Effect of Local Search on the Performance of Cellular Multi-Objective Genetic Algorithms for Designing Fuzzy Rule-based Classification Systems

Tadahiko Murata[*], Hiroyuki Nozawa[**], Yasuhiro Tsujimura[+], Mitsuo Gen[**], Hisao Ishibuchi[++]

*: Kansai University, 2-1-1 Ryozenji-cho, Takatsuki, Osaka 569-1095, Japan

**: Ashikaga Institute of Technology, 268-1 Omae-cho, Ashikaga, Tochigi 326-8558, Japan

+: Nippon Institute of Technology, 4-1 Gakuendai, Miyashiro-cho, Minami-Saitama, Saitama 345-8501, Japan

++: Osaka Prefecture University, 1-1 Gakuen-cho, Sakai, Osaka 599-8531, Japan

Abstract - We show how local search can be combined with cellular multi-objective genetic algorithms for designing fuzzy rule-based classification systems. For achieving a good balance between genetic search and local search, local search is applied to only non-dominated solutions in each generation. Simulation results show the effectiveness of our approach.

I. INTRODUCTION

Since Schaffer [1] proposed the Vector Evaluated Genetic Algorithm (VEGA), evolutionary algorithms have been applied to various multi-objective optimization problems for finding their Pareto-optimal solutions (for reviews, see [2] and [3]). Let us consider the following n-objective maximization problem:

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

$$\text{Subject to} \quad \mathbf{x} \in \mathbf{X}, \tag{2}$$

where $\mathbf{z}$ is the objective vector, $\mathbf{x}$ is the decision vector, and $\mathbf{X}$ is the feasible region in the decision space. If the following solution $\mathbf{x}^*$ is found in the feasible region $\mathbf{X}$, $\mathbf{x}^*$ can be said as the optimal solution for the n-objective optimization problem:

$$f_i(\mathbf{x}^*) \geq f_i(\mathbf{x}) \quad \text{for} \quad \forall i \in \{1,2,...,n\} \quad \text{and} \quad \forall \mathbf{x} \in \mathbf{X}. \tag{3}$$

Usually, there is no optimal solution $\mathbf{x}^*$. Thus the task of multi-objective evolutionary algorithms is not to find a single final solution but to find a number of solutions that are not dominated by any other solutions. Let $\mathbf{a}$ and $\mathbf{b}$ are two decision vectors ($\mathbf{a}, \mathbf{b} \in \mathbf{X}$). Then $\mathbf{b}$ is said to be dominated by $\mathbf{a}$ if and only if the following two conditions hold:

$$f_i(\mathbf{a}) \geq f_i(\mathbf{b}) \quad \text{for} \quad \forall i \in \{1,2,...,n\}, \tag{4}$$

$$f_i(\mathbf{a}) > f_i(\mathbf{b}) \quad \text{for} \quad \exists i \in \{1,2,...,n\}. \tag{5}$$

When $\mathbf{b}$ is not dominated by any other solutions in $\mathbf{X}$, $\mathbf{b}$ is said to be a Pareto-optimal solution. That is, $\mathbf{b}$ is a Pareto-optimal solution when there is no solution $\mathbf{a}$ in $\mathbf{X}$ that satisfies the above two conditions.

The task of multi-objective evolutionary algorithms is to find Pareto-optimal solutions as many as possible. In the case of large-scale problems, it is impractical to try to find true Pareto-optimal solutions. Thus non-dominated solutions among examined ones are presented to decision makers as a result of the search by multi-objective evolutionary algorithms. In this case, multi-objective evolutionary algorithms try to drive a population to true Pareto-optimal solutions as close as possible.

In this paper, we introduce local search into a cellular multi-objective genetic algorithm (C-MOGA) [4, 5], which is a kind of genetic algorithms for multi-objective optimization problems. We refer to a C-MOGA with local search as a cellular multi-objective genetic local search (C-MOGLS) algorithm. We have already shown the effectiveness of the C-MOGLS on flowshop scheduling problems [6]. In this paper, we employ the C-MOGLS for designing fuzzy rule-based classification systems for pattern classification problems. In our former C-MOGLS, local search is applied to all individuals in every generation. If we simply combine local search with genetic algorithms, almost all computation time is spent by local search. Thus the contribution of genetic search is very small in hybrid algorithms. In order to achieve a good balance between genetic search and local search, we have already proposed an idea of restricting the number of examined neighborhood solutions by local search in [6,7]. While this idea was very effective in the application of the C-MOGLS to scheduling problems, it does not work very well on the design of fuzzy rule-based classification systems. In this paper, we propose a modified version of the C-MOGLS where local search is applied to only non-dominated individuals in every generation while it was applied to all individuals in our former studies. The effectiveness of this modification is clearly demonstrated through computer simulations on the design of fuzzy rule-based classification systems. Similar idea of selecting initial solutions for local search is proposed in [8]. The algorithm in [8] was compared to other recent EMO algorithms on a two-objective flowshop scheduling problem and its effectiveness was clearly shown.

