University of Ostrava

Institute for Research and Applications of Fuzzy Modeling

Optimization of fuzzy models using evolutionary algorithms

Ivan Křivý

Research report No. 115

2007

Submitted/to appear:

Proceedings of 7th International Conference APLIMAT 2008

Supported by:

Grant 201/06/0612 of the Czech Grant Agency; Research Scheme MSM 6198898701 of the Czech Ministry of Education, Youth and Sport

University of Ostrava
Institute for Research and Applications of Fuzzy Modeling
30. dubna 22, 701 03 Ostrava 1, Czech Republic

tel.: +420-59-7091401 fax: +420-59-6120478
e-mail: ivan.krivy@osu.cz

Optimization of fuzzy models using evolutionary algorithms

Ivan Křivý

University of Ostrava, Faculty of Science, Computer Department,
30. dubna 22, 701 03 Ostravah
e-mail: ivan.krivy@osu.cz

Keywords: *Fuzzy modeling; Takagi-Sugeno model; Fuzzy rule-base simplification; Optimization; Evolutionary algorithms; Genetic algorithms; Convergence*

1. Introduction

An important characteristic of fuzzy models is that are based on partitioning information into fuzzy regions by means of fuzzy sets. In contrast to classical crisp sets dividing the universe of discourse into members and non-members, fuzzy sets make possible to describe various forms of gradual transition from total membership to total non-membership.
A typical fuzzy model constitutes a base of fuzzy rules (**if-then** rules) that establish relations between relevant system variables (inputs and outputs). Each rule is considered as a fuzzy relation defining a locally valid model, the total relation being composed by combining the fuzzy relations defined by the individual rules.

There are two principal approaches how to implement fuzzy models. While Mamdani fuzzy models [12] use rules, in which both premise and consequent are described by fuzzy sets, Takagi-Sugeno models [25] have a fuzzy premise but their consequents are defined as (mostly linear) functions of the premise variables. In this contribution we will consider the Takagi-Sugeno fuzzy model.

The next Section introduces the Takagi-Sugeno model and shows how to construct an initial fuzzy model. In Section 3 we will discuss the problems related to rule base reduction and simplification. Section 4 deals with the problems related to fuzzy model representation. Finally, Section 5 is devoted to the optimization of fuzzy models by using various evolutionary algorithms, especially genetic ones.

2. Takagi-Sugeno fuzzy model

Takagi-Sugeno model (for short TS model) consists of a set of **if-then** rules, where the rule premises are expressed by fuzzy sets and the rule consequents are considered to be mostly linear functions of input variables. Therefore, we can formulate the model as follows (see [19]):

$$R_i: \ \textbf{If}\ x_i\ is\ A_{i1}\ \textbf{and}\ \ldots\ x_n\ is\ A_{in}\ \textbf{then}$$

$$g_i = p_{i1}x_1 + \ldots p_{in}x_n + p_{i(n+1)},\ i = 1, \ldots, M, \tag{1}$$

where $\mathbf{x} = [x_1, \ldots, x_n]^T$ is the vector of input variables, g_i the output variable, R_i i-th fuzzy rule, $A_{i1}, \ldots, A_{in}$ fuzzy sets defined in the premise space by membership functions $\mu_{Aij}(x_j)$ and $p_{i1} \ldots, p_{i(n+1)}$ consequent parameters.

The output y of the model is given as a weighted mean of the individual fuzzy rule contributions:

$$y = \frac{\sum_{i=1}^{M} \beta_i g_i}{\sum_{i=1}^{M} \beta_i},$$

where β_i denotes the degree of fulfillment of the i-th rule:

$$\beta_i = \prod_{j=1}^{n} A_{ij}(x_j),\ i = 1, \ldots, M,$$

where $A_{ij}(x_j)$ is the membership of input variable x_j in the fuzzy set A_{ij}.

When deriving an initial fuzzy rule-based model, it is very important to partition the model input space (the premise of the rule base). Although fixed membership functions are sometimes used for partitioning the space [7], membership functions derived directly from the data are considered to be much better. The best way how to realize partitioning from data is to use fuzzy clustering algorithms. The algorithms seek for groups within the data that are homogeneous with regard to the structure in both input and output [21].

Let us suppose we have a matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_K]$ and an output vector $\mathbf{y} = [y_1, \ldots, y_K]$ derived from the available data of size K. Then we can derive an initial fuzzy rule-based model in two steps.

