Multi-strategy Multimodal Genetic Algorithm for Designing Fuzzy Rule Based Classifiers

Vladimir Stanovov[1], Evgenii Sopov[2] and Eugene Semenkin[3]
Department of Systems Analysis and Operations Research
Siberian State Aerospace University
Krasnoyarsk, Russia
vladimirstanovov@yandex.ru[1], evgenysopov@gmail.com[2], eugenesemenkin@yandex.ru[3]

***Abstract*—A hybridization of genetic algorithms and machine learning techniques have proved its effectiveness for many complex benchmark and real-world problems. In this study we present a novel approach that combines self-configuring genetic algorithm for multimodal optimization and fuzzy rule based classifier. The proposed search metaheuristic controls the interactions of many techniques for multimodal optimization (different genetic algorithms) and leads to the self-configuring solving of problems with a priori unknown structure. Appling this approach to designing the fuzzy rule based classifiers, we can obtain many optimal solutions with different representation. The results of numerical experiments with popular optimization benchmark problems (for multimodal genetic algorithm) and with well-studied real-world classification problems (for self-configuring fuzzy rule based classifier design) are presented and discussed. The main feature of the proposed approach is that it does not require the participation of the human expert, because it operates in an automated, self-configuring way.**

I. Introduction

Recent advances in computer and internet technologies have led to the need to process, analyze and understand massive amounts of data. Today the area of machine learning proposes a variety of approaches for solving classification problems, and most popular are k-nearest neighbors, neural networks, support vector machines, genetic programming and others. Modern machine learning methods often use evolutionary computation techniques as a design tool, which is universal and can be applied for various structures. These evolutionary algorithms applied for machine learning problems are often called genetics-based machine learning (GBML) algorithms [1]. The GBML is focusing on using evolutionary algorithms, such as genetic algorithms (GAs), evolution strategies, genetic programming, and evolutionary programming to select the structure and/or tune the machine learning method. A number of GBML methods have been developed for solving complex classification problems [2, 3].

The fuzzy rule-based classification systems (FRBCSs) are effective approaches in machine learning, as they can provide easy-to-understand models for the end users. Although FRBCSs are useful and accurate for small datasets, they may require many computational resources for large datasets. Among the fuzzy classification methods, there are many well-developed approaches, which use GAs or other specialized evolutionary algorithms for the rule base design. Although the classical GA is a powerful method, there is a tendency of developing specialized algorithms for various problems.

Many real-world problems have more than one optimal solution, or there exists only one global optimum and several local optima in the feasible solution space. Such problems are called multimodal. As known, GAs are efficient in the multimodal environment as they use a stochastic population-based search instead of the individual search used by conventional algorithms. At the same time, traditional GAs have a tendency to converge to the best-found optimum losing population diversity. Such single best-found solution usually have very good accuracy, but may have a structure that is not convenient for human understanding and analysis. Thus there is a good idea to find many (or all) global and acceptable local optima which represent different solutions to the problem. In a case of the FRBCS, such optima, while saving comparable accuracy, may contain different rules in the rule base and/or different fuzzy term structures.

In this study a novel approach based on a metaheuristic for designing multi-strategy GA for multimodal optimization (MMO) is proposed. Its main idea is to create an ensemble of many MMO techniques and adaptively control their interactions. Such an approach would lead to the self-configuring solving of problems with a priori unknown structure. We have applied the MMO GA to the problem of designing the FRBCSs.

The rest of the paper is organized as follows. Section 2 describes significant related work. Section 3 describes the proposed approaches. In Section 4 the setup and results of numerical experiments are presented. In the Conclusion the results and further research are discussed.

II. Related Works

A. FRBCS

A number of GA-based approaches have been developed for adaptive tuning of fuzzy rule base in the FRBCSs [4, 5, 6, 7]. At the same time, many researchers try to obtain not only the computational algorithm for classification, but solution to the problem in the human-readable form. One of the ways is multi-objective problem statement [8, 9]. Here the first

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.34

objective maximize the accuracy of classification, and the second minimize the number of rules in the rule base. It is worth noting that the multi-objective statement needs more advanced GA-based techniques, and subsequent analysis of the obtained Pareto set approximation.

