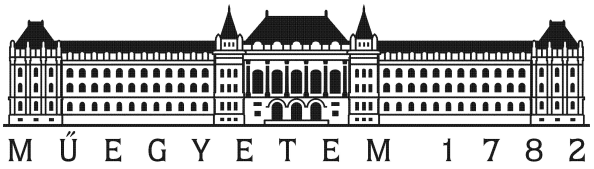

Identification of intelligent computational models by evolutionary and gradient based learning algorithms

Ph.D. Thesis Booklet

János Botzheim

Supervisor:
László T. Kóczy, Ph.D., D.Sc.

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Media Informatics

Budapest, 2007

1 Introduction, objectives

The computational intelligence methods play increasingly significant role in engineering and other applied systems modelling, control and in performing decision making and optimisation tasks related to them. Such intelligent computational models were created which can help efficiently solve high complexity modelling, control, and optimisation tasks. There are several engineering areas and applied science, e.g. control, telecommunication, logistics where the usage of classical mathematical analytical model is precluded or it would be too complex in the computational sense. The tasks can be divided into different categories according to mathematical intractability. In the first category there are those tasks where the analytical model of the system is at least theoretically known. These are problems where the algorithm needed for the solution is not tractable in theoretical sense, meaning that its computational complexity is higher than polynomial, or such problems where although the model is known, but no closed solution can be given for them. In the second group there are those problems where the analytical model is not known, perhaps it does not even exist at all, because the system's behaviour is non-deterministic. In the third, most untractable category there are those tasks where the human psyche, and the personal element are important. These are, for example, the human-machine systems. It is interesting that lots of these problems can be solved by human experts even if their solution is only suboptimal.

Scientists and engineers have tried to find automatisms for such problems for long. The solution became possible only after the invention of computers that caused a big impulse for researching such methods which try to imitate the elements of human intelligence. In the 1950s classical artificial intelligence methods appeared which were based on symbolic logic and they provided solutions for small-size, demonstrative tasks, but not for real-world problems, because they could not overcome the limitations of computational complexity. In the mid 1960s several modern intelligent methods were proposed which were called later soft computing. The aim of these methods is to provide good approximate solutions for the problems mentioned above taking care of the not too big computational complexity at the same time. The three main components of soft computing are the fuzzy systems [8, 9], the artificial neural networks [6, 22], and the different kind of evolutionary algorithms [4, 5].

Within the wide research area of computational intelligence, the investigation of model identification methods is one of the most current research topics, which goal is to determine different parameters and structure of computational intelligence based approximate models such as fuzzy rule bases and artificial neural networks. The goal of the thesis is to develop identification methods based on numerical data that can produce results better in terms of quality criteria (e.g. mean square error) relevant for the given applications than other techniques known from the literature. Model identification contains basically two steps. In the first phase the task is to find the minimal set of variables describing right the given engineering problem. In the second phase a suitable model in terms of a given criterion is determined in the state space of these variables. In this thesis I mainly deal with the second phase, especially for those problems where the identification of an arbitrary dimensional static model is based on numerical data. I determine models built up by different techniques by different identification methods, which can be alternatives for each other or one may be more favourable than the others. The thesis discusses mostly the identification of fuzzy rule based

systems. The aim is to create a (quasi)-optimal rule base in terms of a chosen quality criterion based on measured or observed numerical sample data. Usually in the practice there are numerical input-output data available which can be used for identifying the rule based model more or less precisely. There may be such situations, too, where an initial rule base created by human expert(s) has to be (quasi)-optimised, fine-tuned based on numerical patterns. The model identification is actually an optimisation problem and for this purpose beside the learning algorithms based on different classical mathematical techniques such as the calculation of the gradient, there is an increasingly interest in evolutionary optimisation techniques offering fast and acceptable convergence. A novel improvement of one of these methods, namely the bacterial evolutionary algorithm is discussed in the thesis. I show the applicability of the new method for the identification of Mamdani-type fuzzy rule bases [13]. Beside the bacterial approach I propose a gradient based learning algorithm, namely the Levenberg-Marquardt algorithm [11, 14] for optimising the fuzzy rule base. I also propose a completely new, combined method, the Bacterial Memetic Algorithm, which is a combination of the Bacterial Evolutionary and the Levenberg-Marquardt algorithm. In addition to the extraction of fuzzy rule based models by numerical data, another aim of the thesis is the identification and parameter optimisation of some kind of artificial neural networks. I propose a new technique called Bacterial Programming for the design process of B-spline neural networks [1]. This approach combines the advantages of genetic programming and bacterial evolutionary algorithm. Finally, an objective of the thesis is the investigation of the first phase of the model identification process, where the task is to find the minimal set of variables describing right the given engineering problem.

