Hierarchical Fuzzy System Modeling by Genetic and Bacterial Programming Approaches

Krisztián Balázs, János Botzheim and László T. Kóczy

***Abstract*— In this paper a method is proposed for constructing hierarchical fuzzy rule bases in order to model black box systems defined by input-output pairs, i.e. to solve supervised machine learning problems. The resultant hierarchical rule base is the knowledge base, which is constructed by using structure constructing evolutionary techniques, namely, Genetic and Bacterial Programming Algorithms. Applying hierarchical fuzzy rule bases is a way of reducing the complexity of the knowledge base, whereas evolutionary methods ensure a relatively efficient learning process. This is the reason of the investigation of this combination.**

I. Introduction

This paper proposes a method for constructing hierarchical fuzzy rule bases in order to model black box systems defined by input-output pairs, i.e. to solve a certain type of supervised machine learning problems. The resultant hierarchical rule base is the knowledge base, which is constructed by using structure constructing evolutionary techniques, namely, Genetic and Bacterial Programming Algorithms.

There are two key elements in machine learning systems: the learning architecture including the structure of the knowledge base and the learning algorithm applied during the learning process, i.e. the technique that optimizes the knowledge base according to the particular problem. Hence, in order to establish a learning system having favorable properties (low knowledge complexity, high accuracy, fast learning process, etc.) these elements must be chosen so that they outperform other alternatives of knowledge bases and learning algorithms.

For interpretability purposes, i.e. after the learning process to have a quasi-interpretable knowledge base, this paper considers only knowledge bases in the form of fuzzy rule bases.

The classical approaches of fuzzy control deal with dense rule bases where the universe of discourse is fully covered by the antecedent fuzzy sets of the rule base in each dimension, thus for every input there is at least one activated rule. The main problem is the high computational complexity of these traditional approaches.

Krisztián Balázs is with the Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary (e-mail: balazs@tmit.bme.hu).

János Botzheim is with the Department of Automation, Széchenyi István University, Győr, Hungary (e-mail: botzheim@sze.hu).

László T. Kóczy is with the Institute of Informatics, Electrical and Mechanical Engineering, Faculty of Engineering Sciences, Széchenyi István University, Győr, Hungary (e-mail: koczy@sze.hu) and with the Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary (e-mail: koczy@tmit.bme.hu).

If a fuzzy model contains k variables and maximum T linguistic terms in each dimension, the order of the number of necessary rules is $\mathcal{O}(T^k)$. This expression can be decreased either by decreasing T, or k, or both. Of course, these are very different in terms of the efficiency of the complexity reduction. The first way (decreasing T) leads to sparse rule bases and rule interpolation, a method that was first introduced by Kóczy and Hirota (see e.g. [1], [2]). The second, more effective way (decreasing k) aims to reduce the dimension of the sub-rule bases by using meta-levels or hierarchical fuzzy rule bases (e.g. [3], [4]). The combination of these two methods leads to the decreasing of both T and k, and was introduced in [5].

The scope of this paper focuses only on the second mentioned way of complexity reduction, namely, the application of hierarchical fuzzy rule bases. Future work may aim to consider the combination of interpolative and hierarchical systems.

Nature inspired evolutionary optimization algorithms suitable for global optimization of even non-linear, high-dimensional, multimodal, and non-continuous problems. The original Genetic Algorithm (GA) was developed by Holland [6] and was based on the process of evolution of biological organisms. Recognized as a powerful global search technique, genetic algorithms have been applied to a wide variety of problems. They exhibit a remarkable balance between search domain exploration and exploitation [7]. On one hand, every part of the domain is searched and on the other hand, the search effort is concentrated around the best solutions and their neighborhood, producing even better solutions. Bacterial Evolutionary Algorithm (BEA) [8] inspired by the microbial evolution phenomenon is similar, but based on results in the literature appearing more effective technique than GA (e.g. [9], [10], [11]). Approaches like Genetic Programming (GProg) [12] and Bacterial Programming (BProg) [13], present an alternative to the former algorithms. These techniques use the same operators as GA and BEA, respectively, though they require expression trees for gene representation, which could model hierarchical fuzzy rule bases, for example.