Another way of getting several adequate solutions to the problem is using multimodal problem statement, when the learning algorithm is focused on finding several local optima, as well as the global optimum.

B. MMO GA

The problem of MMO exists since the first GAs. Over the past decade interest for this field has increased. The recent approaches are focused on the goal of exploring the search space and finding many optima to the problem. Many efficient algorithms have been proposed. Good survey of widespread MMO techniques can be found in [10, 11]. As we can see from many studies, there is no universal approach that is efficient for all MMO problems. Many researches design hybrid algorithms, which are generally based on a combination of search algorithms and some heuristic for a niching improvement.

Another way is a combining many basic MMO algorithms to run them in parallel, migrate individuals and combine the results. In [12] an island model is applied, where islands are iteratively revised according to the genetic likeness of individuals. In [13] four MMO niching algorithms run in parallel to produce offspring, which are collected in a pool to produce a replacement step. In [14] the same scheme is realized using the clearing procedure.

The conception of designing MMO algorithms in the form of an ensemble seems to be promising. A metaheuristic that includes many different MMO approaches (different search strategies) can deal with many different MMO problems. And such a metaheuristic can be self-configuring due to the adaptive control of the interaction of single algorithms during the problem solving.

In [15] a self-configuring multi-strategy genetic algorithm in the form of a hybrid of the island model, competitive and cooperative coevolution was proposed. The approach is based on a parallel and independent run of many versions of the GA with many search strategies, which can deal with many different features of optimization problems inside the certain optimization class. The approach has demonstrated good results with respect to multi-objective and non-stationary optimization. In this study, we will apply this concept to the MMO problem.

III. METHODOLOGY

A. Hybrid Evolutionary Fuzzy Classification Algorithm

The fuzzy rule base classification (FRBCS) method, which we have implemented, is based on a simple rule base encoding into the GA chromosome. Let us describe this encoding in details.

The classification problem consists in assigning an F-dimentional object of space R^F a class number C_j from a predefined set $\boldsymbol{C} = \{C_1, \dots, C_k\}$, where k is the number of classes. Let $\mathrm{X} = \{X_1, \dots, X_F\}$ be the set of input variables, and U_f, $f = 1, \dots, F$ be the domain of f-th variable. Let $P_f = \left\{A_{f,1}, \dots, A_{f,T_f}\right\}, f = 1, \dots, F$ be the fuzzy granulation of U_f into T_f fuzzy sets. Then the fuzzy rule is:

$$R_m: IF\ X_1 IS\ A_{1,j_{m,1}}\ AND \dots AND\ X_F IS\ A_{F,j_{m,F}} THEN\ Y\ IS\ C_{j_m}$$

where Y is the output and $C_{j_m} \in \boldsymbol{C}$ is the class number for m-th rule. $j_{m,f} \in \left[1, T_f\right]$ is the number of fuzzy set from partition P_f, selected for variable X_f.

Then the rule base can be presented as:

$$J = \begin{bmatrix} j_{1,1} & \dots & j_{1,F} & C_{j_1} \\ \dots & \dots & \dots & \dots \\ j_{m,1} & \dots & j_{m,F} & C_{j_m} \\ \dots & \dots & \dots & \dots \\ j_{M,1} & \dots & j_{M,F} & C_{j_M} \end{bmatrix}, \quad (1)$$

where $j_{m,f}$ means that for rule R_m and variable X_f the fuzzy set $A_{f,j_{m,f}}$ has been selected.

The values in the matrix are integers, so that the problem of finding the optimal fuzzy rule base reduces to integer non-constrained optimization. This problem can be solved with genetic algorithm, and the key point here is the encoding method. The easiest method is to encode each $j_{m,f}$ into a binary string of length l, so that $2^l \geq T_f + 1$. One is added to the number of fuzzy sets, as we also need to encode the "Don't care" condition. Including this term allows decreasing the size of the rule base and increasing the rules' generalization ability. The class number is encoded in a similar manner.

The number of rules M in our computational experiments was fixed and equal to 12. However, if all terms in a certain rule are set to "Don't care" (DC), the rule is considered as empty and not used in classification, so the algorithm is capable of decreasing the number of rules. The number of fuzzy sets for granulation was fixed and equal to 5.