2 Brief overview of the applied techniques

In this section I briefly introduce the most frequently used techniques in the thesis, namely Fuzzy Systems, Bacterial Evolutionary Algorithms, and the Levenberg-Marquardt method.

2.1 Fuzzy systems

Human reasoning and other related phenomena cannot be accurately described by binary logic. The desire for extending binary logic to multiple valued ones came up a long time ago. The basic idea was that instead of using the ,,true" and ,,false" logical values only, the use of other values between true and false should also be allowed. There are many statements that cannot be evaluated as true or false, only their respective ,,degree of truth" can be determined. This idea led L. A. Zadeh to the creation of fuzzy logic in 1965 [21]. Nowadays, in computers and in many areas of life, the classical binary (Aristotlean or Boolean) logic is used. However, if we want to create more intelligent tools, better results can be achieved if the behaviour of the systems is described in a way that is closer to human thinking. Fuzzy logic is an extension of the classical logic. A fuzzy logic variable may assume any value between 0 and 1. Here, 0 means that the statement is ,,totally false", while 1 means that it is ,,totally true". According to this definition, the value 0.5 corresponds to ,,half true", and the value 0.9 to ,,almost true". The operations of classical logic can also be extended to fuzzy logic.

Based on this concept, fuzzy sets can be defined, fuzzy rules and fuzzy inference systems can be created. A fuzzy set A defined over a universe of discourse X is characterised by the so-called membership function (which is the extension of the characteristic function of ordinary sets). The membership function assigns a real value from the closed unit interval to every element of X describing the degrees for elements x to which they are belonging to the fuzzy set A. There are some well-known inference techniques, however, the Mamdani method is the most commonly used one in practical applications [13]. The inference engine may be viewed as a special kind of generalised function generator as it maps the set of all possible input fuzzy sets into the set of all possible fuzzy outputs. At the beginning of the inference the degree of matching between the observation and the rules is determined. Each component of the observation vector is compared to the same component of the antecedent of each rule. After the degree of matching has been calculated in each dimension, the resultant for the whole antecedent is determined. The degree of applicability of a rule is affected by the degree of matching of its each dimension. The degree of matching shows how important the role of the rule will be in the calculation of the conclusion for the given observation. After the degree of firing was determined for each rule, each conclusion is separately calculated. This can be made by cutting the consequent fuzzy set of the rule at height of the firing degree. The conclusion for the whole rule base can be computed by taking the union of the previously calculated sub-conclusions. After this, the output is converted to a so-called „crisp" value by a defuzzification method. There are many different defuzzification methods described in the literature, in this thesis the Centre of Gravity (COG) method is applied, which is one of the most commonly used defuzzification techniques in practical applications. Beside the Mamdani type fuzzy system in the thesis I also deal with Takagi-Sugeno type systems [20], where the consequent part of the rules is an input-output function rather than a fuzzy set.

