Multi-Objective Genetic Local Search for Minimizing the Number of Fuzzy Rules for Pattern Classification Problems

Hisao Ishibuchi
Department of Industrial Engineering
Osaka Prefecture University
Sakai, Osaka 599-8531, Japan
hisaoi@ie.osakafu-u.ac.jp

Tadahiko Murata
Department of Industrial & Systems Engineering
Ashikaga Institute of Technology
Ashikaga, Tochigi 326-8558, Japan,
murata@genlab.ashitech.ac.jp

Abstract

For constructing compact fuzzy rule-based systems with high classification performance, we have already formulated a rule selection problem. Our rule selection problem has two objectives: to minimize the number of selected fuzzy if-then rules (*i.e.*, to minimize the fuzzy rule base) and to maximize the number of correctly classified patterns (*i.e.*, to maximize the classification performance). In this paper, we apply single-objective and multi-objective genetic local search algorithms to our rule selection problem. High performance of those hybrid algorithms is demonstrated by computer simulations on multi-dimensional pattern classification problems in comparison with genetic algorithms in our former studies. It is shown in computer simulations that local search procedures can improve the ability of genetic algorithms to search for a compact rule set with high classification performance.

1. Introduction

Genetic algorithms [1] have been applied to various optimization problems [2,3]. The main advantage of genetic algorithms over hill-climbing techniques is their global search ability. That is, genetic algorithms are not usually stuck into local optima. The suitability for parallel implementation is another advantage of genetic algorithms. While many successful results have been reported for various application areas in the literature, several studies [4-6] pointed out that the performance of genetic algorithms was inferior to other search algorithms such as simulated annealing and taboo search for some combinatorial optimization problems. This is because genetic algorithms do not have the local search ability. Recent studies [7-10] suggested that the performance of genetic algorithms could be improved by combining local search procedures.

Genetic algorithms have been employed for generating fuzzy if-then rules and tuning membership functions (for example, see [11-19]). We proposed a GA-based rule selection method [20,21] where a small number of fuzzy if-then rules were selected from a large number of candidate rules for pattern classification problems. Genetic algorithms were used for constructing compact fuzzy rule-based systems with high classification performance. The GA-based method in [20,21] was extended to a multi-objective genetic algorithm for finding non-dominated solutions of the rule selection problem with two objectives: to minimize the number of selected rules and to maximize the number of correctly classified patterns [22,23].

The aim of this paper is to show how the performance of our single-objective and two-objective genetic algorithms can be improved by combining local search procedures. In this paper, we first describe a single-objective genetic algorithm for the rule selection. Next we show a simple heuristic procedure and an iterative local search procedure, which are combined with the genetic algorithm to construct a genetic local search algorithm. Then we propose a modified version of the local search procedure for decreasing the computation time of our genetic local search algorithm. Finally our genetic local search algorithm is extended to the case of the two-objective rule selection problem. High performance of our single-objective and two-objective hybrid algorithms is illustrated by computer simulations on real-world pattern classification problems (*i.e.*, iris data with four attributes, and wine data with 13 attributes).

2. GA-based fuzzy rule selection

2.1. Pattern classification problem

Let us consider a c-class pattern classification problem in an n-dimensional pattern space $[0,1]^n$. We assume that m training patterns $\mathbf{x}_p = (x_{p1}, \ldots, x_{pn})$, $p = 1,2,\ldots,m$, are given from the c classes ($c << m$). In computer simulations of this paper, attribute values are normalized into the unit interval [0,1].

2.2. Fuzzy if-then rules for pattern classification

In this paper, we use fuzzy if-then rules of the following type for our pattern classification problem:

$$\text{Rule } R_j : \text{If } x_{p1} \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_{pn} \text{ is } A_{jn} \text{ then Class } C_j \text{ with } CF = CF_j, \quad (1)$$

where R_j is the label of the j-th rule, $A_{j1},\ldots,A_{jn}$ are antecedent fuzzy sets, C_j is the consequent class, and CF_j is the grade of certainty. Membership functions of the antecedent fuzzy sets are to be specified by domain experts, and their specifications depend on characteristic features of each pattern classification problem. In computer simulations, we use three linguistic values and "*don't care*" in Figure 1 as the antecedent fuzzy sets.