Evolutionary algorithms are useful tools for identification problems, they can help with finding a good compromise between the accuracy and the complexity of the model. In case of non-hierarchical rule bases, genetic and bacterial approaches were successfully applied for fuzzy rule based learning by various authors [8], [14]. This is the reason why Genetic and Bacterial Programming is applied in our work as the learning algorithms that optimize the knowledge base.

978-1-4244-8126-2/10/$26.00

Since so far no method has been invented to obtain the main properties (efficiency, accuracy, etc.) of such complex systems exactly, they can be figured out mostly by simulation. Therefore, in order to discover the capabilities of the established hierarchical fuzzy rule based learning systems, simulation runs were carried out.

The next section gives a brief overview of the algorithms and techniques used. The third section proposes a method for constructing and optimizing hierarchical fuzzy rule bases by Genetic and Bacterial Programming approaches. The simulation results and the observed behavior will be discussed in the fourth section. Finally, in the last section we will summarize our work and draw some conclusions.

II. Overview of the techniques and algorithms

In order to carry out this investigation, it is necessary to have familiarity with some theoretical fields including the concept of hierarchical fuzzy rule bases, structure constructing evolutionary optimization techniques (GProg, BProg) and machine learning.

The following subsections aim to give a brief overview of some important points of these theoretical aspects, which will be repeatedly referred to later in the paper.

A. Hierarchical fuzzy rule based systems

The basic idea of using hierarchical fuzzy rule bases is the following [3], [4]. Often the multi-dimensional input space $X = X_1 \times X_2 \times \cdots \times X_m$ can be decomposed, so that some of its components, e.g. $Z_0 = X_1 \times X_2 \times \cdots \times X_p$ determine a subspace of $X (p < m)$, so that in Z_0 a partition $\Pi = D_1, D_2, \ldots, D_n$ can be determined: $\bigcup_{i=1}^{n} D_i = Z_0$.

In each element of Π, i.e. in each D_i, a sub-rule base R_i can be constructed with local validity. In the worst case, each sub-rule base refers to exactly $X/Z_0 = X_{p+1} \times \cdots \times X_m$. The complexity of the whole rule base $\mathcal{O}(T^m)$ is not decreased, as the size of R_0 is $\mathcal{O}(T^p)$, and each R_i, $i > 0$, is of order $\mathcal{O}(T^{m-p})$, $\mathcal{O}(T^p) \times \mathcal{O}(T^{m-p}) = \mathcal{O}(T^m)$.

A way to decrease the complexity would be finding in each D_i a proper subset of $X_{p+1} \times \cdots \times X_m$, so that each R_i contains only less than $m - p$ input variables. In some concrete applications in each D_i a proper subset of $X_{p+1}, \ldots, X_m$ can be found so that each R_i contains only less than $m - p$ input variables, and the rule base has the structure shown in Table I., where $z_i \in Z_i$, $Z_0 \times Z_i$ being a proper subspace of X for $i = 1, .., n$.

If the number of variables in Z_i is $k_i < m - p$ and $\max_{i=1}^{n} k_i = K < m - p$, then the resulting complexity will be $\mathcal{O}(T^{p+K}) < \mathcal{O}(T^m)$, so the structured rule base might lead to a reduction of the complexity.

The task of finding such a partition is often difficult, if not impossible, (sometimes such a partition does not even exist), however there are cases when, locally, some variables unambiguously dominate the behavior of the system, and consequently the omission of the other variables allows an acceptably accurate approximation.

B. Structure constructing evolutionary techniques

1) Genetic Programming: There are various optimization algorithms, which were inspired by processes in the nature. The advantage of these algorithms is their ability to solve and quasi-optimize problems with non-linear, high-dimensional, multimodal, and discontinuous character. These processes can easily be applied in optimization problems where one individual corresponds to one possible solution of the problem. In the original Genetic Algorithm (GA) [6] an individual is represented by a sequence of numbers, for example a sequence of bits. This sequence is called chromosome. GA uses three operators: selection, crossover, and mutation. A more recent approach is Genetic Programming (GProg) [12], which uses the same operators as GA, though the individuals are represented by the so called expression tree. This tree is composed of the non-terminal nodes (also called as inner or function nodes) and the terminal nodes (leaves).