- Fuzzy sets A_{ij} in the premise space are determined by using fuzzy clustering. There are a few available clustering algorithms, one of the most popular being fuzzy c-means algorithm [4].
- As soon as the premise is fixed, the rule consequents are obtained by least squares method [2]. i.e. by minimizing the well-known formula

$$f^* = \sum_{k=1}^{K} (y_k - y_k^*)^2, \tag{2}$$

  where $\mathbf{y}$ is the true output vector and $\mathbf{y}^*$ the model output vector.

3. Fuzzy rule-base simplification

The popular rule base simplification method [20] uses a similarity measure to reduce the redundancy among the fuzzy sets in the model. The similarity measure is based on the set operations of union and intersection. It is defined as

$$S(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

where $|.|$ denotes the set cardinality, $\cap$ and $\cup$ the operators of intersection and union, respectively. For discrete input variables $\mathbf{x} = \{x_l; l = 1, \ldots, m\}$ we have

$$S(A, B) = \frac{\sum_{l=1}^{m} (\mu_A(x_l) \wedge \mu_B(x_l))}{\sum_{l=1}^{m} (\mu_A(x_l) \vee \mu_B(x_l))},$$

where $\wedge$ and $\vee$ are the minimum and maximum operators, respectively.

It is clear that the similarity measure S is symmetric in $[0, 1]$. If $S(A, B) = 1$, the membership functions μ_A and μ_B are identical. In case $S(A, B) = 0$ the membership functions are non-overlapping. Fuzzy sets are considered to be similar (merged), when their similarity measure exceeds a user-defined threshold $\gamma \in [0, 1]$ (usually $\gamma = 0.5$).

Similarity of fuzzy sets (merging of similar membership functions) makes possible reducing the number of different fuzzy sets in the model premise. When all the fuzzy sets for a given input variable are similar to the universal set or if their merging results in only one membership function, then this input variable can be removed from the model.

Let us suppose that we have an initial fuzzy model established from data. Roubos and Setnes [19] recommend to simplify and optimize it successively using a combination of an evolutionary algorithm with the rule base simplification described above. Therefore, it is necessary to minimize the following multi- objective function

$$f = (1 + \lambda S^*) f^*.$$

Here, $\lambda \in [-1, 1]$ is the weighting function determining whether similarity is rewarded ($\lambda < 0$) or penalized ($\lambda > 0$), and S^* the aggregated similarity measure for the total model defined by

$$S^* = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\max(S(A_{ij} A_{ik}))}{ns_i - 1} \right), \; j, k = 1, \ldots, ns_i, \; j \neq k,$$

where n denotes the number of inputs and ns_i the number of sets for each input variable.

The combination of an evolutionary algorithm (EAs) with the rule base simplification can lead to the modeling scheme given in Figure 1 (see [19]).

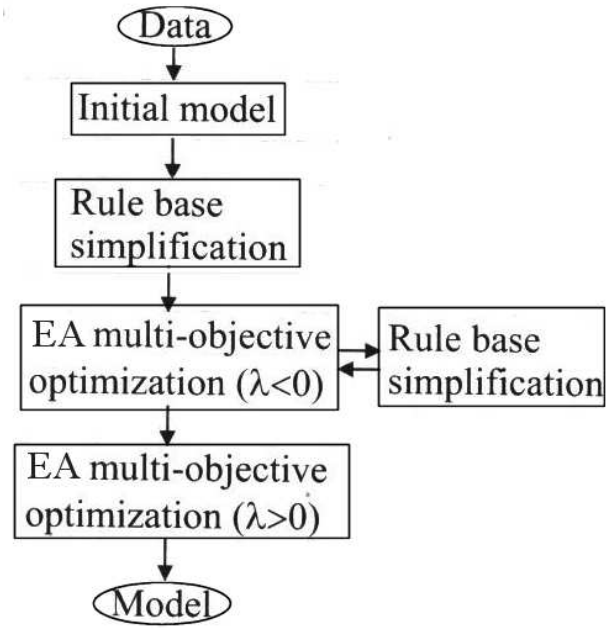

Figure 1: Modeling scheme of multi-objective optimization

4. Fuzzy model representation

Fuzzy models are commonly represented as a **population of chromosomes**. With a population size N, it is necessary to create chromosomes $\mathbf{c}_l$, $l = 1, \ldots, N$, each of them contains both the codes describing the fuzzy sets in the rule antecedents and the codes for the parameters of the rule consequents.