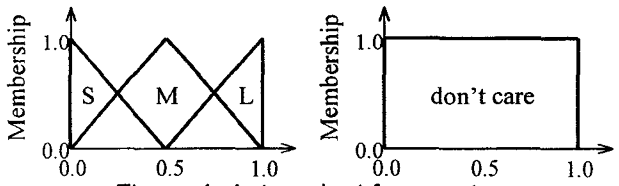

Figure 1. Antecedent fuzzy sets.
(S: *small*, M: *medium*, and L: *large*)

The consequent class C_j and the grade of certainty CF_j of each fuzzy if-then rule in (1) are determined by training patterns in the fuzzy subspace specified by the antecedent fuzzy sets $A_{j1},\ldots,A_{jn}$ (see [20-23]).

Let us denote a set of fuzzy if-then rules by S. The rule set S can be also viewed as a fuzzy rule base and a fuzzy rule-based classification system. When a new pattern $\mathbf{x}_p = (x_{p1},\ldots,x_{pn})$ is presented to the rule set S, $\mathbf{x}_p$ is classified by a single winner rule, which is defined as follows [20-23]:

$$\mu_{j^*}(\mathbf{x}_p) \cdot CF_{j^*} = \max\{\mu_j(\mathbf{x}_p) \cdot CF_j \mid R_j \in S\}, \quad (2)$$

where

$$\mu_j(\mathbf{x}_p) = \mu_{j1}(x_{p1}) \cdot \mu_{j2}(x_{p2}) \cdot \ldots \cdot \mu_{jn}(x_{pn}). \quad (3)$$

2.3. Rule selection problem

For constructing a compact fuzzy rule-based system with high classification performance, the following rule selection problem was formulated [20-23]:

$$\text{Maximize } NCP(S) \text{ and minimize } |S|, \quad (4)$$

where $NCP(S)$ is the number of correctly classified patterns by the rule set S, and $|S|$ is the number of fuzzy if-then rules in S. The performance of the rule set S and its compactness are measured by $NCP(S)$ and $|S|$, respectively, in the rule selection problem.

2.4. Genetic algorithm for the rule selection

The two objectives in our rule selection problem were combined using non-negative weights W_{NCP} and W_S into the following scalar fitness function in the single-objective genetic algorithm in our former work [20,21]:

$$fitness(S) = W_{NCP} \cdot NCP(S) - W_S \cdot |S|. \quad (5)$$

Let us assume that r fuzzy if-then rules are given as candidate rules for our rule selection problem. A subset S of those candidate rules is represented by a bit string as $S = s_1 s_2 s_3 \ldots s_r$ where $s_j = 1$ means that the j-th candidate rule is included in the rule set S, and $s_j = 0$ means that the j-th candidate rule is not included. In the genetic algorithm, first a number of strings are randomly generated as an initial population. Then the population update is iterated using genetic operations such as selection, crossover, mutation, and elitist strategy. In computer simulations, we used the roulette wheel selection with the linear scaling, the uniform crossover, and the standard bit mutation.

2.5. Computer simulations

We applied our GA-based rule selection method to the well-known iris data. The iris data set is a four-dimensional three-class pattern classification problem with 150 samples. Because we have four antecedent fuzzy sets in Figure 1 for each attribute, the total number of possible combinations of antecedent fuzzy sets is $4^4 = 256$. Using those combinations, we generated 222 fuzzy if-then rules as candidate rules. The other 34 fuzzy if-then rules were not generated because the consequent class of each of those rules could not be uniquely specified. The genetic algorithm with the following parameter specifications was employed for selecting a small number of fuzzy if-then rules.