The evolutionary process involves the following steps:

a) creation of an initial population
b) evaluation of the candidates
c) application of genetic operators:
   - **selection**: pairs of trees are selected based on their fitness for reproduction
   - **crossover**: nodes in the trees are selected at random and the sub-trees belonging to the selected nodes are exchanged producing a pair of offspring trees
   - **mutation**: this is performed by either replacing a node selected at random by a sub-tree generated randomly or by changing the inner properties of the node
   - **replacement**: substituting the created offsprings to the population
d) test of the termination criteria (e.g. reaching the maximum number of generations, or the time limit) and

TABLE I
HIERARCHICAL RULE BASE

R_0: **If** z_0 is D_1 **then** use R_1
If z_0 is D_2 **then** use R_2
...
If z_0 is D_n **then** use R_n

R_1: **If** z_1 is A_{11} **then** y is B_{11}
If z_1 is A_{12} **then** y is B_{12}
...
If z_1 is A_{1r_1} **then** y is B_{1r_1}

R_2: **If** z_2 is A_{21} **then** y is B_{21}
If z_2 is A_{22} **then** y is B_{22}
...
If z_2 is A_{2r_2} **then** y is B_{2r_2}

...

R_n: **If** z_n is A_{n1} **then** y is B_{n1}
If z_n is A_{n2} **then** y is B_{n2}
...
If z_n is A_{nr_n} **then** y is B_{nr_n}

if any of the termination criteria is achieved, the algorithm stops, otherwise it continues from step b)

The cycle of evolution is summarized in Fig. 1.

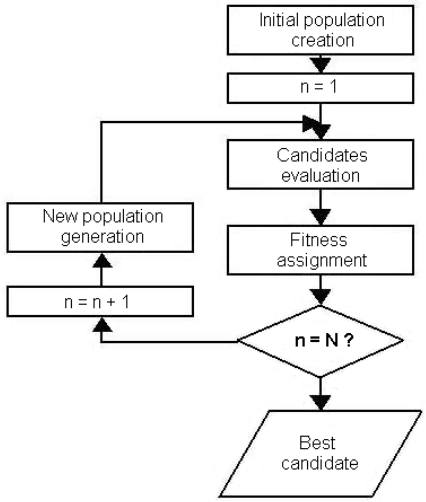

Fig. 1. Flowchart of Genetic Programming

2) Bacterial Programming: Compared to GA, a slightly different evolutionary technique is called Bacterial Evolutionary Algorithm (BEA). This algorithm was introduced by Nawa and Furuhashi in [8]. The first version of this algorithm was called Pseudo-Bacterial Genetic Algorithm (PBGA) [15] which proposed a modified mutation operator called bacterial mutation, based on the natural phenomenon of microbial evolution. Bacterial Evolutionary Algorithm introduced a new operator called gene transfer operator. While PBGA incorporates bacterial mutation and crossover operator, the BEA substitutes the classical crossover with the gene transfer operation. Both of these new operators were inspired by bacterial evolution. Bacteria can transfer genes to other bacteria. The bacterial mutation performs local optimization whilst the gene transfer allows the bacteria to directly transfer information to the other individuals in the population.

Based on these bacterial operations, but using the tree structures similar to the ones in the Genetic Programming, a new technique was proposed, named Bacterial Programming (BProg) [13].

The evolutionary process of BProg involves the following steps:

a) creation of an initial population
b) application of bacterial operators:
   - **bacterial mutation**: this is performed by either replacing a node selected at random by a sub-tree generated randomly or by changing the inner properties of the node
   - **gene transfer**: nodes in the trees are selected at random in the selected superior individual and the sub-trees belonging to the selected nodes are copied to the selected inferior individual

c) test of the termination criteria (e.g. reaching the maximum number of generations, or the time limit) and if any of the termination criteria is achieved, the algorithm stops, otherwise it continues from step b)

The cycle of evolution is summarized in Fig. 2.