There are in principle two ways how to make a reasonable encoding of chromosomes [8] : (i) using binary strings and (ii) real number (floating-point) coding, the latter proved to be faster, more consistent from run to run, and provides a higher precision [13]. Therefore, real-coded chromosomes are used when optimizing fuzzy models.

For a TS model (1) of M fuzzy rules, an n-dimensional premise with triangular fuzzy sets (each determined by three parameters a, b, c) and $(n+1)$ parameters in each consequent function, any chromosome can be coded as (see [19]):

$$\mathbf{c}_l = (\mathrm{ant}_1, \ldots, \mathrm{ant}_M, \theta_1, \ldots, \theta_M), \; l = 1, \ldots, N,$$

where $\text{ant}_i = (a_{i1}, b_{i1}, c_{i1}, \ldots, a_{in}, b_{in}, c_{in})$ denotes the parameters of the antecedent fuzzy sets and θ_i the consequent parameters. Therefore, the total length of chromosome is $L = M(3n + (n+1)) = M(4n+1)$.
When using trapezoidal fuzzy sets (each defined by four parameters), the chromosome length is $L = M(5n+1)$.

The initial population of chromosomes can be written as $\mathcal{P}^0 = (\mathbf{c}_1^0, \ldots, \mathbf{c}_N^0)$, where $\mathbf{c}_1^0$ is the chromosome corresponding to the initial model created by minimizing (2) and the other chromosomes $\mathbf{c}_2^0, \ldots, \mathbf{c}_N^0$ are produced by random variations around $\mathbf{c}_1^0$.

5. Evolutionary algorithms

Evolutionary algorithms (EAs) are optimization algorithms inspired by Darwinian theory of natural evolution. They differ significantly from most of optimization algorithms; their basic features can be summarized as follows.

- EAs operate on codes of the individual parameters, not on their true values.
- EAs manipulate with a population of possible solutions when searching for a global optimum.
- EAs are gradient-free, which means that they use only the values of an objective function, not their derivatives.
- EAs use stochastic rules for the transition from one population to the next population.

In real-coded EA, all the parameters appear directly in chromosomes and are modified by using special evolutionary operators. Various EAs are reviewed in [3, 9].

5.1. Genetic algorithms

Genetic algorithms (GAs) were introduced by Holland [**?**] and their properties are analyzed in [5, 6, 13]. The algorithms guarantee the process of natural evolution by using so-called genetic operators, namely:

- selection operator,
- crossover operator,
- mutation operator.

The **selection operator** ensures the selection of well-performing chromosomes (chromosomes with a higher chance of surviving) for the reproduction process. To each chromosome $\mathbf{c}_l$, $l = 1, \ldots, N$, it is necessary to assign its **fitness** $\varphi(\mathbf{c}_l)$ so that the chromosomes with a low (high) objective-function value will have a high (low) value of their fitness. The assignment is usually realized as follows (see [6]: the individual chromosomes are first ordered according to the values of their objective function f^* into a non-decreasing sequence $\mathbf{c}^{(l)}, \ldots, \mathbf{c}^{(N)}$ and then the corresponding fitness values are assigned by the formula

$$\varphi(\mathbf{c}^{(j)}) = \frac{1}{1-N}[(1-\varepsilon)j + \varepsilon - N],$$

where ε is a small positive number, e.g. $\varepsilon = 0.5$. Therefore, the chromosome fitness is a decreasing linear function of the chromosome rank in the sequence defined above, whereas $\varphi(\mathbf{c}^{(1)}) = 1$, $\varphi(\mathbf{c}^{(N)}) = \varepsilon$. The selection of well-performing chromosomes is realized by using the biased roulette wheel method [13] so that each chromosome is assigned the segment of roulette wheel, the length of which is proportional to the chromosome fitness.

The following notation is accepted for defining the operators of crossover and mutation [19]: $r \in [0, 1]$ denotes a random number with uniform distribution, $t = 0, 1, \ldots,$ is the generation number,

$$\mathbf{c}_\nu = (\nu_1, \ldots, \nu_L) \text{ and } \mathbf{c}_\omega = (\omega_1, \ldots, \omega_L)$$

are the chromosomes selected for a given operation, $k \in \{1, 2, \ldots, L\}$ is the position of an element in the chromosome, and ν_k^{min}, ν_k^{max} are the lower and upper bounds, respectively, on the parameters encoded by element k.