The fitness function included two main values: error on the training set with weight 1 and the complexity of the rule base with weight 0.1. The complexity of the rule base was calculated as the ratio of number of non-empty fuzzy sets to the total possible number of fuzzy sets in the rule base (which is equal to $F \times M$). Including complexity of the rule base into the fitness function allows creating of simpler rule bases.

The distance between two rule bases for the MMOGA was calculated as the number of different fuzzy sets in these rule bases.

B. SelfMMOGA

In the field of statistics and machine learning, ensemble methods are used to improve decision making. This concept can be also used in the field of GA. The main idea is to include different search strategies in the ensemble and to design effective control of algorithm interaction. The general structure of the self-configuring multi-strategy GA proposed in [15] is called Self*GA (the star sign corresponds to the certain optimization problem) and it is presented in Fig.1.

The total population size is called the computational resource. The resource is distributed between algorithms,

which run in parallel and independent over the predefined number of iterations (called the adaptation period). After the distribution, each GA included in Self*GA has its own population which does not overlap with populations of other GAs. At the first iteration, all algorithms get an equal portion of the resource. This concept corresponds to the island model, where each island realizes its own search strategy. After the adaptation period, the performance of individual algorithms is estimated with respect to the objective of the optimization problem. After that algorithms are compared and ranked. Search strategies with better performance increase their computational resource (the size of their populations). This concept corresponds to the competitive coevolution scheme. Finally, migrations of the best solutions are set to equate the start positions of algorithms for the run with the next adaptation period. This concept corresponds to cooperative coevolution.

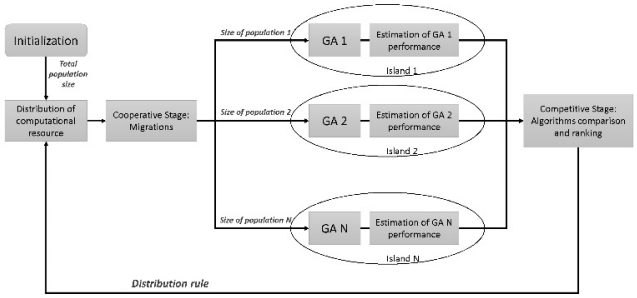

Fig. 1. The Self*GA structure

Such a technique eliminates the necessity to define an appropriate search strategy for the problem as the choice of the best algorithm is performed automatically and adaptively during the run.

Now we will discuss the design of a Self*GA for MMO problems that can be named SelfMMOGA.

At the first step, we need to define the set of individual algorithms included in the SelfMMOGA. In this study we use six basic techniques, which are well-studied and discussed [16, 17], and they can be used with binary representation with no modification. This feature is important as we encode complex structures like FRBCSs.

Single algorithms and their specific parameters are presented in Table 1. All values for radiuses and distances in Table 1 are in the Hamming metric for binary encoding and in the Euclidean metric for real-valued encoding.

The key point of any coevolutionary scheme is the performance evaluation of a single algorithm. For MMO problems performance metrics should estimate how many optima were found and how the population is distributed over the search space. Unfortunately, good performance measures exist only for benchmark MMO problems, which contain knowledge of the optima. Performance measures for black-box MMO problems are still being discussed. Some good recommendations can be found in [18]. In this study, the following criteria are used.

TABLE I. ALGORITHMS INCLUDE IN THE SELFMMOGA.

	Algorithm	Parameters
Alg1	Clearing	Clearing radius, Capacity of a niche
Alg2	Sharing	Niche radius, α
Alg3	Clustering	Number of clusters, min distance to centroid, max distance to centroid
Alg4	Restricted Tournament Selection (RTS)	Window size
Alg5	Deterministic Crowding	-
Alg6	Probabilistic Crowding	-

The first measure is called Basin Ratio (BR). The BR calculates the number of covered basins, which have been discovered by the population. It does not require knowledge of optima, but an approximation of basins is used. The BR can be calculated as

$$BR(pop) = \frac{l}{k}, \tag{2}$$

$$l = \sum_{i=1}^{k} min\left\{1, \sum_{\substack{x \in pop \\ x \neq z_i}} b(x, z_i)\right\},$$

$$b(x,z) = \begin{cases} 1, if\ x \in basin(z) \\ 0, otherwise \end{cases},$$

where *pop* is the population, k is the number of identified basins by the total population, l is the indicator of basin coverage by a single algorithm, *b(*)* is a function that indicates if an individual is in basin z.

