x}^*) = w_1 f_1(\boldsymbol{x}^*) + w_2 f_2(\boldsymbol{x}^*) \\ = \max\{w_1 f_1(\boldsymbol{x}) + w_2 f_2(\boldsymbol{x}) \mid \boldsymbol{x} \in \Omega\}. \quad (10)$$

Esbensen (1996) proposed an idea of measuring the quality of the solution set Ω by calculating the expected value of $f(\boldsymbol{x}^*)$ over possible weight vectors. In this paper, we calculate the expected value of $f(\boldsymbol{x}^*)$ by randomly generating 10,000 weight vectors by (6). That is, the quality of the solution set Ω is calculated as follows:

$$q(\Omega) = \frac{1}{10000}\sum_{i=1}^{10000} \max\{w_1^i f_1(\boldsymbol{x}) + w_2^i f_2(\boldsymbol{x}) \mid \boldsymbol{x} \in \Omega\}, \tag{11}$$

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

4) Maximum distance between two solutions

The maximum distance between two solutions in the solution set shows its variety (i.e., the spread of solutions in the multi-dimensional objective space). This is defined for a solution set Ω as

$$D(\Omega) = \max\left\{ \sqrt{\sum_{i=1}^{n} (f_i(\boldsymbol{x}) - f_i(\boldsymbol{y}))^2} \;\middle|\; \boldsymbol{x}, \boldsymbol{y} \in \Omega \right\}. \tag{12}$$

6.3 SIMULATION RESULTS

In our computer simulations, we employed the following parameter specifications in each algorithm:

Population size: $N_{\text{pop}} = 100$ (i.e., 100 cells),
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,
The stopping condition for the local search in Section 3.2:
$k = 10$,
The number of neighboring cells:
$k_{\text{neigh}} = 6, 10, 14, 20, 40$
Stopping condition: Examination of 50,000 solutions.

We used the above stopping condition in order to compare different algorithms under the same computation load. In a single trial of each algorithm, 50,000 solutions were examined. Since the population size was 100, we used 100 cells. The weight vectors of these cells were specified as $\boldsymbol{w} = (w_1, w_2) = (1.00, 0.00), (0.99, 0.01), \ldots, (0.00, 1.00)$.

We examined effects of several elements of the proposed algorithms on the quality of solutions in the following:

(1) Effect of the cellular structure

We examined the effect of the introduction of the cellular structure (i.e., the locally restricted genetic operations). We compared the obtained set of nondominated solutions by the MOGA with that by the C-MOGA with $k_{\text{neighbor}} = 10$.

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, "SD of Q" shows the standard deviation of the value of Quality, and "D" is the maximum Euclid distance between two solutions in the obtained solution set by each algorithm. As for the calculation of "SD of Q", we average the standard deviation for each of ten test problems.

From Table 1, we can see that most solutions obtained by the MOGA are dominated by solutions obtained by the C-MOGA. Thus we can conclude that the C-MOGA outperformed the MOGA, and the standard deviation of Quality value for the C-MOGA is much less than that for the MOGA. This shows that the introduction of the cellular structure in the MOGA improves the performance of the MOGA. Table 2 shows the average results over 100 trials for each specification of k_{neighbor} in the C-MOGA. From this table, we can see that the performance of the C-MOGA is not sensitive to the specification of k_{neighbor}. We specified k_{neighbor} as $k_{\text{neighbor}} = 10$ in the following experiments for the cellular genetic algorithms.

(2) Effect of the local search procedure

We examined the effect of the hybridization with the local search procedure. We applied the MOGA, the MOGLS, the C-MOGA, and the C-MOGLS to the test problems in the same manner as in the previous experiments. Simulation results are shown in Table 3. We can see that the introduction of the local search procedure improved the quality measure of Esbensen ("Quality" and "SD of Q" in Table 3). Nevertheless the best survival rate (i.e. B/A) was obtained by the C-MOGA. This may be because the local search procedure mainly worked for expanding the range of solution sets as suggested by the last column of Table 3.

Table 1: Comparison of MOGA with C-MOGA.

	A	B	B/A	Quality	SD of Q	D
MOGA	14.6	4.2	0.296	-1065.2	66.0	1512.4
C-MOGA	15.9	13.6	0.857	-990.0	36.1	1342.2

A: The number of nondominated solutions of the method.
B: The number of nondominated solutions that are not dominated by the other solutions obtained by the other method.
Quality: Set quality measure of Esbensen.
SD of Q: Standard deviation of Quality.
D: Euclid distance between two extreme solutions of the method.

Table 2: Effect of the choice of k_{neighbor} in C-MOGA.

	6	10	14	20	40
Quality	-1004.4	-990.0	-986.8	-985.8	-986.9

Table 3: Effect of the local search.

	A	B	B/A	Quality	SD of Q	D
MOGA	14.6	2.9	0.211	-1065.2	66.0	1512.4
MOGLS	14.0	3.4	0.257	-971.5	32.4	1829.4
C-MOGA	15.9	8.4	0.540	-990.0	36.1	1342.2
C-MOGLS	17.1	7.0	0.427	-964.0	29.4	1854.5

Table 4: Effect of immigration.