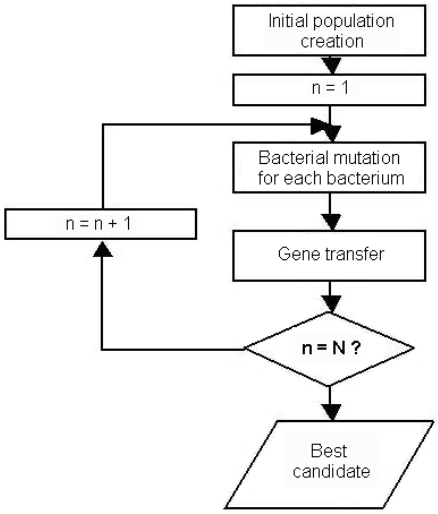

Fig. 2. Flowchart of Bacterial Programming

C. Supervised machine learning

Supervised machine learning [16] means a process where parameters of a 'model' are being adjusted so that its behavior becomes similar to the behavior of the 'system', which is to be modeled. Since the behavior can be characterized by input-output pairs, the aim of the learning process can be formulated so that the model should give similar outputs for the input as the original system does.

The model can be, for example, a simple function (e.g. a polynomial function), where the parameters are the coefficients, or it can be a neural network, where the parameters are the weights, or it can be a hierarchical fuzzy rule base together with an inference engine [17]. In this case the parameters determine the structure of the rule base as well as the membership functions of the rules in the rule base. In our work we applied a hierarchical fuzzy rule base containing trapezoidal membership functions.

If a function $\phi(\boldsymbol{x})$ denotes the system and $f(\boldsymbol{x}, \boldsymbol{p})$ denotes the model, where $\boldsymbol{x} \in X$ is the input vector and $\boldsymbol{p}$ is the adjustable parameter vector, the previous requirement can be expressed as follows:

$$\forall \boldsymbol{x} \in \boldsymbol{X}: \ \phi(\boldsymbol{x}) \overset{!}{\approx} f(\boldsymbol{x}, \boldsymbol{p})$$

In a supervised case the learning happens using a set of training samples (input-output pairs). If the number of samples is m, the input in the ith sample is $\boldsymbol{x}_i$, the desired

output is $d_i = \phi(\boldsymbol{x}_i)$ and the output of the model is $y_i = f(\boldsymbol{x}_i, \boldsymbol{p})$, the following formula can be used:

$$\forall i \in [1, m]\colon\ d_i \overset{!}{\approx} y_i$$

The error (ε) shows how similar the model to the system is (see Fig. 3). It is the function of the parameter vector, so it can be denoted by $\varepsilon(\boldsymbol{p})$. A widely applied definition for the error is the Mean Squared Error (MSE):

$$\varepsilon(\boldsymbol{p}) = \frac{\sum_{i=1}^{m} (d_i - y_i)^2}{m}$$

Obviously, the task is to minimize this $\varepsilon(\boldsymbol{p})$ function. It can be done by numerical optimization algorithms.

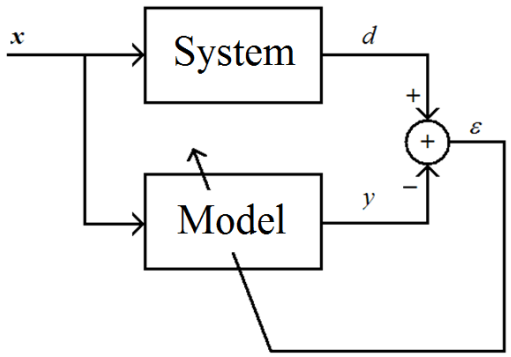

Fig. 3. Supervised machine learning

This way machine learning problems can be traced back to optimization problems.

III. The Encoding Method

This section describes the method how the individuals represent the applicant hierarchical fuzzy rule bases, i.e. how the rule bases are encoded.

The non-terminal (inner) nodes hold the antecedent parts of one or more rules. In case of each non-terminal node the number of considered input dimensions (number of *decision variables*) is a random value, whose maximum can be parameterized. A decision variable can only be such a dimension, which has not been decision variable in the ancestors of the current node. (This way the size of the expression tree becomes limited.) The consequent part of a rule is a child node of the particular node, which can be either a non-terminal or a terminal node. Thus, the number of rules a non-terminal node holds is determined by the number of the children of the particular node. The number of children of the nodes is a random value, whose maximum, i.e. the maximum number of rules belonging to a node, can be parameterized.

Each terminal node (leaf) holds one single fuzzy set, as a conclusion (output) set.