To use the metric (1), we need to define how to identify basins in the search space and how to construct the function *b(x,z)*.

For continuous MMO problems, basins can be identified using different clustering procedures like Jarvis-Patrick, the nearest-best and others [19]. In this study, for MMO problems with binary representation we use the following approach. We use the total population (the union of populations of all individual algorithms in the SelfMMOGA). For each solution, we consider a predefined number of its nearest neighbours (with respect to the Hamming distance). If the fitness of the solution is better, it is denoted as a local optima and the centre of the basin. The number of neighbours is a tunable parameter. For a real-world problem, it can be set from some practical point of view.

The function *b(x,z)* can be easily evaluated by defining if individual x is in a predefined radius of basin centre z. The radius is a tunable parameter. In this study, we define it as

$$radius = \frac{total\ population\ size}{k}, \tag{3}$$

where k is the number of identified basins ($k=|Z|$).

The second measure is called Sum of Distances to Nearest Neighbour (SDNN). The SDNN penalizes the clustering of

solutions. This indicator does not require knowledge of optima and basins. The SDNN can be calculated as

$$SDNN(pop) = \sum_{i=1}^{pop\ size} d_{nn}(x_i, pop), \quad (4)$$
$$d_{nn}(x_i, pop) = \min_{y \in pop \setminus \{x\}} \{dist(x, y)\},$$

where d_{nn} is the distance to the nearest neighbour, *dist* is the Hamming distance.

Finally, we combine the BR and the SDNN in an integrated criterion *K*:

$$K = \alpha \cdot BR(pop) + (1 - \alpha) \cdot \overline{SDNN}(pop), \quad (5)$$

where $\overline{SDNN}$ is a normalized value of SDNN, α defines weights of the BR and the SDNN in the sum ($\alpha \in [0,1]$).

Next, we need to design a scheme for the redistribution of computational resources. New population sizes are defined for each algorithm. In this study, all algorithms give to the "winner" algorithm a certain percentage of their population size, but each algorithm has a minimum guaranteed resource that is not distributed. The guaranteed resource can be defined by the population size or by problem features.

At the cooperative stage, in many coevolutionary schemes, all individual algorithms begin each new adaptation period with the same starting points (such a migration scheme is called "the best displaces the worst"). For MMO problems, the best solutions are defined by discovered basins in the search space. As we already have evaluated the approximation of basins (*Z*), the solutions from *Z* are introduced in all populations replacing the most similar individuals.

All general GA parameters are tuned using a self-configuration procedure, introduced in [20]. For each genetic operator we define a finite number of its versions (for example, low, normal, high mutation rate). The operator is chosen according to its probability, which is defined by operator's success rates and is estimated on every generation. The success estimation for every type of operator is performed based on the averaged fitness values of offspring produced by the operator.

IV. Experimental Setup and Results

A. SelfMMOGA

To estimate the SelfMMOGA performance we have used the following list of benchmark problems. Six binary MMO problems from [13]. These test functions are based on the unitation functions, and they are massively multimodal and deceptive. And, eight real-valued MMO problems from CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization [21].

In the case of binary problems, we use the Peak Distance (PD) measure for estimating the performance. The PD indicator calculates the average distance of known optima to the nearest individuals in the population [18].

$$PD = \frac{1}{k} \sum_{i=1}^{k} d_{nn}(q_i, pop), \quad (6)$$

The following criteria for estimating the performance of the SelfMMOGA over continuous benchmark problems are used:

- Peak Ratio (PR) measures the percentage of all optima found by the algorithm (4).
- Success Rate (SR) measures the percentage of successful runs (a successful run is defined as a run where all optima were found) out of all runs.

$$PR = \frac{|\{q \in Q \mid d_{nn}(q, pop) \le \varepsilon\}|}{k} \quad (7)$$

where $Q = \{q_1, q_2, \ldots, q_k\}$ is a set of known optima, ε is accuracy level.

The maximum number of function evaluation and the accuracy level for the PR evaluation are the same as in CEC completion rules. In all experiments all computational resources are equal for all compared algorithms.