2.2 Bacterial evolutionary algorithms

There are several optimisation methods inspired by processes in the nature. The advantage of these algorithms is their ability to solve and quasi-optimise problems with non-linear, high-dimensional, multi-modal, and discontinuous character. It has been shown that evolutionary algorithms are efficient tools for solving non-linear, multi-objective and constrained optimisations. These algorithms have the ability to explore large admissible spaces, without demanding the use of derivatives of the objective functions, such as the gradient-based training methods. Their principles are based on the search for a population of solutions, which tuning is done using mechanisms similar to biological recombination. The original Genetic Algorithm was developed by Holland [7] and was based on the process of evolution of biological organisms. A more recent approach is the *Bacterial Evolutionary Algorithm* which was inspired by the microbial evolution phenomenon [16, 17].

First the initial (random) bacteria population is created. The population consists of N_{ind} bacteria. After this there is an evolutionary cycle with N_{gen} generations, where in each generation the bacterial mutation is applied to each bacterium one by one, and the gene transfer is applied in the population.

The *bacterial mutation* is applied to each bacterium one by one. First, N_{clones} copies (clones) of the individual are generated. Then a certain part of the bacterium is randomly

selected and the parameters of this selected part are randomly changed in each clone (mutation). Next all the clones and the original bacterium are evaluated by an error criterion. The best individual transfers the mutated part into the other individuals. This cycle is repeated for the remaining parts, until all parts of the bacterium have been mutated and tested. At the end the best rule base is kept and the remaining N_{clones} are discharged. At the end of this process we get a better individual than the original one, or in the worst case, when all mutation cycles were unsuccessful the individual will be the same as at the beginning.

The *gene transfer* operation allows the recombination of genetic information between two bacteria. First the population must be divided into two halves. The better bacteria are called the superior half, the other bacteria are called the inferior half. One bacterium is randomly chosen from the superior half, this will be the source bacterium, and another is randomly chosen from the inferior half, this will be the destination bacterium. A part from the source bacterium is chosen and this part will overwrite a part of the destination bacterium or simply it will be added, if the length of the individuals can be different. This cycle is repeated for N_{inf} times, where N_{inf} is the number of „infections" per generation.

2.3 Levenberg-Marquardt algorithm

The Levenberg-Marquardt method was proposed originally by Levenberg [11] and Marquardt [14] for least-square estimation of non-linear parameters. The method can be used for adjusting the weights of neural networks [18] and for other optimisation problems as well.

In each step of the algorithm the calculation of the *Jacobian matrix* is needed, which contains the partial derivatives of the output computed by the patterns of the given network, system or optimisation problem according to the parameters (weights in case of neural networks):

$$\mathbf{J} = \frac{\partial y(\underline{x}^p)}{\partial \underline{w}^T}.$$

For calculating the parameter values as $\underline{w}[k+1] = \underline{w}[k] + \underline{s}[k]$, $\underline{s}[k]$, the LM update, is given as the solution of:

$$(\mathbf{J}^T[k]\mathbf{J}[k] + \alpha\mathbf{I})\underline{s}[k] = -\mathbf{J}^T[k]\underline{e}[k], \tag{1}$$

where $\underline{e}[k]$ is the error vector in the k-th step, which is $\underline{e}[k] = \underline{y}[k] - \underline{t}[k]$, the difference between the output calculated by the model (e.g. neural network) and the desired output for each pattern.

In equation (1) α is a regularisation parameter, which controls both the search direction and the magnitude of the update. The search direction varies between the Gauss-Newton direction and the steepest direction, according to the value of α. If $\alpha \to 0$, then the algorithm converges to the Gauss-Newton method, if $\alpha \to \infty$, then it gives the steepest descent approach.

Equation (1) can be recast as:

$$\underline{s}[k] = -\begin{bmatrix} \mathbf{J}[k] \\ \sqrt{\alpha}\mathbf{I} \end{bmatrix}^+ \begin{bmatrix} \underline{e}[k] \\ \underline{0} \end{bmatrix}, \tag{2}$$

where operator $^+$ means the pseudo-inverse of the matrix. The complexity of the calculation of $\underline{s}[k]$ is $O(n^3)$, where n is the number of columns of the Jacobian matrix.

In each iteration step the determination of the elements of the Jacobian matrix is needed, i.e. the partial derivatives need to be calculated.