There can be non-terminal nodes as well as terminal nodes on the same level, hence meta-rules and rules can appear together (see Fig. 4).

Since a decision variable cannot be such a dimension, which has already been a decision variable in any ancestor of the current node, in case of crossover in GProg the transfered nodes change their decision variables so that the new decision variables be such dimensions, which have not been decision variables in the ancestors. The same solution of this technical difficulty is applied in gene transfer in BProg, too.

In case of mutation in GProg and bacterial mutation in BProg there are no technical difficulties: only new sub-trees or nodes must be generated with the consideration of the above described encoding rules.

The applied inference algorithm is a simple extension of the well-known Mamdani-inference technique [18]. This extension is necessary, because there are child nodes in the consequent parts of the rules. Since the conclusions of sub-trees are fuzzy sets (specially, the contents of terminal nodes can also be considered as conclusions), they can be substituted into the consequent parts of the rules of the parent nodes. This can be performed recursively.

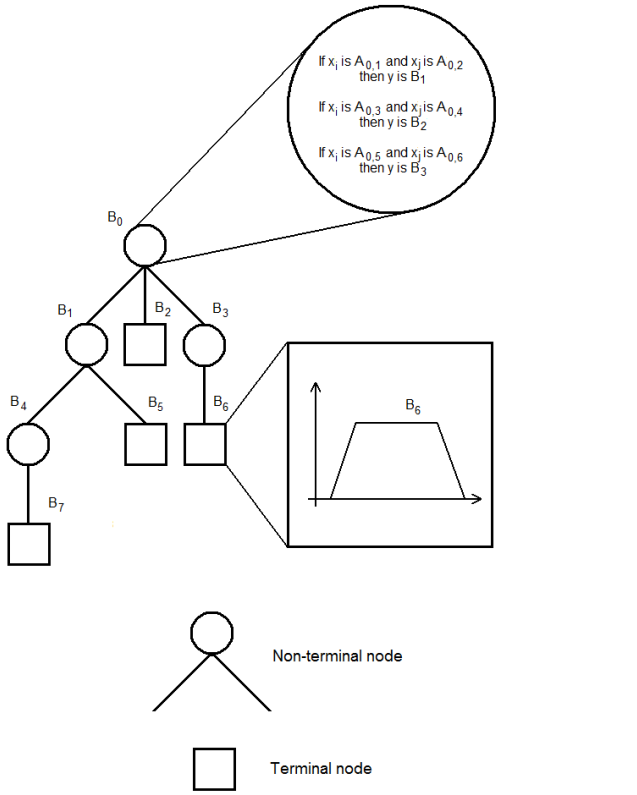

Fig. 4. Tree of a hierarchical rule base (rules and meta-rules appear together on the same levels)

IV. Simulation Results

Simulation runs were carried out in order to discover the important properties of the established learning systems (speed of learning, accuracy of the resultant knowledge base, etc.).

In our work a six dimensional learning problem was applied that was also used in [8] to evaluate the performance

of Bacterial Evolutionary Algorithm. This problem is to approximate the following function:

$$f_{6dim} = x_1 + \sqrt{x_2} + x_3 x_4 + 2e^{2(x_5 - x_6)}$$
$$x_1, x_2 \in [1, 5], \quad x_3 \in [0, 4], \quad x_4 \in [0, 0.6],$$
$$x_5 \in [0, 1], \quad x_6 \in [0, 1.2].$$

In the simulations the parameters had the following values, because after a number of test runs these values seemed to be suitable. The number of individuals in a generation was 8 in both Genetic and Bacterial Programming algorithms. The maximum number of rules belonging to a node, i.e. the maximum number of children of a node, was set to 4 and the maximum number of decision variables in a node was 3. In case of genetic techniques the selection rate was 0.3 and the mutation rate was 0.1, in case of bacterial techniques the number of clones was 5 and 4 gene transfers were carried out in each generation. The genetic methods applied elitist strategy.

The numbers of training samples were 200 in the learning processes.

Two type of simulation were carried out, which differed in the termination criteria. The runs applying the first criterion stopped after reaching the maximum number of generations, which was 1000 generations, whereas the other optimization processes stopped after reaching the time limit, which was 1000 seconds.