To demonstrate the control of algorithm interaction in the SelfMMOGA, we have chosen an arbitrary run of the algorithm on an arbitrary problem and have visualized the distribution of the computational resource (see Figure 2). The total population size is 200 and the minimal guaranteed amount of the computational recourse is 10. The maximum number of generations is 200 and the size of the adaptation period is 10, thus the horizontal axis contains numeration of 20 periods.

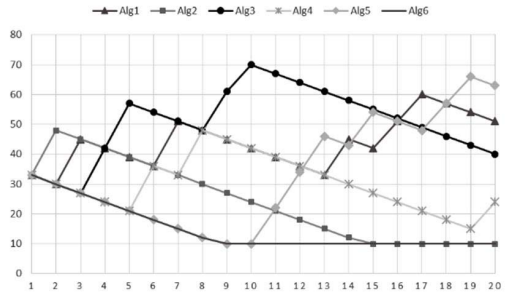

Fig. 2. Example of the SelfMMOGA run

As we can see, there is no algorithm that wins all the time. At the first two periods, Sharing (Alg2) and Clearing (Alg1) had better performance. The highest amount of the resource was won by Clustering (Alg3) at the 10th period. At the final stages, Deterministic Crowding (Alg5) showed better performance.

The results of estimating the performance of the SelfMMOGA with on binary problems are presented in Table 2. The table contains the values of the PR, the SR and the PD averaged over 50 independent runs. We also have compared the results with Ensemble of niching algorithms (ENA) proposed in [13]. There is only the SR value for the ENA.

We have also compared the results with the average of 6 stand-alone algorithms (Alg1-Alg6). The average value can be viewed as the average performance of a randomly chosen algorithm. Such an estimation is very useful for black-box optimization problems, because we have no information about problem features and, consequently, about what algorithms to use. If the performance of the SelfMMOGA is

better than the average of its component, we can conclude that on average the choice of the SelfMMOGA will be better. As we can see from Table 2, the SelfMMOGA always outperforms the average of its stand-alone component algorithms for binary problems. Moreover, for some problems no stand-alone algorithm has a SR value equal to 1, but the SelfMMOGA does.

TABLE II. RESULTS FOR BINARY PROBLEMS

Problem	Average of 6 stand-alone algorithms			SelfMMOGA			ENA
	PR	SR	PD	PR	SR	PD	SR
binaryF11	0,91	0,89	2,30	1,00	1,00	0,00	1,00
binaryF12	0,96	0,95	1,38	1,00	1,00	0,00	1,00
binaryF13	0,95	0,93	2,34	1,00	1,00	0,00	1,00
binaryF14	0,89	0,91	2,37	1,00	1,00	0,00	1,00
binaryF15	0,84	0,82	2,61	1,00	1,00	0,00	1,00
binaryF16	0,78	0,79	3,08	1,00	1,00	0,00	0,99

TABLE III. AVERAGE PR AND SR FOR EACH ALGORITHM

ε	SelfMMOGA		DE/nrand/1/bin		cDE/rand/1/bin		N-VMO		dADE/nrand/1		PNA-NSGAII	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1e-01	0,962	0,885	0,850	0,750	0,963	0,875	1,000	1,000	0,998	0,938	0,945	0,875
1e-02	0,953	0,845	0,848	0,750	0,929	0,810	1,000	1,000	0,993	0,828	0,910	0,750
1e-03	0,943	0,773	0,848	0,748	0,847	0,718	0,986	0,813	0,984	0,788	0,906	0,748
1e-04	0,907	0,737	0,846	0,750	0,729	0,623	0,946	0,750	0,972	0,740	0,896	0,745
1e-05	0,816	0,662	0,792	0,750	0,642	0,505	0,847	0,708	0,835	0,628	0,811	0,678
Average	**0,916**	**0,780**	**0,837**	**0,750**	**0,822**	**0,706**	**0,956**	**0,854**	**0,956**	**0,784**	**0,893**	**0,759**

TABLE IV. ALGORITHMS RANKING OVER CEC'13 BENCHMARK PROBLEMS

Rank by PR criterion	Algorithm	Rank by SR criterion	Algorithm
1	N-VMO and dADE/nrand/1	1	N-VMO
2	SelfMMOGA	2	dADE/nrand/1
3	PNA-NSGAII	3	SelfMMOGA
4	DE/nrand/1/bin	4	PNA-NSGAII
5	cDE/rand/1/bin	5	DE/nrand/1/bin
-	-	6	cDE/rand/1/bin

The results of estimating the performance of the SelfMMOGA with on continuous problems are presented in Tables 3 and 4. Table 3 shows results averaged over all problems. Table 4 contains ranks of algorithms by separate criteria.