The value of parameter α is changeable during the optimisation process. Thus, the k-th iteration step of the algorithm based on [3] is as follows:

1. Given $\underline{w}[k]$ and $\alpha[k]$
   (In the initialisation arbitrary positive value for α can be chosen, thus $\alpha[1] > 0$)
2. Determining $\mathbf{J}[k]$ and $\underline{e}[k]$
3. Calculation of $\underline{s}[k]$ based on (2)
4. Calculation of the so-called trust region, $r[k]$ as follows: $r[k] = \frac{E(\underline{w}[k]) - E(\underline{w}[k] + \underline{s}[k])}{E(\underline{w}[k]) - \frac{1}{2}\|\mathbf{J}[k]\underline{s}[k] + \underline{e}[k]\|^2}$, where $E = \frac{\|\underline{e}[k]\|^2}{2}$
5. The value of parameter α is adjusted dynamically depending on the value of $r[k]$:
   - If $r[k] < 0.25$ then $\alpha[k+1] = 4\alpha[k]$
   - If $r[k] > 0.75$ then $\alpha[k+1] = \alpha[k]/2$
   - Else $\alpha[k+1] = \alpha[k]$
6. If $r[k] \leq 0$ then $\underline{w}[k+1] = \underline{w}[k]$, else $\underline{w}[k+1] = \underline{w}[k] + \underline{s}[k]$

If the stopping condition is fulfilled or a predefined maximal generation number is reached then the algorithm stops, otherwise it continues with the $(k+1)$-th iteration step.

3 New methods and approaches in the identification of intelligent computational models

In this section I briefly introduce my own results and the statements based on the results.

3.1 Applying bacterial evolutionary algorithm in conjunction with rule reduction operators

In this subsection I introduce the improvement of the bacterial evolutionary algorithm proposed by Nawa and Furuhashi [16]. Instead of the triangular shaped membership functions I use the more general and widely used trapezoidal shaped membership functions, and I also propose novel rule reduction operators which can help optimise the size of the rule base, too.

Statement group 1. *I have introduced the bacterial evolutionary algorithm in conjunction with novel rule reduction operators for optimising Mamdani type fuzzy rule base using trapezoidal shaped membership functions and COG type defuzzification method.*

My publications related to this statement can be found under [B6, B7, B8, B12, B18, B20].

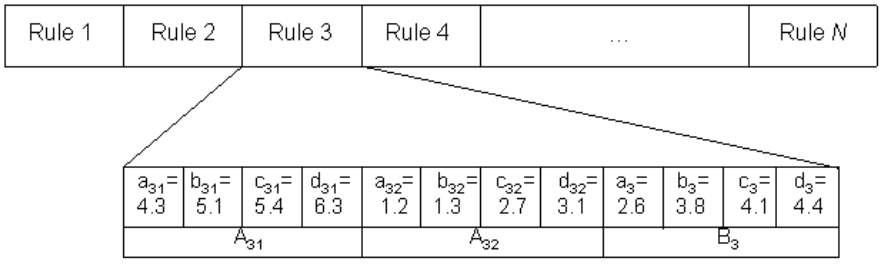

Figure 1: *The encoding method*

3.1.1 The proposed algorithm

The first step is to determine how the problem can be encoded in a bacterium (chromosome). The task is to find the optimal fuzzy rule base for a pattern set. Thus, the parameters of the fuzzy rules must be encoded in the bacterium. The parameters of the rules are the breakpoints of the trapezoids, thus, a bacterium will contain these breakpoints. Each rule has $4(n+1)$ parameters, where n is the number of rule inputs. For example, the encoding method of a fuzzy system with two inputs and one output can be seen in Figure 1. In Figure 1 the number of rules in the rule base is N. For example Rule 3 in Figure 1 means:

$$R_3: \quad \text{If } x_1 \text{ is } A_{31}(4.3; 5.1; 5.4; 6.3) \text{ and } x_2 \text{ is } A_{32}(1.2; 1.3; 2.7; 3.1) \text{ then } y \text{ is } B_3(2.6; 3.8; 4.1; 4.4).$$

In the evolutionary process between the bacterial mutation and gene transfer step I propose using rule reduction operators for each individuals.