At the end of each optimization processes the accuracy of the resultant rule bases was measured by using three error definitions:

- *Mean Squared Error* (*MSE*):

$$\frac{1}{m} \sum_{i=1}^{m} (d_i - y_i)^2$$

- *Mean Squared Relative Error* (*MSRE*):

$$\frac{1}{m} \sum_{i=1}^{m} \frac{(d_i - y_i)^2}{y_i^2}$$

- *Mean Relative Error Percentage* (*MREP*):

$$\frac{100}{m} \sum_{i=1}^{m} \left| \frac{d_i - y_i}{y_i} \right|$$

The MSE, MSRE, and MREP values were calculated on the test samples.

The computation time and the number of executed generations were also observed.

In case of both algorithms for each parameterization 10 runs were carried out. Then we took the mean of the obtained values.

During the runs the fitness values of the best individuals were monitored. These fitness values were calculated based on the MSE values (measured on the training samples) as follows:

$$F = \frac{10}{\text{MSE} + 1} = \frac{10m}{\sum_{i=1}^{m} (d_i - y_i)^2 + m}.$$

The means of the fitness values of the best individuals during the runs were presented in Fig. 5 and Fig 6. to get a better overview. The horizontal axes show the executed number of generations and the elapsed computation time in seconds, respectively, and the vertical axes show the fitness values of the best individuals at the current time.

In the figures the dashed line shows the results of GProg and the solid line presents the graph of BProg.

The results of the simulation runs are presented in Table II and Table III. In Table II 'Comp. time' denotes the mean of the computation time the optimization processes required and in Table III 'No. of gen.' denotes the mean of the number of executed generations.

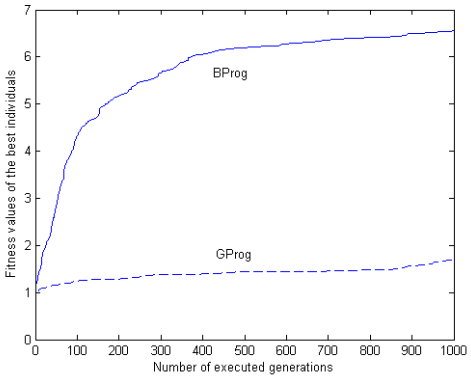

Fig. 5. Results for the six-dimensional learning problem in case of generation limit

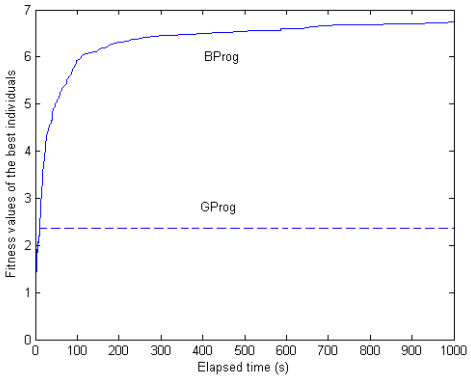

Fig. 6. Results for the six-dimensional learning problem in case of time limit

As it can be observed, BProg outperformed GProg in both simulation types (see Table II and Table III). In case of the time limited runs GProg stuck in local optima and its convergence speed became zero in 20 seconds, whereas BProg had a positive convergence speed even after 1000 seconds and continued advancing to better fitness values (see Fig. 6).

Compared to previous results produced by non-hierarchical, Mamdani-inference based learning systems

[11], the results given by the established hierarchical learning systems are promising, however further simulations and comparisons are needed.

TABLE II
RESULTS FOR THE SIX DIMENSIONAL LEARNING PROBLEM IN CASE OF 1000 GENERATIONS

	GProg	BProg
MSE	5.4206	1.2774
MSRE	0.0835	0.0169
MREP	21.6679	9.8298
Comp. time	0.4	75.8

TABLE III
RESULTS FOR THE SIX DIMENSIONAL LEARNING PROBLEM IN CASE OF THE 1000 SECONDS TIME LIMIT

	GProg	BProg
MSE	3.6465	1.0334
MSRE	0.0576	0.0157
MREP	18.0081	9.5196
No. of gen.	1172084	20183

V. CONCLUSIONS

Our work proposed a method for constructing hierarchical fuzzy rule bases in order to solve supervised machine learning problems. The rule base was constructed by using Genetic and Bacterial Programming algorithms.