We have also compared the results of the SelfMMOGA runs with some efficient techniques from the competition. The techniques are DE/nrand/1/bin and Crowding DE/rand/1/bin, N-VMO, dADE/nrand/1, and PNA-NSGAII.

As we can see from Tables 3 and 4, the SelfMMOGA shows results comparable with popular and well-studied techniques. It yields to dADE/nrand/1 and N-VMO, but we should note that these algorithms are specially designed for continuous MMO problems, and have taken 2nd and 4th places, respectively, in the CEC competition. At the same time, the SelfMMOGA has very close average values to the best two algorithms, and outperforms PNA-NSGAII, CrowdingDE and DE, which have taken 7th, 8th and 9th places respectively.

We have included only basic MMO search techniques in the SelfMMOGA. Nevertheless, it performs well due to the effect of collective decision making in the ensemble. The key feature of the approach is that it operates in an automated, self-configuring way. Thus, the SelfMMOGA can be a good tool for designing FRBCSs.

B. FRBCS

The computational experiments for the fuzzy classification were performed on 7 datasets from UCI and KEEL repositories [22, 23]. Table 5 contains the information about the datasets.

TABLE V. DATASETS DESCRIPTION

Dataset	Number of instances	Number of features	Number of classes
Australian credit	690	14	2
Banknote	1372	4	2
Column 2c	310	6	2
Column 3c	310	6	3
Ionosphere	351	34	2
Liver	345	6	2
Seeds	210	7	3

The next table contains the classification results for the standard GA and three best solutions from SelfMMOGA on the test sample.

TABLE VI. CLASSIFICATION RESULTS FOR TEST SAMPLE

Dataset	GA+ FRBCS	SelfMMOGA Solution 1	SelfMMOGA Solution 2	SelfMMOGA Solution 3
Australian	0.839	0.862	0.816	**0.867**
Banknote	**0.947**	0.892	0.862	0.867
Column 2c	0.773	**0.789**	0.751	0.768
Column 3c	0.668	**0.741**	0.619	0.674
Ionosphere	**0.747**	0.680	0.665	0.656
Liver	0.567	0.586	**0.598**	0.597
Seeds	**0.874**	0.793	0.621	0.691

As we can see, for three datasets the standard GA allows finding most accurate solutions. However, the SelfMMOGA outperforms the standard GA on 4 datasets out of 7. Moreover, the best solution is not always the first one – for example, for datasets Australian and Liver, the best solution was second or even third. Thus, using this method, several local optima have been found, and the researcher is able to select one of them.

As an example, we provide three rule bases for Liver dataset with the best accuracy, obtained on the last iteration. These rule bases contain 10, 12 and 10 rules and are very different, although they have almost the same accuracy about 0.544. We suggest that the results can help the human experts in a field of the solving problem to obtain better (or may be very new) information about the problem features.

The rule bases are presented in Figures 3-5.

Each row is one rule, where every position contains the fuzzy term for corresponding variable, and the last position is the assigned class label. The DC ("Don't Care") term means that corresponding variable is ignored in a rule.

As we can see from the results, the SelfMMOGA for FRBCS design performs well for many complex classification problems. Moreover, it provides the user not only with computational algorithm, but with explicit, human-readable representation of many solutions. Another feature of the approach is that it does not require the participation of the human-expert, because it operates in an automated, self-configuring way.