When a membership function becomes too narrow, the rule using it must be deleted. The evaluation criterion is as follows:

$$l_i \mu_i \left(\frac{a_j + b_j + c_j + d_j}{4} \right) \geq \beta l_j, \tag{3}$$

where l_i and l_j are the lengths of the medians of the membership functions of the given input or output variables in the i-th and j-th rule, and β is the annihilation parameter. The larger is the value of β the more severe is the annihilation criterion.

If two membership functions, belonging to the same variable are near to each other, and the difference between the lengths of their median is small enough, then they are fused in a single membership function. There are two criteria of fusion:

$$\left| \frac{l_i}{l_j} - 1 \right| < \gamma \text{ and } |f| < \gamma, \tag{4}$$

where l_i and l_j are the lengths of the median of the membership functions of the given variable in the i-th and j-th rule, and f is the distance between the centres of l_i and l_j. To execute fusion both criteria must be satisfied, but only one parameter, γ is used. The smaller

is the value of γ the more severe is the criterion of fusion. The parameters of the fused membership function will be:

$$z_{fus} = \frac{z_i l_i + z_j l_j}{l_i + l_j}, \tag{5}$$

where z stands for the four breakpoints (a, b, c and d) of a trapezoidal membership function.

If two rules have the same antecedents but a different consequent, the membership functions of the output variable are fused in one output membership function by using Equation (5).

If two rules are identical the second one is deleted. After the fusion operators it can happen that the antecedent parts of two rules become the same. In this case the algorithm merges their consequent parts, thus the two rules become identical, one of them can be eliminated. So the effect of the fusions can be seen after applying these two other operators.

Statement 1.1. *I have developed novel rule reduction operators for optimising the size of the rule base.*

3.1.2 Testing the method

I have done simulations in order to test and analyse the operators in the algorithm. I applied the following widely used test function, where the goal is to approximate this six dimensional function as close as possible:

$$y = x_1 + \sqrt{x_2} + x_3 x_4 + 2e^{2(x_5 - x_6)}, \tag{6}$$

where $x_1 \in [1; 5]$, $x_2 \in [1; 5]$, $x_3 \in [0; 4]$, $x_4 \in [0; 0.6]$, $x_5 \in [0; 1]$, $x_6 \in [0; 1.2]$.

In the bacterial operators I used the following error function to evaluate the individuals and I used this function to compare the simulation results:

$$E = \frac{1}{m} \sum_{i=1}^{m} \frac{|t_i - y_i|}{I_{max} - I_{min}}, \tag{7}$$

where t_i is the desired output for the i-th pattern, y_i is the output of the model for the i-th pattern, m is the number of patterns, I_{max} is the upper and I_{min} is the lower bound of the interval of the output variable, so the error is normalised by the length of the output interval rather than the actual value of the output.

The simulation results for the best bacterium can be seen in Table 1. The fusion parameter is $\gamma = 5\%$, the initial rule number is 10, the number of individuals is 10, the number of clones is 20, the number of infections is 4. The table shows the average result of 10 runnings. With the increase of the value of β the remaining rule number increased too, because the criteria of annihilation became more severe.

Statement 1.2. *I have developed a computer program for bacterial evolutionary algorithm in conjunction with novel rule reduction operators which is able to optimise the rule base of Mamdani type fuzzy rule based systems using trapezoidal shaped membership functions.*

Statement 1.3. *I have performed simulations for a reference problem used in the literature in order to analyse the effects of the rule reduction operators.*

Table 1: *Effect of the annihilation parameter* β

β	average number of rules	training error	test error
5	4.7	5.23	8.94
6	5.4	5.02	8.47
8	7.6	4.52	8.32
10	7.8	4.13	8.17

3.2 The application of the Levenberg-Marquardt algorithm for optimising Mamdani type fuzzy rule bases

The Levenberg-Marquardt (LM) algorithm is a general optimisation method for solving such problems where the derivatives of the objective function are known. The algorithm has good convergence properties in local search. In this subsection I introduce the novel application of the algorithm for determining the parameters of Mamdani type fuzzy rule base using trapezoidal shaped membership functions and COG type defuzzification method.