Population size: 20,
Crossover probability: 1.0,
Mutation probability: 0.1 for the mutation from 1 to 0,
0.001 for the mutation from 0 to 1,
Weights in the fitness function: $W_{NCP} = 10$, $W_S = 1$,
Stopping condition: Evaluation of 20,000 rule sets.

The genetic algorithm selected four fuzzy if-then rules that can correctly classify 146 patterns (97.3% of the given 150 patterns). The mutation probability was biased for efficiently decreasing the number of fuzzy if-then rules. When the mutation probability was not biased, good results were not obtained. For example, the genetic algorithm selected 81 fuzzy if-then rules when we specified the mutation probability as 0.01.

3. Genetic local search for the rule selection

3.1. Simple heuristic procedure

In this subsection, we show that the performance of the genetic algorithm can be improved by combining a simple heuristic procedure. Because each pattern is classified by a single winner rule in a rule set S, some fuzzy if-then rules in S have no contribution to the classification (*i.e.*, they classify no patterns). Those fuzzy if-then rules can be removed from the rule set S with no deterioration of the classification performance. A simple heuristic procedure is to remove those fuzzy if-then rules from each rule set (*i.e.*, from each string).

The genetic algorithm with the heuristic procedure selected four fuzzy if-then rules that can correctly classify 146 patterns (97.3% of the given 150 patterns). While the final result was not improved, the heuristic procedure accelerated the evolution by the genetic algorithm. In Figure 2, we show the number of fuzzy if-then rules included in the best string at each generation in the early stage of the evolution by each algorithm. From this figure, we can see that the heuristic procedure had a large effect on the decrease of the number of fuzzy if-then rules. Because the evaluation of compact rule sets does not require long computation time, the heuristic procedure can decrease not only the number of fuzzy if-then rules but also computation time. For example, computation time required for 50 generations (*i.e.*, 1000 evaluations of rule sets) was 26.9 (sec.) with the heuristic procedure while it was 58.3 (sec.) for the genetic algorithm without the heuristic procedure.

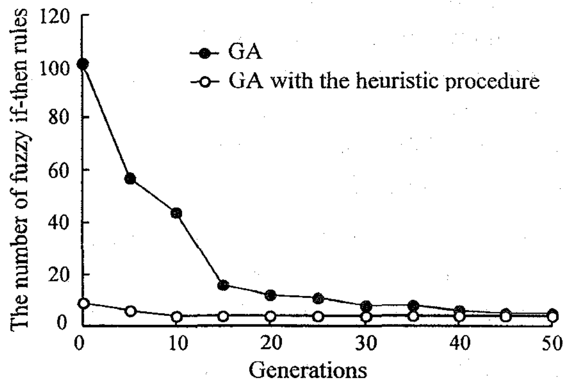

Figure 2. The number of fuzzy if-then rules included in the best string at each generation.

3.2. Genetic local search algorithm

The above heuristic procedure is a special kind of local search. Formally, a local search procedure to maximize the fitness function $fitness(S)$ can be written as follows:

Step 1: Specify an initial solution S.
Step 2: Examine the fitness value of a randomly selected neighborhood solution S' of the current solution S.
Step 3: If $fitness(S') > fitness(S)$, then move to the neighborhood solution S' (*i.e.*, replace the current solution S with S') and return to Step 2.
Step 4: If all the neighborhood solutions of S have already been examined, then stop the local search procedure. Otherwise return to Step 2 to examine another neighborhood solution.

This local search procedure is combined with the genetic algorithm. In our genetic local search algorithm, the local search procedure is applied to each string in each population after the above heuristic procedure. Neighborhood solutions of the current solution S are generated by changing the value of a single bit of S. This means that the Hamming distance between the current solution S and its neighbors is 1. For example, neighborhood solutions of "0011" are "1011", "0111", "0001", and "0010".

Our genetic local search algorithm with the same parameter specifications as in Subsection 3.1 was applied to the iris data. This means that the computation load of the genetic local search algorithm was the same as the genetic algorithm. Seven rules with a 97.3% classification rate were selected by the genetic local search algorithm. This is not as good as the result by the genetic algorithm (*i.e.*, four rules with the same classification rate). That is, the performance of the genetic algorithm was deteriorated by combining the local search procedure. We discuss this deterioration of the performance in the next subsection.