The crossover operator creates two new chromosomes from a pair of selected chromosomes $(\mathbf{c}_\nu^t, \mathbf{c}_\omega)^t$ of the parents' generation. There are a number of special crossover operations for real-coded data:

- **Simple arithmetic crossover**: The the parents' chromosomes are crossed at the k-th position, the crossing position being selected at random from uniform discrete distribution on $[2, \ldots, N-1]$. Starting from the parents' chromosomes, the resulting offsprings are

$$\mathbf{c}_\nu^{t+1} = (\nu_1, \ldots, \nu_k, \omega_{k+1}, \ldots, \omega_L) \text{ and } \mathbf{c}_\omega^{t+1} = (\omega_1, \ldots, \omega_k, \nu_{k+1}, \ldots, \nu_L).$$

  In some cases two crossing sites are used for crossing the parents' chromosomes.
- **Whole arithmetic crossover**: Resulting offsprings are produced as a linear combination of the parents' ones. Therefore, the new chromosomes are

$$\mathbf{c}_\nu^{t+1} = r\mathbf{c}_\nu^t + (1-r)\mathbf{c}_\omega^t \text{ and } \mathbf{c}_\omega^{t+1} = r\mathbf{c}_\omega^t + (1-r)\mathbf{c}_\nu^t.$$

- **Heuristic crossover**: The parents' chromosomes are combined in such way that the offsprings are

$$\mathbf{c}_\nu^{t+1} = \mathbf{c}_\nu^t + r(\mathbf{c}_\omega^t - \mathbf{c}_\nu^t) \text{ and } \mathbf{c}_\omega^{t+1} = \mathbf{c}_\omega^t + r(\mathbf{c}_\nu^t - \mathbf{c}_\omega^t).$$

The mutation operator changes the value of a random selected element of the chromosome $\mathbf{c}_\nu^t$. The following operators are recommended for executing the mutation operation:

- **Simple uniform mutation**: One randomly selected element $\nu_k,\ k = 1, 2 \ldots, L$ of the chromosome $\mathbf{c}_\nu^t$ is replaced by another element $\nu_k^{'}$, which is the random number with uniform distribution on $[\nu_k^{min}, \nu_k^{max}]$. Therefore, the resulting chromosome can be written as $\mathbf{c}_\nu^{t+1} = (\nu_1, \ldots, \nu_k^{'}, \ldots, \nu_L)$.
- **Multiple uniform mutation**: Total number of n randomly selected element of the chromosome $\mathbf{c}_\nu^t$ are replaced by another elements taking values from permissible ranges, n being a random integer from $\{1, 2, \ldots, L\}$.
- **Gaussian mutation**: All the elements of the chromosome $\mathbf{c}_\nu^t$ are changed so that the resulting chromosome is $\mathbf{c}_\nu^{t+1} = (\nu_1^{'}, \ldots, \nu_k^{'}, \ldots, \nu_L^{'})$, where $\nu_k^{'} = \nu_k + u_k,\ k = 1, 2, \ldots, L$ and u_k are random numbers taken from a Gaussian distribution with mean $\mu = 0$ and an adaptive variance

$$\sigma_k^2 = \frac{T-t}{T}\frac{(\nu_k^{max} - \nu_k^{min})}{3}.$$

When optimizing fuzzy models, two types of constraints must be applied: partition constraints and search space constraints.

The **partition constraints** ensures that the fuzzy model can derive an output for all the occurring inputs by suppressing gaps in the partitions of input variables. It is apparent that the coding of each fuzzy set must fulfill the requirement $a \leq b \leq c$ and each pair of neighboring fuzzy sets are constrained by $a_R \leq c_L$, where L and R denotes left and right set, respectively. Moreover, these conditions must be verified after the creation of each new generation of chromosomes.

The **search space** is **constrained** by two user-defined bounds α_1 and α_2 [19]. The bound α_1, applied to the antecedent of the rules, allows the parameters describing the fuzzy sets A_{ij} to vary only within a range of $\pm\alpha_1|X_j|$ around their initial values, where $|X_j|$ denotes the length of the domain, on which the fuzzy sets A_{ij} are defined. The bound α_2, applied to the consequent parameters, allows the q-the consequent parameter of ith rule (p_{iq}) to vary within a range of $\pm\alpha_2(\max_i(p_{iq}) - \min_i(p_{iq}))$ around its initial value.

The search space constraints are coded in the vectors

$$\nu^{max} = [\nu_1^{max}, \ldots, \nu_L^{max}] \text{ and } \nu^{min} = [\nu_1^{min}, \ldots, \nu_L^{min}]$$

that define the upper and lower bounds, respectively, for each of the elements of a chromosome. It is necessary to take into consideration that operations oh heuristic crossover and Gaussian mutation can result in chromosomes violating the bounds.