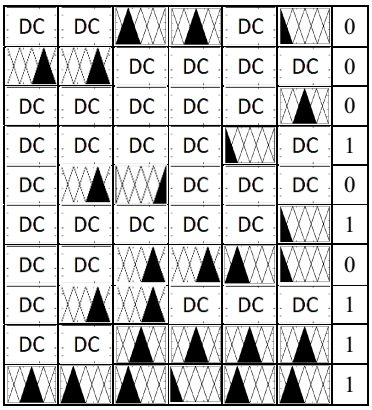

Fig. 3. Solution 1 for Liver dataset

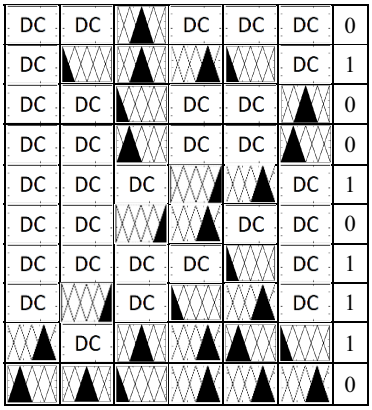

Fig. 4. Solution 2 for Liver dataset

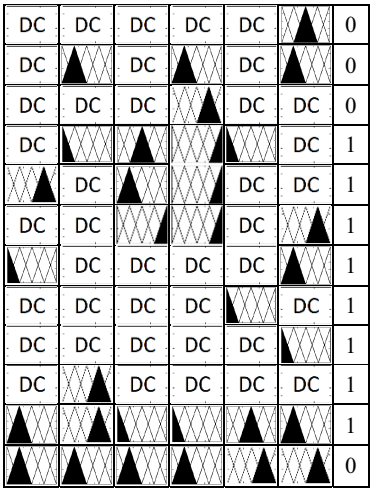

Fig. 5. Solution 3 for Liver dataset

V. CONCLUSIONS

The fuzzy rule-based classification systems are effective approach for solving many complex classification problems, and they can provide easy-to-understand models for the end users. In this work we have proposed the self-configuring approach for the multimodal designing of FRBCSs. The approach uses the multi-strategy metaheuristic that includes different multimodal search strategies in the ensemble and controls these search algorithms interaction. The main objective of the multimodal approach is to find many (possibly all) global and local optima of the problem. For the FRBCS designing problem, it means that we can obtain many effective rule bases with comparable accuracies, but different structures. Different rule bases are useful information that can help to find and understand features of the particular problem and the proposed fuzzy rule based solution.

We have investigated the proposed SelfMMOGA with a number of benchmark problems. The results show that it outperforms its component algorithms on average. This means that the SelfMMOGA is better than its component algorithm randomly chosen in a case of the black-box optimization problem, or in a case when the choice of proper algorithm requires expert knowledge and much computational effort.

The performance of the hybridization of the SelfMMOGA and the FRBCS have been tested with a number of real-world

data sets from popular repositories. The results shows that the approach demonstrates efficiency comparable with the standard approach (GA+FRBCS). In addition, it provides the end user with many optimal solutions, which can be analyzed. We can recommend this approach for the users who are not experts in a field of machine learning, because the approach operates in an automated and self-configuring way.

In further works, we will compare the proposed approach with other widespread techniques. Also we will examine the hybridization of the SelfMMOGA with other machine learning techniques.

ACKNOWLEDGMENT

Research is performed with the financial support of the Ministry of Education and Science of the Russian Federation within the State Assignment for the Siberian State Aerospace University, project 2.1889.2014/K.

REFERENCES

[1] A. Fernandez, S. Garcia, J. Luengo, E. Bernado-Mansilla, F. Herrera "Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy, and Comparative Study," Evolutionary Computation, IEEE Transactions on (Volume:14, Issue: 6), pp. 913 – 941, June 21, 2010.

[2] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," Artif. Intell., vol. 40, no. 1–3, pp. 235–282, Sep. 1989.

[3] U. Bodenhofer, F. Herrera Ten Lectures on Genetic Fuzzy Systems, Preprints of the International Summer School: Advanced Control—Fuzzy, Neural, Genetic. – Slovak Technical University, Bratislava. – 1997. p. 1–69.

[4] H. Ishibuchi, S. Mihara, Y. Nojima. Parallel Distributed Hybrid Fuzzy GBML Models With Rule Set Migration and Training Data Rotation. *IEEE Transactions on fuzzy systems*, VOL. 21, No. 2. – April 2013.