3.3. Modification of the local search procedure

Because the local search procedure is applied to all strings generated by genetic operations, it usually spends long computation time in our genetic local search algorithm. In the rule selection problem, the number of neighborhood solutions is the same as the string length (*i.e.*, the number of candidate rules: r). This means that at least r neighborhood solutions are examined for each string generated by genetic operations in the genetic local search algorithm. Thus long computation time is spent by the local search procedure. This leads to a small number of generation updates because the total number of evaluated solutions was used as the stopping condition in our computer simulations. In fact, the number of generation updates was three in the genetic local search algorithm in Subsection 3.2 while it was 1000 in the genetic algorithm in Subsection 3.1. The main reason of the deterioration of the performance in the genetic local search algorithm is that the number of generation updates is not enough. The global search ability of the genetic algorithm was not utilized well in the genetic local search algorithm because the number of generation updates is small.

One strategy to reduce the computation time spent by the local search procedure is to restrict the number of examined neighborhood solutions [9,10]. That is, all the neighborhood solutions are not examined in our modified local search procedure. Let k ($k \leq r$) be the number of examined neighborhood solutions for each current solution. When a better solution is not found among k neighborhood solutions of a current solution, our modified local search procedure is terminated.

Using various values of k, we applied the genetic local search algorithm to the iris classification problem. In this computer simulation, k neighborhood solutions were randomly selected. Simulation results for training patterns are summarized in Table 1. In Table 1, $k = 0$ means no local search procedure (*i.e.*, the genetic algorithm with only the heuristic procedure). On the other hand, $k = 222$ (*i.e.*, $k = r$) means no modification of the local search. From this table, we can see that the number of generation updates was increased by using small values for k. Because the rule selection problem for the iris data is not difficult, the advantage of the genetic local search over the genetic algorithm is not clear in Table 1. The advantage will be clearly shown in computer simulations for the two-objective rule selection in the next section.

Our genetic local search algorithm was also applied to wine data (available from UC Irvine database). The wine data set is a three-class pattern classification problem with 13 continuous attributes. When we use four antecedent fuzzy sets in Figure 1 for each of the 13 attributes, the total number of possible combinations of the antecedent fuzzy sets is $4^{13} \approx 6.7 \times 10^7$. Because it is impossible to use such a large number of combinations for generating fuzzy if-then rules, we only generated fuzzy if-then rules with at least eleven "*don't care*" attributes in the antecedent part. That is, each of the generated fuzzy if-then rules has only one or two antecedent conditions. In this manner, 741 fuzzy if-then rules were generated.

Simulation results for training patterns of wine data are summarized in Table 2. From this table, we can see the performance of the genetic algorithm (*i.e.*, the result with $k = 0$) was improved by combining the local search procedure (see the result with $k = 5$).

Table 1. Simulation results by the genetic local search algorithm for the iris data.

The value of k	0	5	10	222
The number of rules	4	4	4	7
Classification rate (%)	97.3	97.3	97.3	97.3
CPU time (min.)	7.6	7.1	7.7	9.1
Generation updates	1000	127	66	3

Table 2. Simulation results by the genetic local search algorithm for the wine data.

The value of k	0	5	10	100
The number of rules	8	7	6	13
Classification rate (%)	100	100	99.4	100
CPU time (min.)	30.4	30.6	32.3	56.3
Generation updates	1000	116	63	3

4. Extension to the two-objective rule selection

4.1. Two-objective rule selection problem

In the previous sections, our rule selection problem was handled by combining its two objectives into a single scalar fitness function. In this section, our rule selection problem is handled as a two-objective optimization problem. That is, our aim is to find all the non-dominated solutions of the rule selection problem.