The **stopping condition** for the model optimization can be defined in a few different ways:

- by entering the total number T of generations,
- by introducing a positive number ϵ such that the optimization is stopped, if the condition $f_{max} - f_{min} < \epsilon$ is fulfilled, f_{max} and f_{min} being maximum and minimum objective function values, respectively,
- by using an adaptive ϵ, the value of which decreases in the course of the optimization process [26].

Before starting the optimization algorithm, it is necessary to input a fuzzy rule base of the model and select the value of ϵ in the stopping condition (or the number of generations T), the population size N, and the constraints α_1 and α_2.

The **optimization algorithm** can be written (in rather simplified form) using a pseudo-code as follows:

```
1   set t = 0:
2   create the initial population P^t of size N ;
3   calculate the values of both f and φ for all chromosomes of P^t;
4   calculate the constraint vectors ν^max and ν^min;
5   repeat
6     select from P^t the mating pool for P^{t+1} using the selection operator;
7     create chromosomes of P^{t+1} using a crossover operator;
8     apply a mutation operator to a randomly selected chromosome of P^{t+1};
9     calculate the values of both f and φ for all chromosomes of P^{t+1};
10    set t = t + 1;
11  until stopping condition is fulfilled;
12  select the best chromosome from P^t
```

Comments.

- In the course of optimization, it is desirable to apply both the partition and search space constraints.
- It is also recommended to preserve the best chromosomes, i.e. the chromosomes with the lowest value of f, in the population (so-called elitist strategy).

5.2. Another evolutionary algorithms

In addition to GAs, there are a number of EAs frequently used for solving the problem of global optimization [3, 9]. In this Section, some of them (namely control random search, evolutionary search and differential evolution) are discussed in more detail, the special attention being paid to their tools for implementing the operations of selection, crossover and mutation.

The **controlled random search**, originally proposed by Price [17], is based on the well-known simplex method [15]. The initial population $\mathcal{P}$ is taken at random from a search space D with dimensionality of d. A new trial point $\mathbf{x}$ (chromosome in the terminology of GAs) is generated from a simplex ($d+1$ linearly independent points from $\mathcal{P}$) by the reflection operation

$$\mathbf{x} = \mathbf{g} - (\mathbf{z} - \mathbf{g}),$$

where $\mathbf{z}$ denotes a randomly taken pole of the simplex and $\mathbf{g}$ the centroid of the remaining $d+1$ poles of the simplex. Several modifications of the CRS algorithm are described in [1]. Our modification (**modified controlled random search**–MCRS) [10] consists of randomizing the reflection using the formula

$$\mathbf{x} = \mathbf{g} + U(\mathbf{g} - \mathbf{z_H}).$$

Here, the $\mathbf{z_H}$ is the point with maximum objective- function value in $\mathcal{P}$ and the multiplication factor U is a random variable distributed uniformly in $[c, \gamma - c)$, $\gamma > 0$ and c being input parameters, $0 < c < \gamma/2$. It is apparent that the reflection represents a "generalized" operation of crossover. No operation like mutation is applied.

The **evolutionary search** (ES) [11] algorithm is based on the following principles:

- The initial population is generated at random from the search space.
- The new population $\mathcal{P}^{t+1}$ inherits the properties of the old one $\mathcal{P}^t$ in two ways: (i) by surviving the best N_s points (the points with minimum objective-function values) and (ii) by applying the reflection to the points of $\mathcal{P}^t$.
- Each of the simplex poles can be selected with the probability proportional to its fitness.
- Mutation is allowed with a small probability p_m.

When compared with MCRS, there are two additional input parameters: mutation probability p_m and the number N_s of surviving points.

The **differential evolution** (DE) introduced by Storn and Price [23] is simple but powerful evolutionary algorithm for global optimization over the box-constrained search space. The algorithm can be written in pseudo-code as follows (see [27]):

```
1    create population P = (x_1, ..., x_N) from the search space;
2    repeat
3        for i := 1 to N do
4            compute a mutant point u;
5            create y by the crossover of u and x_i;
6            if f(y) < f(x_i) then  insert y into population Q
7                           else   insert x_i into Q
8            endif;
9        endfor;
10       P := Q;
11   until stopping condition;
```

Two versions of DE are described: DE-rand and DE-best. Regarding the version DE-rand, a mutant point $\mathbf{u}$ is generated according to the formula

$$\mathbf{u} = \mathbf{r}_1 + F(\mathbf{r}_2 - \mathbf{r}_3)\,,$$

where $\mathbf{r}_1, \mathbf{r}_2$ and $\mathbf{r}_3$ are three distinct points taken randomly from $\mathcal{P}$ and $F > 0$ is an input parameter.