II. MULTI-OBJECTIVE OPTIMIZATION BY CELLULAR GENETIC ALGORITHMS

The concept of cellular genetic algorithms was proposed by Whitley [9]. In cellular genetic algorithms, each

0-7803-7282-4/02/$10.00 ©2002 IEEE

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 and Spiessens [10]. Since each cell in our C-MOGA is related to a uniformly generated weight vector, it is allocated in an n-dimensional weight space of an n-objective optimization problem. In this section, we first describe a multi-objective genetic algorithm (MOGA) [11]. Then the C-MOGA and the C-MOGLS are shown as extensions of the MOGA.

A. Multi-Objective Genetic Algorithm

In this subsection, we show the outline of a multi-objective genetic algorithm [11], which is the basic algorithm of the C-MOGA and C-MOGLS. There are two main characteristic features in the MOGA. One is that the weighted sum of the n objectives is used as a fitness function:

$$f(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_n f_n(\mathbf{x}), \qquad (6)$$

where $w_1, \ldots, w_n$ are nonnegative weights for the n objectives. A different set of weights is used in each selection of parent individuals. Therefore N_{pop} sets of weights are employed when N_{pop} pairs of individuals are selected as parents in each generation. 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 to store two different sets of individuals during its execution of the algorithm. One is a current population and the other is a tentative set of non-dominated 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 non-dominated 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 non-dominated 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 non-dominated solutions is updated at every generation in our MOGA. By this trick our MOGA can preserve non-dominated solutions on Pareto front with concave surface.

From the tentative set of non-dominated solutions, a few solutions are randomly selected and their copies are added to the current population (see Figure 1). The randomly selected non-dominated solutions can be viewed as elite solutions because they are added to the current population with no modification.

B. Cellular Multi-Objective Genetic Algorithms

Next we show how to introduce the cellular concept into the MOGA described in the previous subsection. In cellular algorithms, each individual (i.e., 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-dimensional weight space. Figure 2 shows an example of structured cells for a two-objective 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 11 because a single individual exists in each cell. As shown in Figure 2, 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 Figure 2 are (10, 0), (9, 1), ..., (0, 10).

As shown in Figure 2, 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 proposed a weight specification method on

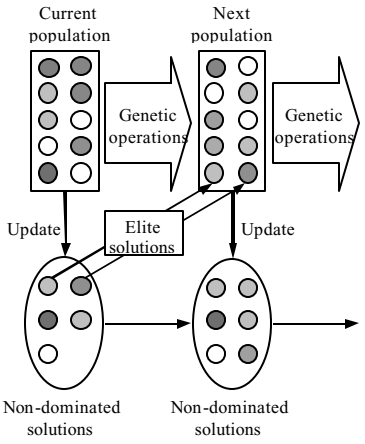

Figure 1. Two sets of solutions in the MOGA.

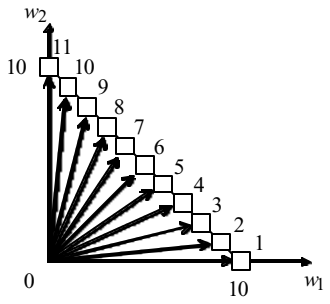

Figure 2. Location of each cell in the two-dimensional weight space.

an n-dimensional grid-world [5]. Let us consider weight vectors satisfying the following conditions.

$$w_1 + w_2 + \cdots + w_n = d\,, \tag{7}$$

$$w_i \in \{0,1,2,...,d\}\,. \tag{8}$$

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 (7) 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 Figure 2.