Statement group 2. *I have proposed a new local optimisation algorithm, namely the Levenberg-Marquardt algorithm for determining the parameters of Mamdani type fuzzy rule base using trapezoidal shaped membership functions and COG type defuzzification method, and I have shown that the method has more favourable properties than other techniques known from the literature.*

My publications related to this statement can be found under [B2, B9, B16, B17, B18, B20].

3.2.1 Determination of the Jacobian matrix

In each iteration step of the algorithm the calculation of the *Jacobian matrix* is needed, all partial derivatives have to be computed. The number of rules is constant. Trapezoidal membership functions are used which can be written in the following form:

$$\mu_{ij}(x_j) = \frac{x_j - a_{ij}}{b_{ij} - a_{ij}} N_{i,j,1}(x_j) + N_{i,j,2}(x_j) + \frac{d_{ij} - x_j}{d_{ij} - c_{ij}} N_{i,j,3}(x_j). \tag{8}$$

In this equation a_{ij}, b_{ij}, c_{ij}, d_{ij} denote the four parameters of the membership function belonging to the i-th rule and the j-th input variable. The a_i, b_i, c_i, d_i values belong to the output variable in the i-th rule, $N_{i,j,k}$ functions are rectangular window functions. As in the original Mamdani algorithm, the standard t-norm (min) is used in the inference mechanism meaning that the degree of matching of the i-th rule in case of an n-dimensional crisp $\underline{x}$ vector is:

$$w_i = \min_{j=1}^{n} \mu_{ij}(x_j). \tag{9}$$

The output of the fuzzy inference:

$$y(\underline{x}) = \frac{1}{3} \frac{\sum_{i=1}^{R} 3w_i(d_i^2 - a_i^2)(1 - w_i) + 3w_i^2(c_i d_i - a_i b_i) + w_i^3(c_i - d_i + a_i - b_i)(c_i - d_i - a_i + b_i)}{\sum_{i=1}^{R} 2w_i(d_i - a_i) + w_i^2(c_i + a_i - d_i - b_i)}. \tag{10}$$

The number of rules is R, and n is the number of input dimensions. One row of the Jacobian matrix belongs to one pattern. The p-th row can be written in the following form:

$$\mathbf{J} = \left[\frac{\partial y(\underline{x}^{(p)})}{\partial a_{11}} \quad \frac{\partial y(\underline{x}^{(p)})}{\partial b_{11}} \cdots \frac{\partial y(\underline{x}^{(p)})}{\partial a_{12}} \cdots \frac{\partial y(\underline{x}^{(p)})}{\partial d_1} \cdots \frac{\partial y(\underline{x}^{(p)})}{\partial d_R} \right], \tag{11}$$

where p is the identifier of the pattern and

$$\frac{\partial y(\underline{x}^{(p)})}{\partial a_{ij}} = \frac{\partial y}{\partial w_i} \frac{\partial w_i}{\partial \mu_{ij}} \frac{\partial \mu_{ij}}{\partial a_{ij}} \tag{12}$$

for the first breakpoint of the trapezoid and similarly for the other b_{ij}, c_{ij} and d_{ij} parameters, too. From Equation (9) it can be seen that the w_i values depend only on the membership functions, and each membership function depends only on four parameters (breakpoints). So, the derivatives of w_i will be:

$$\frac{\partial w_i}{\partial \mu_{ij}} = \begin{cases} 1, & \text{if } \mu_{ij} = \min_{k=1}^{n} \mu_{ik} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The derivatives of the membership functions for parameter a_{ij} will be:

$$\frac{\partial \mu_{ij}}{\partial a_{ij}} = \frac{x_j^{(p)} - b_{ij}}{(b_{ij} - a_{ij})^2} N_{i,j,1}(x_j^{(p)}) \tag{14}$$

and can be calculated similarly for the other parameters as well. $\frac{\partial y}{\partial w_i}$ and the derivatives of the output membership functions parameters have to be also computed. From Equation (10) the following can be written:

$$\frac{\partial y}{\partial w_i} = \frac{1}{3} \frac{D \frac{\partial F_i}{\partial w_i} - N \frac{\partial G_i}{\partial w_i}}{D^2} \tag{15}$$

and similarly for the other a_{ij}, b_{ij}, c_{ij} and d_{ij} parameters, where D is the denominator of Equation (10) and N is its numerator. F_i is the i-th member of the sum in the numerator, G_i is the i-th member of the sum in the denominator. The derivatives will be as follows:

$$\begin{aligned} \frac{\partial F_i}{\partial w_i} &= 3(d_i^2 - a_i^2)(1 - 2w_i) + 6w_i(c_i d_i - a_i b_i) + 3w_i^2[(c_i - d_i)^2 - (a_i - b_i)^2] \\ \frac{\partial G_i}{\partial w_i} &= 2(d_i - a_i) + 2w_i(c_i + a_i - d_i - b_i) \end{aligned} \tag{16}$$

$$\begin{aligned} \frac{\partial F_i}{\partial a_i} &= -6w_i a_i + 6w_i^2 a_i - 3w_i^2 b_i - 2w_i^3(a_i - b_i) & \frac{\partial G_i}{\partial a_i} &= -2w_i + w_i^2 \\ \frac{\partial F_i}{\partial b_i} &= -3w_i^2 a_i + 2w_i^3(a_i - b_i) & \frac{\partial G_i}{\partial b_i} &= -w_i^2 \\ \frac{\partial F_i}{\partial c_i} &= 3w_i^2 d_i - 2w_i^3(d_i - c_i) & \frac{\partial G_i}{\partial c_i} &= w_i^2 \\ \frac{\partial F_i}{\partial d_i} &= 6w_i d_i - 6w_i^2 d_i + 3w_i^2 c_i + 2w_i^3(d_i - c_i) & \frac{\partial G_i}{\partial d_i} &= 2w_i - w_i^2. \end{aligned} \tag{17}$$

The number of columns of the Jacobian matrix is $4(n+1)R$, the number of rows is equal to the number of patterns.

Statement 2.1. *I have determined the partial derivatives of the output of the fuzzy inference systems using trapezoidal shaped membership functions and COG type defuzzification method according to the parameters of the model, and I have also determined the whole Jacobian matrix based on these derivatives.*

3.2.2 The application of the method

I have performed simulations for reference problem used in the literature and I have compared the new method with the error-backpropagation (BProp) method [19].

The first reference problem is the so-called *pH problem*, which is a one-dimensional problem. The aim of this example is to approximate the inverse of a titration-like curve. The other reference problem is the Inverse Coordinate Transformation (ICT) problem. This two dimensional example illustrates an inverse kinematical transformation between two Cartesian coordinates and one of the angles of a two-links manipulator. The training set contains 101 patterns for the pH problem, 110 patterns for the ICT problem. Mean Square of Error (MSE) is used as error criterion to test the algorithm.

In the first experiment only two membership functions parameters will be adjusted. These are the b and c parameters belonging to the first input membership function of the first rule. The results for the ICT problem can be seen in Table 2. From this table we can conclude that the LM converges much faster than the BProp method. In the second experiment the aim is to estimate every parameter in the fuzzy rule based system. The LM approach provided good results in this case, too.