The version DE-best generates the mutant point $\mathbf{u}$ as follows:

$$\mathbf{u} = \mathbf{x}_{min} + F\left(\mathbf{r}_1 + \mathbf{r}_2 - \mathbf{r}_3 - \mathbf{r}_4\right),$$

where $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ and $\mathbf{r}_4$ are four different points taken randomly from $\mathcal{P}$ (not coinciding with the current point $\mathbf{x}_i$)and $\mathbf{x}_{min}$ is the point of $\mathcal{P}$ with minimum objective-function value.

The elements y_j, $j = 1, 2, \ldots, d$, of a new point $\mathbf{y}$ are built up by the crossover of a randomly taken point $\mathbf{x}$ (not coinciding with the current $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$) and the mutant point $\mathbf{u}$ using the following rule:

$$y_j = \begin{cases} u_j & \text{if } U_j \le C \quad \text{or} \quad j = l \\ x_j & \text{if } U_j > C \quad \text{and} \quad j \ne l\,, \end{cases}$$

where l is a randomly chosen integer from $\{1, 2, \ldots, d\}$, $U_1, U_2, \ldots, U_d$ are independent random variables distributed uniformly in $[0, 1]$, and $C \in [0, 1]$ is an input parameter affecting the number of elements to be exchanged by crossover. A way of adapting the value of the scaling factor F during searching process was suggested by Ali and Törn [1].

We have recently proposed an adaptive optimizing algorithm (**controlled random search algorithm with competing heuristics**) [26]. The algorithm can be written in pseudo-code as follows:

```
1   create population P = (x_1, ..., x_N) from the search space ;
2   find the point x_max with the highest objection-function value;
3   repeat
4     generate a new point y by using a heuristic;
5     if f(y) < f(x_max) then
6       x_max := y;
7       find new x_max;
8     endif
9   until stopping condition;
```

The role of a heuristic at the line 4 can play any non-deterministic algorithm generating a new point $\mathbf{y}$ in the search space. There are many different heuristics that can be used (e.g. MCRS or DE) and, moreover, the heuristics can alternate during the course of optimizing process.
Several different versions of the CRS algorithm with competing heuristics [26, 27] were implemented in Matlab and tested on nonlinear regression tasks collected by NIST [16]. The experiments proved that the algorithms are more reliable and less time-consuming as compared not only with deterministic optimizing algorithms but also with the individual evolutionary algorithms used in themselves.

5.3. Convergence of evolutionary algorithms

First, let us briefly reformulate the optimization problem for the study of its convergence. Let $D \subset \mathcal{R}^d$. Denote by $\mathcal{L}$ the system of all Lebesgue measurable subsets of D and λ the Lebesgue measure on $\mathcal{L}$. Let f be a real Lebesgue measurable function defined on D. If f is not continuous it can be impossible to find good approximations of $\mathbf{x}^* = \arg\min_{\mathbf{x}\in D} f(\mathbf{x})$. Nevertheless, using evolutionary algorithms, it is always possible to find arbitrary good approximations of the value $\mu = inf\{t\,;\ \lambda(f^{-1}(-\infty, t)) > 0\}$, where $f^{-1}(A) = \{x \in D\,;\ f(x) \in A\ \}$, which is said to be the *essential minimum* of f. Thus our task is to find an arbitrary good approximation of μ.
Let p_0 be a probability measure on $(D, \mathcal{L})$ which is positive on each open subset of D. Let N be population size and let D^N be the set of all populations. Let π be a probability mapping defined on D^N assigning to each population $\mathcal{P}$ the probability measure $\pi(\mathcal{P})$ on $(D, \mathcal{L})$. Finally, let $\{pm_k\}$ be a sequence of numbers from interval $\langle 0, 1\rangle$ and let $\mathcal{C}$ be a rule according to which some points in the old population are replaced by new ones. We will consider a **generalized evolutionary algorithms** described as follows:

1: Generate an initial population $\mathcal{P}^0 = \{x_1, x_2, \cdots, x_N\}$ chosen as an independent identically distributed sample according to the probability measure p_0 and set $k = 0$.