This weight specification method is easily used for the case with three or more objectives. For example, Figure 3 shows an example of the three-objective case where d is specified as $d = 4$. From Figure 3, 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) in Figure 4). By this weight specification method, we can uniformly distribute cells on the n-dimensional weight space. The number of cells generated for n-objective problems is calculated as follows:

$$N_2(d) = d + 1 \approx O(d)\,, \tag{9}$$

$$N_3(d) = \sum_{i=0}^{d} N_2(i) = \sum_{i=0}^{d} (i+1) = (i+1)(i+2)/2 \approx O(d^2)\,, \tag{10}$$

$$N_4(d) = \sum_{i=0}^{d} N_3(i) = \sum_{i=0}^{d} (i+1)(i+2)/2 \approx O(d^3)\,, \tag{11}$$

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

$$N_n(d) = \sum_{i=0}^{d} N_{n-1}(i) \approx O(d^{n-1})\,, \tag{12}$$

where $N_j(d)$, $j = 2,...,n$, is 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.

In the C-MOGA, genetic operations are applied to each individual that resides in a cell. In the C-MOGLS, a local search procedure is applied to each individual generated by genetic operations. Figure 4 shows the outline of the MOGLS.

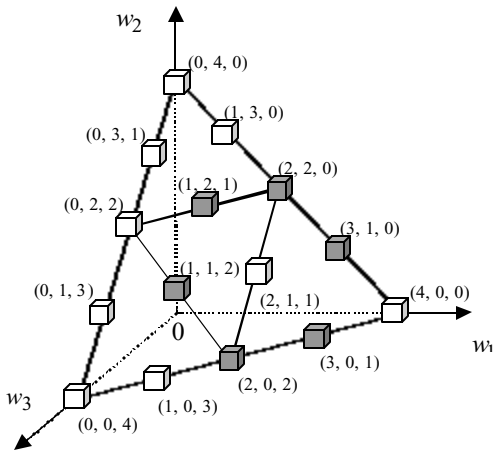

Figure 3. Location of each cell in the three-dimensional weight space by the proposed method.

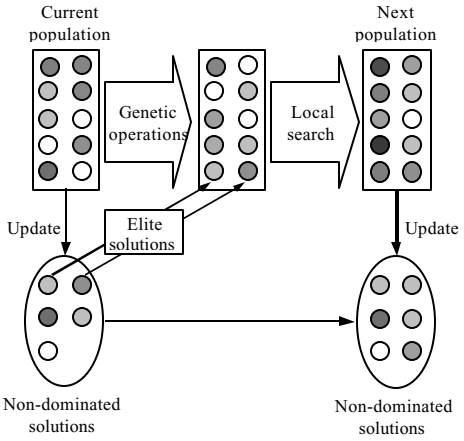

Figure 4. Local search procedure and two sets of solutions in the MOGLS.

III. INTRODUCTION OF LOCAL SEARCH PROCEDURE

Our C-MOGLS was successfully applied to scheduling problems [6]. A local search was employed to every individual in order to improve its fitness value defined by the weights in the cell where that solution resides. When we compare the C-MOGLS with the C-MOGA fairly, we should apply each algorithm to test problems under the same computation time or the same number of examined solutions during the execution of the algorithm. In either case, the number of generation updates by genetic operations in the C-MOGLS decreases since a local search consumes the computational resource. In general, when we introduce a local search procedure to a genetic algorithm, we need to find a good balance between the global search by genetic operations and the local search. In order to cope with this issue, we restricted the number of examined

neighborhood solutions in a local search procedure in [7]. We initially employ the same strategy for local search in the C-MOGLS for designing classification systems.

IV. FUZZY CLASSIFICATION SYSTEMS

In this paper, we apply the C-MOGLS to multi-objective classification problems. The aim of classification problems in this paper is to design classification systems which classify a pattern vector with several attributes into one of the already-known classes. Therefore, a classification system can be viewed as a mapping from a pattern vector in a multi-dimensional pattern space to a class label. Fuzzy theory [12], neural networks [13, 14], decision trees [15] are often employed to design classification systems. We try to design classification systems in the view of the following aspects:

1) *Learning ability*: The classification ability on training patterns that were used for the design of classification systems.
2) *Generalization ability*: The classification ability on test patterns that were not used for the design of classification systems. This performance is usually measured by dividing the given patterns into two subsets: training patterns and test patterns. Then employ the training patterns for designing classification systems, and apply the obtained systems to the test patterns.
3) *Interpretability*: A criterion of comprehensibility of classification systems. Each rule should be understood easily by users or decision makers.

The learning ability and the generalization ability are related to the classification performance of classification systems. On the other hand, the interpretability is related to understandability of classification systems. This aspect of designing classification systems may help users and decision makers to see input-output relations. Therefore we can regard the designing of classification systems with the high understandability as one of data mining tools [16, 17].