[5] H. Ishibuchi et al. Hybridization of fuzzy GBML approaches for pattern classification problems. *IEEE Trans. on Systems, Man, and Cybernetics* – Part B: Cybernetics, Volume 35,Issue 2, pp. 359-365, April 2005.

[6] E. Zhou, A. Khotanzad, Fuzzy classifier design using genetic algorithms, Pattern Recognition, Volume 40, Issue 12, 2007, pp. 3401–3414.

[7] R. Sergienko, E. Semenkin and V. Bukhtoyarov, Michigan and Pittsburgh methods combination for fuzzy classifier design with coevolutionary algorithm, in 2013 IEEE Congress on Evolutionary Computation (CEC'2013), pp. 3252-3259, 2013.

[8] R. Alcala, P. Ducange, F. Herrera, B. Lazzerini, and F. Marcelloni, A multiobjective evolutionary approach to concurrently learn rule and data bases of linguistic fuzzy-rule-based systems, IEEE Trans. Fuzzy Syst., vol. 17, no. 5, pp. 1106–1122, 2009.

[9] M. Fazzolari, R. Alcala, Y. Nojima, H. Ishibuchi, F. Herrera, A Review of the Application of Multi-Objective Evolutionary Fuzzy Systems: Current Status and Further Directions. IEEE Transactions on Fuzzy Systems, 21:1 (2013) 45-65.

[10] S. Das, S. Maity, B.-Y. Qub, P.N. Suganthan, Real-parameter evolutionary multimodal optimization: a survey of the state-of-the art. Swarm and Evolutionary Computation 1, pp. 71–88, 2011.

[11] Y. Liu, X. Ling, Zh. Shi, M. Lv, J. Fang, L. Zhang, A Survey on Particle Swarm Optimization Algorithms for Multimodal Function Optimization. Journal of Software, Vol. 6, No. 12. pp. 2449-2455, 2011.

[12] M. Bessaou, A. Petrowski, P. Siarry, Island Model Cooperating with Speciation for Multimodal Optimization. Parallel Problem Solving from Nature PPSN VI, Lecture Notes in Computer Science, Volume 1917. pp. 437-446, 2000.

[13] E.L. Yu, P.N. Suganthan, Ensemble of niching algorithms. Information Sciences, Vol. 180, No. 15. pp. 2815-2833, 2010.

[14] B. Qu, J. Liang, P.N. Suganthan, T. Chen, Ensemble of Clearing Differential Evolution for Multi-modal Optimization. Advances in Swarm Intelligence Lecture Notes in Computer Science, Volume 7331. pp. 350-357, 2012.

[15] E. Sopov, A Self-configuring Metaheuristic for Control of Multi-Strategy Evolutionary Search. ICSI-CCI 2015, Part III, LNCS 9142. pp. 29-37, 2015.

[16] G. Singh, K. Deb, Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: Proc. of the Genetic and Evolutionary Computation Conference, Seattle. pp. 1305–1312, 2006.

[17] S. Das, S. Maity, B.-Y. Qub, P.N. Suganthan, Real-parameter evolutionary multimodal optimization: a survey of the state-of-the art. Swarm and Evolutionary Computation 1, pp. 71–88, 2011.

[18] M. Preuss, S. Wessing, Measuring multimodal optimization solution sets with a view to multiobjective techniques. EVOLVE – A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV. AISC, vol. 227, Springer, Heidelberg. pp. 123–137, 2013.

[19] M. Preuss, C. Stoean, R. Stoean, Niching foundations: basin identification on fixed-property generated landscapes. In: Proc. of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011. pp. 837–844, 2011.

[20] E.S. Semenkin, M.E. Semenkina, Self-configuring Genetic Algorithm with Modified Uniform Crossover Operator. Advances in Swarm Intelligence. Lecture Notes in Computer Science 7331. Springer-Verlag, Berlin Heidelberg. pp. 414-421, 2012.

[21] Li, X., Engelbrecht, A., Epitropakis, M.G.: Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization. Evol. Comput. Mach. Learn. Group, RMIT University, Melbourne, Australia. Tech. Rep., 2013.

[22] UC Irvine Machine Learning Repository, http://archive.ics.uci.edu/ml/

[23] KEEL (Knowledge Extraction based on Evolutionary Learning), http://www.keel.es