2_k: Copy a portion of M best points of $\mathcal{P}^k$ directly into the new population. Here, the best points means the points at which the function f has its lowest values and M is an integer from the set $\{0, 1, 2, \ldots, N-1\}$.

3_k: Select a new point at random according to the probability measure $p_{k+1} = \pi(\mathcal{P}^k)$ and include it to the new population when a condition $\mathcal{C}$ is fulfilled. Repeat the procedure until the new population is complete.

4_k: Replace a randomly chosen point by its mutation with the probability pm_{k+1}.

5: If the stopping condition is not satisfied, set $k = k + 1$ and go back to 2_k.

By convergence of an algorithm we mean the convergence with probability 1, i.e. an algorithm is convergent if

$$P(\lim_{k\to\infty} min\{f(x);\ x \in \mathcal{P}_k\} = \mu) = 1.$$

Starting from the results by [22], we proved (in [14]) the necessary and sufficient condition for convergence of the generalized evolutionary algorithm. Regarding the algorithm without explicit mutation, the following theorem holds.

Theorem 1. For any measurable subset $H \subset D$ denote by $p_k(H)$ the probability that in the k-th step the algorithm produces a new trial point belonging to the set H. Suppose the algorithm saves the best point. Then the algorithm is convergent if and only if for every set $H \subset D$ of positive Lebesgue measure we have

$$\prod_{k=1}^{\infty}(1 - p_k(H)) = 0.$$

Remark. Notice that according to [24] the last condition is equivalent to the following one: The series $\sum_{k=1}^{\infty} p_k(H)$ is divergent.

Let us consider the generalized evolutionary algorithm with explicit mutation. In such case the mutation can violate the convergence condition. As regards the case, we proved the following theorem:

Theorem 2. Let for each set $H \in D$ of positive Lebesgque measure, the probability that the new point created by mutation belongs to H is positive. Suppose the mutation excludes the best point and the series $\sum_{k=1}^{\infty} pm_k$ is divergent. Then the generalized evolutionary algorithm is convergent provided that the corresponding algorithm without explicit mutation saves the best point.

Let us consider the convergence of the controlled random search algorithm with competing heuristics. Let h denotes the total number of heuristics to be used. Denote by π_i the probability mappings used in the i-th heuristic and let $\mathcal{A}_i$ be a simple algorithm using the probability mapping π_i, $i = 1, 2, \ldots, h$. An algorithm such that the i-th heuristics can be used in k-th step with probability $q_i(k)$; $i = 1, 2, \ldots, h$; $k = 1, 2, \ldots$; $\sum_{i=1}^{h} q_i(k) = 1$, is said to be random combination of algorithms $\mathcal{A}_i$; $i = 1, 2, \ldots, h$. It can be seen easily that any such random combination of evolutionary algorithms is again an evolutionary algorithm using probability mapping $\pi = \sum_{i=1}^{h} q_i \pi_i$.

For the convergence of the algorithm $\mathcal{A}$ obtained by random combination of h algorithms $\mathcal{A}_i$; $i = 1, 2, \ldots, h$, we have the theorem (see [26])

Theorem 3. Suppose that there exists $\delta > 0$ such that $q_i(k) \geq \delta$ for all $i = 1, 2, \ldots, h$; $k = 1, 2, \ldots$. Then the algorithm $\mathcal{A}$ is convergent if and only if for every $H \subset D$ of positive Lebesgue measure there exists an integer $j \in \{1, 2, \ldots, h\}$ such that the series $\sum_{k=1}^{\infty} \pi_j(\mathcal{P}^k)(H)$ is divergent. Here $\mathcal{P}^k$ denotes the population in the k-th step.

6. Conclusions

This contribution shows how to construct TS fuzzy models from data as well as how to optimize them by using GAs. The special attention is paid to the possibility of using another EAs (CRS, MCRS, DE-rand, DE-best, and CRS with competing heuristics) for optimizing fuzzy models. We keep at disposal the procedure library for the model optimization and suppose that the particular tasks will be solved in a cooperation with the Institute for Research and Applications of Fuzzy Modeling.

References

[1] Ali, M.M., Törn, A. Population set based global optimization algorithms: Some modifications and numerical studies. Computers and Operations Research, vol. 31, 2004, pp. 1703 – 1725.

[2] Babuska, R. Fuzzy Modeling for Control. Boston: Kluwer Academic Publishers 1998.