In this paper, we design fuzzy classification systems from the views of the described criteria except for the generalization ability. A fuzzy classification system consists of several fuzzy if-then rules in the following format:

$$\begin{aligned} &\text{If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_m \text{ is } A_{jm} \\ &\text{then Class } C_j \text{ with } CF_j, \quad j = 1,2,\ldots,N, \end{aligned} \tag{13}$$

where $\mathbf{x} = (x_1,\ldots, x_m)$ is a pattern vector with m attributes, A_{ji} is a fuzzy set related to a linguistic term (see Figure 5) of the i-th attribute of the j-th rule, C_j is a consequent class, and CF_j is a certainty factor of the rule. We employ a heuristic method to calculate the consequent class C_j and the certainty factor CF_j for each rule [12].

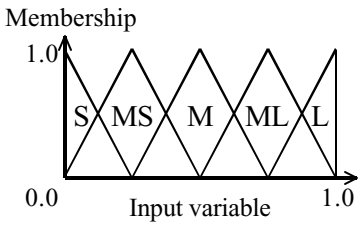

Figure 5. Linguistic fuzzy sets (S: *small*, MS: *medium small*, M: *medium*, ML: *medium large*, and L: *large*).

We employ the following objective functions related to the learning ability and the interpretability.

$$\text{Maximize} \quad f_1(S) = NCP(S), \tag{14}$$
$$\text{Minimize} \quad f_2(S) = |S|, \tag{15}$$
$$\text{Minimize} \quad f_3(S) = Length(S), \tag{16}$$

where S is a classification system, $NCP(S)$ is the number of correctly classified training patterns by S, $|S|$ is the number of fuzzy rules in S, and $Length(S)$ is the total number of antecedent conditions of all rules in S. $NCP(S)$ is related to the learning ability. As for the interpretability we employed two criteria: $|S|$ and $Length(S)$. $|S|$ and $Length(S)$ are related to the interpretability for a rule base, and for each rule, respectively. That is, a classification system with a small number of rules is easily understood by users or decision makers. Moreover we consider the interpretability of each rule. It is difficult for us to understand each rule with a lot of conditions. We try to find a better set of non-dominated solutions by genetic algorithms for multi-objective optimization problems.

V. SIMULATION RESULTS

In this paper, we try to find a better set of non-dominated solutions with respect to the three objectives in (14)-(16). We employ wine data and glass data as test problems to which the C-MOGLS is applied. The wine data set has 178 training patterns with 13 attributes and 3 classes. The glass data set has 214 patterns with 9 attributes and 6 classes.

We compare three algorithms with one another: MOGA, C-MOGA, and C-MOGLS on the two test problems. In the C-MOGLS, we reverse bit values as a local search operation. Since each bit corresponds to a fuzzy rule where "0" and "1" show the exclusion and the inclusion of the corresponding rule in the classification system S, the bit reverse local search tries to find a system including another rule with the better classification performance, or to find a system excluding a rule with the same or the better classification ability. As we have already described in Section III, we restricted the number of examined solutions in our local search procedure. Let us denote the number of examined neighborhood solutions by a local search

0-7803-7282-4/02/$10.00 ©2002 IEEE

procedure for the current solution as k. If there is no better solution among k neighborhood solutions, the local search for the current solution is terminated. In this paper, we specified k as $k = 2$. That is, if there is no solution which improves the fitness function in two randomly selected neighbors of the current solution, the local search for the solution is terminated and the local search for another solution in the current generation is executed.

We specified the parameter d in the C-MOGA and the C-MOGLS as $d = 13$. Thus, the number of the cells in the cellular algorithms was 105. This parameter was most effective in simulations on three objective flowshop scheduling problems in [5]. In order to compare the MOGA with the cellular algorithms under the same conditions, we specified the population size as 105 in the MOGA. In the MOGA and the C-MOGA, we specified the number of the generation updates as 1,000. Therefore $105 \times 1000 = 105{,}000$ classification systems were searched by those algorithms. In the C-MOGLS, we stopped the algorithms when 105,000 classification systems were examined. We compared three sets of non-dominated solutions obtained by the three algorithms from the same initial population. We generated ten different initial populations for each algorithm, and then obtained averaged results.