Simulation runs were carried out to discover the properties of the obtained learning systems. Bacterial Programming outperformed Genetic Programming in both executed simulation types.

Further research may aim to improve both the encoded learning architecture and the applied optimization methods (e.g. using gradient steps during the optimization process) and to compare the established hierarchical fuzzy systems with non-hierarchical ones.

ACKNOWLEDGMENT

This paper was supported by the National Scientific Research Fund Grant OTKA K75711, a Széchenyi István University Main Research Direction Grant and the Social Renewal Operation Programme TÁMOP-4.2.2 08/1-2008-0021.

REFERENCES

[1] L. T. Kóczy and K. Hirota: Approximate reasoning by linear rule interpolation and general approximation, International Journal of Approximate Reasoning, 9, 1993, pp. 197–225.
[2] L. T. Kóczy and K. Hirota: Interpolative reasoning with insufficient evidence in sparse fuzzy rule bases, Information Sciences, 71, 1993, pp. 169–201.
[3] M. Sugeno, T. Murofushi, J. Nishio, and H. Miwa: Helicopter control based on fuzzy logic, Presented at the Second Fuzzy Symposium on Fuzzy Systems and Their Applications to Human and Natural Systems, Tokyo – Ochanomizu, Japan, 1991.
[4] M. Sugeno, M. F. Griffin, and A. Bastian: Fuzzy hierarchical control of an unmanned helicopter, Proceedings of the 5th IFSA World Congress (IFSA93), Seoul, South Korea, 1993, pp. 1262-1265.
[5] L. T. Kóczy and K. Hirota: Interpolation in structured fuzzy rule bases, Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'93, San Francisco, 1993, pp. 803–808.
[6] J. H. Holland: Adaption in Natural and Artificial Systems, The MIT Press, Cambridge, Massachusetts, 1992.
[7] D. E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Massachusetts, 1989.
[8] N. E. Nawa and T. Furuhashi: Fuzzy system parameters discovery by bacterial evolutionary algorithm, IEEE Transactions on Fuzzy Systems, vol. 7, no. 5, 1999, pp. 608–616.
[9] K. Balázs, J. Botzheim and L. T. Kóczy: Comparative Analysis of Various Evolutionary and Memetic Algorithms, Proceedings of the 10th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, CINTI 2009, Budapest, Hungary, 2009, pp. 193–205.
[10] K. Balázs, J. Botzheim, L. T. Kóczy: Comparison of Various Evolutionary and Memetic Algorithms, Proceedings of the International Symposium on Integrated Uncertainty Management and Applications, IUM 2010, Ishikawa, Japan, 2010, pp. 431–442.
[11] K. Balázs, J. Botzheim, L. T. Kóczy: Comparative Analysis of Interpolative and Non-interpolative Fuzzy Rule Based Machine Learning Systems Applying Various Numerical Optimization Methods, World Congress on Computational Intelligence, WCCI 2010, Barcelona, Spain, 2010, published in this very same proceedings.
[12] J. R. Koza: Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press, 1992.
[13] J. Botzheim, C. Cabrita, L. T. Kóczy, and A. E. Ruano: Genetic and bacterial programming for B-spline neural networks design, Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 11, no. 2, 2007, pp. 220–231.
[14] J. Botzheim, B. Hámori, L. T. Kóczy, and A. E. Ruano: Bacterial algorithm applied for fuzzy rule extraction, Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, IPMU 2002, pp. 1021–1026, Annecy, France, 2002.
[15] N. E. Nawa, T. Hashiyama, T. Furuhashi, and Y. Uchikawa: Fuzzy logic controllers generated by pseudo-bacterial genetic algorithm, Proceedings of the IEEE International Conference on Neural Networks, ICNN97, Houston, 1997, pp. 2408–2413.
[16] E. Alpaydin: Introduction to Machine Learning, The MIT Press, 2004, 445 p.
[17] D. Driankov, H. Hellendoorn, M. Reinfrank: An Introduction to Fuzzy Control, Springer, New York, 316 p.
[18] E. H. Mamdani: Application of fuzzy algorithms for control of simple dynamic plant, IEEE Proc., vol. 121, no. 12, 1974, pp. 1585–1588.