[3] Bäck, T. and Schwefel, H.P. (1993). An Overview of Evolutionary Algorithms for Parameter Optimization, *Evolutionary Computation* **1**, 1993, pp. 1–23.

[4] Bezdek, J. C. Pattern Recognition with Fuzzy Objective Functions. New York: Plenum Press 1981.

[5] Davis, L. (Ed.). Handbook of Genetic Algorithms. New York: Van Nostrand Reihold 1991.

[6] Goldberg, D. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading: Addison-Wesley 1989.

bibitemHolland Holland, J. Adaptation in Natural and Artificial Systems. Ann Arbor: Univ. of Michigan Press 1975.

[7] Ishibuchi, H., Nakashima, T., Murata, T. Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics, vol. 29, 1999, pp. 601–618.

[8] Kalyanmoy Deb. Encoding and decoding functions. In: Evolutionary Computattion 2. Advanced Algorithms and Operators. (Bäck, T., Fogel, D. B., Michalewicz, Z. (Eds.). Bristol and Philadelphia: Institute of Physics Publishing 2000.

[9] Kvasnička, V., Pospíchal, J., Tiňo, P. Evolutionary Algorithms (in Slovak). Bratislava: STU Bratislava 2000.

[10] Křivý, I., Tvrdík, J. The controlled random search algorithm in optimizing regression models. Comp. Statist. & Data Anal., vol. 20, no. 2, 1995, pp. 229–234.

[11] Křivý, I., Tvrdík, J. and Krpec, R. Stochastic algorithms in nonlinear regression. Comp. Statist. & Data Anal., vol. 33, no. 3, 2000, pp. 277–290.

[12] Mamdani, E. H. Application of fuzzy algorithms of a simple dynamic plant. In: Proceedings IEE, no. 121, 1974, pp. 1585–1588.

[13] Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Springer Verlag 1992.

[14] Mišík, L. On Convergence of a Class of Stochastic Algorithms. In: Proceedings of 6th International Conference on Soft Computing. Brno: Technical University Press 2000, pp. 97–100.

[15] Nelder, J. A., Mead, R. A simplex method for function mimimization. Computer J., vol. 7, 1964, pp. 308–313.

[16] Statistical Reference Datasets. Nonlinear regression. NIST Information Technology Laboratory, 2001, http://www.itl.nist.gov/div898/strd/.

[17] Price, W. L. A controlled random search procedure for global optimization. Computer J., vol. 20, 1976, pp. 367–370.

[18] Price, K.V., Storn, R., Lampinen J. Differential Evolution: A Practical Approach to Global Optimization. Springer-Verlag 2005.

[19] Roubos, H., Setnes, M. Compact Fuzzy Models and Classifiers through Model Reduction and Evolutionary Optimization. In: Lance Chambers (Ed.) The Practical Handbook of Genetic Algorithms Applications , chapter 2. New York: Chapman & Hall/CRS 2000, pp. 31–59.

[20] Setnes, M., Babuska, R., Kaymak, U., van Nauta Lemke, H. R. Similaarity measures in fuzzy rule base simplification. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics, vol. 28, 1998, no. 3, pp. 376–386.

[21] Setnes, M., Babuska, R., Verbruggen, H. B. Rule-based modeling: Precision and transparency. IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews, vol. 28, 1998, pp. 165–169.

[22] Solis, F. J., Wets, R. J-B. Minimization of Random Search Techniques. Mathematics of Operations Research, vol. 6, 1981, pp. 19–30.

[23] Storn R., Price K. (1997) *Differential Evolution – a Simple and Efficient Heuristic for Global Optimization.* J. Global Optimization **11**, 341 – 359.

[24] Šalát, T. Infinite Series (in Slovak). Prague: Academia 1974.

[25] Takagi, T., Sugeno, M. Fuzzy identification of systems and its application to modeling and control. IEEE Transactions on Systems, Man and Cybernetics, vol. 15, 1985, pp. 116–132.

[26] Tvrdík, J., Křivý, I., Mišík, L. Adaptive population-based search: application to estimation of nonlinear regression parameters. Comp. Statist. & Data Anal., to appear in 2007.

[27] Tvrdík, J., Křivý, I. Competitive self-adaptation in evolutionary algorithms. This paper submitted to the 5th Conference on the European Society for Fuzzy Logic and Technolgy to be held in Ostrava.

Acknowledgement: This work was supported by the grant 201/06/0612 of the Czech Grant Agency as well as by the Research Scheme MSM 6198898701 of the Czech Ministry of Education, Youth and Sport.