In order to compare three algorithms, we show the averaged number of obtained non-dominated solutions for each algorithm, and the number of solutions which are not dominated by the non-dominated solutions obtained by other algorithms. TABLE I shows the simulation results for the wine data and the glass data. From TABLE I, we can see that the number of obtained solutions and the number of the non-dominated solutions by the C-MOGA were the best of the three algorithms. On the other hand, we can find the cases that the performance of the C-MOGLS is worse than that of the MOGA. Thus, we can find that the introduction of the local search procedure into the C-MOGA was not effective in this case.

In order to find a good balance between global genetic search and local search, we specified the parameter k as $k = 1$ to examine more systems by genetic operations. The results are shown in TABLE II. From this table, we can find that the performance of the C-MOGLS is slightly improved, but still worse than that of the MOGA. In order to consider the reason why the performance of the C-MOGLS is worse than that of the MOGA, we examined the number of generation updates in the algorithms. TABLE III shows the averaged number of generation updates. From the table, we can see that the number of generation updates in the C-MOGLS with $k = 2$ is 1/3 of those in the MOGA and the C-MOGA. In the case of $k = 1$, the number of generation updates is less than half of 1,000 generations. In our previous study [7], 1/5 generation updates of the MOGLS were enough for the flowshop scheduling problems. Therefore we can see that a good balance between global genetic search and local search should be found for each problem carefully.

In order to examine more solutions by genetic operations, we apply the local search procedure only to non-dominated solutions. As the weight vector in the local search procedure, we used the weight vector of the cell from which the non-dominated solution was obtained. We specified the number of examined neighborhood solutions k in the modified MOGLS as $k = 1$. TABLE IV shows simulation results. We can see that the best performance is obtained by the modified C-MOGLS in this table.

TABLE I. MOGA, C-MOGA, and C-MOGLS ($k = 2$).

Data set	Wine		Glass	
	Obtained solutions	Non-dominated	Obtained solutions	Non-dominated
MOGA	11.6	7.8	18.9	17.4
C-MOGA	12.4	12.0	21.1	18.9
C-MOGLS	12.2	5.5	18.6	12.3

TABLE II. MOGA, C-MOGA, and C-MOGLS ($k = 1$).

Data set	Wine		Glass	
	Obtained solutions	Non-dominated	Obtained solutions	Non-dominated
MOGA	11.6	7.5	18.9	17.2
C-MOGA	12.4	12.0	21.1	18.8
C-MOGLS	12.0	8.0	18.7	13.4

TABLE III. Generation updates by MOGA, C-MOGA, and C-MOGLSs.

	Wine	Glass
MOGA	1000	1000
C-MOGA	1000	1000
C-MOGLS ($k = 2$)	317.4	299.8
C-MOGLS ($k = 1$)	471.5	434.6

TABLE IV. MOGA, C-MOGA, and Modified C-MOGLS.

Data set	Wine			Glass		
	OS	NS	G	OSs	NS	G
MOGA	11.6	7.2	1000	18.9	15.8	1000
C-MOGA	12.4	11.0	1000	21.1	16.6	1000
M-C-MOGLS	14.2	11.5	835.8	25.9	23.3	793.9

OS: # of Obtained Solutions,
NS: # of Non-dominated Solutions,
G: # of Generation updates.

Figures 6 and 7 show the number of non-dominated solutions among the three sets of non-dominated solutions. The horizontal axis of each figure is the number of examined solutions by the algorithm. From Figure 6 for the wine data, we can see that the number of non-dominated solutions of the modified C-MOGLS is large in an early

stage of evolution. Then the result of the C-MOGA approaches that of the proposed method. This is because non-dominated solutions by the C-MOGA will be similar to those of the proposed method. On the other hand, we can see that the effectiveness of the local search is significant for the glass data since the number of non-dominated solutions increases in the latter stage of the horizontal axis.

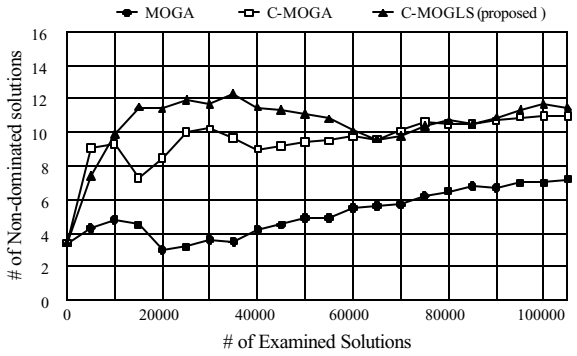

Figure 6. The number of non-dominated solutions of each algorithm over the number of examined solutions (Wine).