	A	B	B/A	Quality	SD of Q	D
C-MOGA	15.9	4.4	0.290	-990.0	36.1	1342.2
C-MOGLS	17.1	3.7	0.226	-964.0	29.4	1854.5
CI-MOGA	17.5	10.6	0.613	-967.5	30.7	1269.2
CI-MOGLS	19.0	6.1	0.328	-962.0	29.9	2094.9

(3) Effect of the immigration procedure.

We examined the performance of C-MOGA, C-MOGLS, CI-MOGA, and CI-MOGLS using the same test problems. Simulation results are summarized in Table 4. We can see that the immigration procedure is effective because the overall performance of the CI-MOGA and CI-MOGLS are better than those of the C-MOGA and C-MOGLS, respectively. The best quality value and the best distance value were obtained by the CI-MOGLS while the best survival rate (i.e. B/A) was obtained by the CI-MOGA.

7 CONCLUSION

In this paper, we proposed a cellular multi-objective genetic local search (C-MOGLS) algorithm, which is an extension of a multi-objective genetic local search (MOGLS) algorithm in our former study (Ishibuchi & Murata 1998). In the proposed C-MOGLS algorithm, each individual is located in a cell with a different weight vector. This weight vector governs the selection operation and the local search procedure at that cell. The selection is performed in the neighborhood of each cell. We also extended the proposed C-MOGLS algorithm by introducing a relocation procedure for assigning an appropriate local search direction to each solution. The effectiveness of the proposed algorithms was demonstrated by computer simulations on two-objective flowshop scheduling problems. The implementation of the proposed algorithms for optimization problems with three or more objectives is left for future work.

References

R.L.Daniels and R.J.Chambers (1990). Multiobjective flow-shop scheduling. *Naval Research Logistics* **37**: 981-995.

L. A. Dugatkin and M. Mesterton-Gibbons (1996). Cooperation among unrelated individuals: Reciprocal altruism, by-product mutualism and group selection in fishes. *BioSystems*, **37**: 19-30.

H.Esbensen (1996). Defining solution set quality. *Memorandum* (No.UCB/ERL M96/1, Electric Research Laboratory, College of Engineering, University of California, Berkeley, USA, January, 1996).

C. M. Fonseca and P. J. Fleming (1995). An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary Computation* 3: 1-16.

D.E.Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

P. Grim (1996). Spatialization and greater generosity in the stochastic Prisoner's Dilemma. *BioSystems*, **37**: 3-17.

J.Horn, N.Nafpliotis and D.E.Goldberg (1994). A niched Pareto genetic algorithm for multi-objective optimization. *Proc. of 1st IEEE International Conference on Evolutionary Computation*: 82-87.

H.Ishibuchi and T.Murata (1998). A multi-objective genetic local search algorithms and its application to flowshop scheduling. *IEEE Trans. on System, Man, and Cybernetics, Part C,* **28** (3): 392-403.

H. Ishibuchi, T. Nakari, and T. Nakashima (2000). Evolution of Strategies in Spatial IPD Games with Structured Demes, *Proc. of GECCO-2000* (in this proceedings).

S.M.Johnson (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly* **1** (1): 61-68.

F.Kursawe (1991). A variant of evolution strategies for vector optimization. In H.-P.Schwefel and R.Männer (Eds.), *Parallel Problem Solving from Nature*. 193-197. Berlin: Springer-Verlag.

B.Manderick and P.Spiessens (1989). Fine-grained parallel genetic algorithms. *Proc. of 3rd International Conference on Genetic Algorithms*: 428-433.

T.Murata and H.Ishibuchi (1995). MOGA: Multi-objective genetic algorithms. *Proc. of 2nd IEEE International Conference on Evolutionary Computing*: 289-294.

T.Murata, H.Ishibuchi, and M.Gen (1999). Specification of local search directions in genetic local search algorithms for multi-objective optimization problems. *Proc. of the Genetic and Evolutionary Computation Conference 1999*: 441-448.

M. A. Nowak and M. May (1992). Evolutionary games and spatial chaos. *Nature*, **359**: 826-859.

J.D.Schaffer (1985). Multi-objective optimization with vector evaluated genetic algorithms. *Proc. of 1st International Conference on Genetic Algorithms*: 93-100.

M. Oliphant (1994). 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): 349-352.

D. Whitley (1993). Cellular Genetic Algorithms. *Proc. of 5th International Conference on Genetic Algorithms*: 658.

D. S. Wilson (1977). Structured demes and the evolution of group-advantageous traits. *The American Naturalist*, **111** (977): 157-185.

D. S. Wilson, G. B. Pollock, and L. A. Dugatkin (1992). Can altruism evolve in purely viscous populations? *Evolutionary Ecology*, **6**: 331-341.

E. Zitzler and L. Thiele (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto Approach. *IEEE Trans. on Evolutionary Computation* 3: 257-271.