4.2. Two-objective genetic algorithm

In our former work [22,23], we applied a two-objective genetic algorithm to the rule selection problem for finding its non-dominated solutions. Population update in our two-objective genetic algorithm is illustrated in Figure 3. Main differences between our single-objective and two-objective genetic algorithms are as follows:

(1) The weight values in the fitness function are not constant in our two-objective genetic algorithm. They are randomly specified whenever a pair of parent strings are selected for the crossover operation.

(2) A tentative set of non-dominated solutions is separately stored from the current population in our two-objective genetic algorithm (see Figure 3).

(3) Multiple non-dominated solutions are used as elite solutions in our two-objective genetic algorithm (see Figure 3).

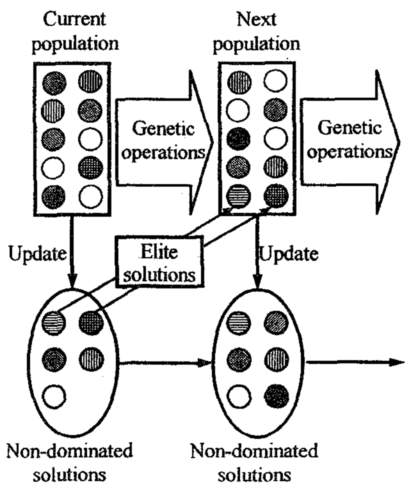

Figure 3. Generation update in the two-objective genetic algorithm.

4.3. Two-objective genetic local search

The performance of our two-objective genetic algorithm can be also improved by employing the simple heuristic procedure and the modified local search procedure. The local search for each string was performed to maximize the fitness function in (5) with the weight values used in the selection procedure of its parents. Because the weight values are randomly specified whenever a pair of parent strings are selected, each sting has its own local search direction. Various local search directions are necessary for obtaining non-dominated solutions with large variety [24].

4.4. Computer simulations

Our two-objective genetic algorithm and genetic local search algorithm were applied to the wine data. In computer simulations, the number of elite solutions (see Figure 3) was specified as 3. The value of k in the modified local search procedure was specified as $k = 5$. Non-dominated solutions obtained by each algorithm are summarized in Figure 4 and Table 3. From the simulation results in Figure 4 and Table 3, we can see that the heuristic procedure and the local search procedure improved the search ability of the genetic algorithm. That is, compact rule sets with higher classification rates were obtained by the genetic algorithm with the heuristic procedure and the genetic local search algorithm. The

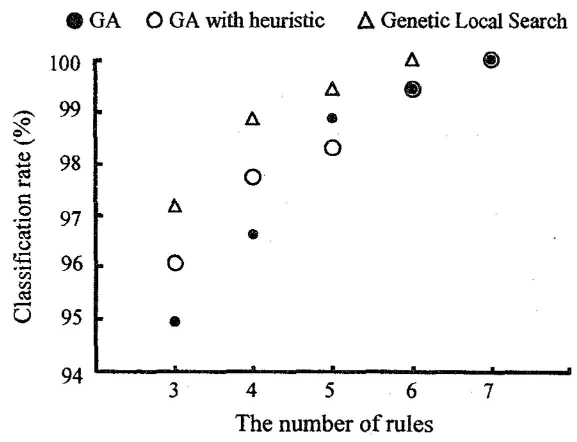

Figure 4. Non-dominated solutions obtained by each algorithm for the wine data.

Table 3. Classification rates of obtained rule sets by each algorithm for the wine data.

The number of rules	3	4	5	6	7
Genetic Algorithm	94.9	96.6	98.9	99.4	100
GA with heuristic	96.1	97.8	98.3	99.4	100
Genetic Local Search	97.2	98.9	99.4	100	---

CPU time of each trial (*i.e.*, 1000 generations of the genetic algorithm) in Table 3 was about 25-40 minutes on the same workstation used to obtain the results of Table 2.