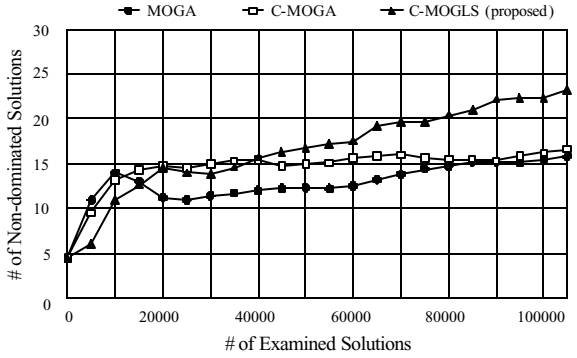

Figure 7. The number of non-dominated solutions of each algorithm over the number of examined solutions (Glass).

VI. CONCLUSION

In this paper, we considered how to introduce a local search procedure into the cellular multi-objective genetic algorithm (C-MOGA) [4]. In our previous study in [10], we obtained better results by applying a local search procedure to all individuals in each generation for scheduling problems. However, we did not obtain better results by the same hybridization scheme of the local search procedure for pattern classification problems. This may be because the balance between genetic global search and local search was not good for the application to pattern classification problems. Therefore, we applied the local search procedure only to non-dominated solutions. Simulation results show the effectiveness of this modification.

REFERENCES

[1] J.D. Schaffer, "Multi-objective optimization with vector evaluated genetic algorithms," *Proc. of 1st International Conference on Genetic Algorithms and Their Applications*, pp. 93-100 (1985).

[2] C.A. Coello Coello, "A comprehensible survey of evolutionary-based multi-objective optimization techniques," *Knowledge and Information Systems*, vol. 1, no. 3, pp. 269-308 (1999).

[3] D.A. Van Veldhuizen and G.B. Lamont, "Multi-objective evolutionary algorithms: Analyzing the state-of-art," *Evolutionary Computation*, vol. 8, no. 2, pp. 125-147 (2000).

[4] T. Murata, M. Gen, "Cellular genetic algorithm for multi-objective optimization," *Proc. of The Fourth Asian Fuzzy System Symposium*, pp. 538-542 (2000).

[5] T. Murata, H. Ishibuchi, M. Gen, "Specification of genetic search directions in cellular multi-objective genetic algorithms," *Proc. of First International Conference on Evolutionary multi-criterion optimization*, pp. 82-95 (2001).

[6] T. Murata, H. Ishibuchi, M. Gen, "Cellular genetic local search for multi-objective optimization," *Proc. of The Genetic and Evolutionary Computation Conference 2000*, pp. 307-314 (2000).

[7] H. Ishibuchi, T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transaction on Systems, Man, and Cybernetics*, Part C, vol. 28, no.3, pp.392-403 (1998).

[8] H. Ishibuchi, T. Yoshida, T. Murata, "Selection of initial solutions for local search in multiobjective genetic local search," *Proc. of 2002 Congress on Evolutionary Computation*, in this proceedings.

[9] D. Whitley, "Cellular genetic algorithms," *Proc. of 5th International Conference on Genetic Algorithms*, p.658 (1993).

[10] B. Manderick, P. Spiessens, "Fine-grained parallel genetic algorithms," *Proc. of 3rd International Conference on Genetic Algorithms*, pp. 428-433 (1989).

[11] T. Murata, H. Ishibuchi, "MOGA: Multi-objective genetic algorithms," *Proc. of The 2nd IEEE International Conference on Evolutionary Computing*, pp. 289-294 (1995).

[12] H Ishibuchi, K. Nozaki, H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification," *Fuzzy Sets and Systems*, vol. 52, pp. 21-32 (1992).

[13] D. Hammerstrom, "Neural networks at work," *IEEE Spectrum*, June, pp. 26-32 (1993).

[14] D. Hammerstrom, "Working with neural networks," *IEEE Spectrum*, July, pp. 46-53 (1993).

[15] J.R. Quinlan, "Introduction of decision trees," *Machine Learning*, vol. 1, pp. 71-99 (1985).

[16] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, "Knowledge discovery and data mining: Towards a unifying framework," *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining*, 82-88 (1996).

[17] H. Ishibuchi, T. Yamamoto, and T. Nakashima, "Fuzzy data mining: Effect of fuzzy discretization," *Proc. of 1st IEEE International Conference on Data Mining*, pp. 241-248 (2001).