5. Conclusion

In this paper, we demonstrated that the performance of the GA-based rule selection methods was significantly improved by the simple heuristic procedure and the local search procedure. These two procedures compensate the lack of the local search ability of genetic algorithms. Thus our hybrid algorithms have the local search ability as well as the global search ability. By computer simulations, we showed that a small number of fuzzy if-then rules with high classification performance were selected by our hybrid algorithms. The classification performance can be improved further by tuning the grade of certainty of each fuzzy if-then rule (see [25]). Because a small number of fuzzy if-then rules with linguistic interpretation are selected, our rule selection methods can be also viewed as a promising knowledge acquisition tool.

Reference

[1] J. H. Holland. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, 1989.

[3] L. Davis. *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[4] C. A. Glass *et al.* Genetic algorithms and neighborhood search for scheduling unrelated parallel machines, *Preprint Series*, No. OR47, University of Southampton, 1992.

[5] H. Ishibuchi *et al.* Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems, *Fuzzy Sets and Systems*, 67: 81-100, 1994.

[6] T. Murata, and H. Ishibuchi. Performance evaluation of genetic algorithms for flowshop scheduling problems, *Proc. 1st ICEC*: 812-817, 1994.

[7] P. Jog *et al.* The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem, *Proc. 3rd ICGA*: 110-115, 1989.

[8] N. L. J. Ulder *et al.* Genetic local search algorithms for the traveling salesman problem, in: H. -P. Schwefel and R. Manner (Eds.), *Parallel Problem Solving from Nature*, Springer-Verlag, Berlin: 109-116, 1991.

[9] P. Merz, and B. Freisleben. Genetic local search for the TSP: New results, *Proc. 4th ICEC*: 159-164, 1997.

[10] H. Ishibuchi *et al.* Effectiveness of genetic local search algorithms, *Proc. 7th ICGA*: 505-512, 1997.

[11] M. Valenzuela-Rendon. The fuzzy classifier system: A classifier system for continuously varying variables, *Proc. 4th ICGA*: 346-353, 1991.

[12] P. Thrift. Fuzzy logic synthesis with genetic algorithms, *Proc. 4th ICGA*: 509-513, 1991.

[13] H. Nomura *et al.* A self-tuning method of fuzzy reasoning by genetic algorithm, *Proc. International Fuzzy Systems and Intelligent Control Conference*: 236-245, 1992.

[14] C. L. Karr, and E. J. Gentry. Fuzzy control of pH using genetic algorithms, *IEEE Trans. on Fuzzy Systems*, 1 (1): 46-53, 1993.

[15] D. Park *et al.* Genetic-based new fuzzy reasoning models with application to fuzzy control, *IEEE Trans. on SMC*, 24 (1): 39-47, 1994.

[16] A. Homaifar, and E. McCormick. Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, *IEEE Trans. on Fuzzy Systems*, 3 (2): 129-139, 1995.

[17] F. Herrera *et al.* Tuning fuzzy logic controllers by genetic algorithms, *Approximate Reasoning*, 12: 299-315, 1995.

[18] K. Shimojima *et al.* Self-tuning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm, *Fuzzy Sets and Systems*, 71: 295-309, 1995.

[19] B. Carse *et al.* Evolving fuzzy rule based controllers using genetic algorithms, *Fuzzy Sets and Systems*, 80: 273-293, 1996.

[20] H. Ishibuchi *et al.* Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms, *Fuzzy Sets and Systems*, 65: 237-253, 1994.

[21] H. Ishibuchi *et al.* Selecting fuzzy if-then rules for classification problems using genetic algorithms, *IEEE Trans. on Fuzzy Systems*, 3: 260-270, 1995.

[22] H. Ishibuchi *et al.* Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems, *Fuzzy Sets and Systems*, 89 (2): 135-150, 1997.

[23] H. Ishibuchi and T. Murata. Minimizing the fuzzy rule base and maximizing its performance by a multi-objective genetic algorithm, *Proc. FUZZ-IEEE'97*: 259-264, 1997.

[24] H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling, *IEEE Trans. on SMC* (to appear).

[25] K. Nozaki *et al.* Adaptive fuzzy rule-based classification systems, *IEEE Trans. on Fuzzy Systems*, 4 (2): 238-250, 1996.