Statement 2.2. *I have developed a new computer program for local optimisation of the parameters of Mamdani type fuzzy rule base using trapezoidal shaped membership functions and COG type defuzzification method.*

Statement 2.3. *I have performed investigations by applying simulations for modelling reference problems obtained from the literature. I found that the new method thus developed had given much better convergence results in terms of mean square error than the backpropagation algorithm in all of the investigated cases. Based on these results and on theoretical considerations I expect that the advantages of my proposed new method will appear in other specific applications, too.*

Table 2: *Summary of the results for the ICT problem*

method	initial point	final point	initial MSE	final MSE	iterations
LM	{0.40; 0.70}	{0.100; 0.239}	0.890	0.8650	4
LM	{0.30; 0.40}	{0.190; 0.256}	0.870	0.8651	4
LM	{0.50; 0.65}	{0.113; 0.218}	0.890	0.8653	7
BProp	{0.40; 0.70}	{0.155; 0.301}	0.890	0.8652	21
BProp	{0.30; 0.40}	{0.162; 0.279}	0.870	0.8650	14
BProp	{0.50; 0.65}	{0.156; 0.283}	0.890	0.8650	24

3.3 The application of bacterial memetic algorithm for optimising Mamdani type fuzzy rule bases

The bacterial approach is an evolutionary type method which performs global search, however, the solution found by the algorithm converges pretty slowly. The Levenberg-Marquardt algorithm is a gradient-based technique, which can find a more accurate solution, however, it is a local searcher, thus, this method often find only the local optimum, nearest to the initial point of state space. Combining the evolutionary and gradient-based approaches might be useful in improving the performance of the basic evolutionary algorithm, which may find the global optimum with sufficient precision in this combined way. Combinations of evolutionary and local-search methods are usually referred to as memetic algorithms [15]. I propose the combination of the bacterial approach and the Levenberg-Marquardt algorithm for optimising fuzzy rule base. I named this new combination *Bacterial Memetic Algorithm*.

Statement group 3. *I have introduced the bacterial memetic algorithm which is a novel optimisation method. I have applied the method for determining the parameters of the membership functions of Mamdani type fuzzy rule base using trapezoidal shaped membership functions and COG type defuzzification method, and I have shown by simulation experiments, that the new method thus developed was faster and more effective than the bacterial evolutionary algorithm in all of the investigated cases. Based on these results and on theoretical considerations I expect that the advantages of my proposed new method will appear in other specific applications, too.*

My publications related to this statement can be found under [B1, B3, B11, B19].

3.3.1 The proposed algorithm

I applied the Levenberg-Marquardt algorithm for each individual between the bacterial mutation and gene transfer steps. I proposed two versions of the algorithm. In the first version the size of the rule base is constant during the process. In the second version of the algorithm the length of the bacteria is changeable and can be different from each other. The goal is not only to optimise the rules but the automatic determination of the optimal size of the rule base, too. In this case the bacterial mutation and gene transfer operators can cause the change of the length of bacteria. In the bacterial mutation after the clones are created we can randomly choose from three different things in the clones. The length of the clone will either increase, decrease or remain the same. If it increases, a new rule with random parameters is added to the bacterium besides changing the parameters of the selected rule. If it will decrease, then a rule is deleted. If the length remains the same, then only the parameters are changed (this is same as in the original algorithm).

In the gene transfer we can randomly choose from two different cases: the part (rule) from the source bacterium will either overwrite a rule of the destination bacterium or will be added to the destination bacterium as a new rule. These improved bacterial mutation and gene transfer operators allow the increasing and decreasing of the length of the bacteria. In the evaluation criterion not only the approximate error calculated by the rule base must be considered, but the size of the rule base, too. More rule provides generally lower error but it increases the complexity of the model, thus the goal is to decrease the number of rules, too.

By the Bayesian Information Criterion (BIC) both goals can be achieved, namely the claim for low error and for low complexity can be described in one equation:

$$BIC = m \cdot \ln(MSE) + n \cdot \ln(m), \tag{18}$$

where m is the number of training patterns, n is the number of rules.

Statement 3.1. *I have developed a novel optimisation method named as bacterial memetic algorithm.*

3.3.2 The application of the algorithm

I applied the same reference problems as in the previous subsections and I compared the proposed algorithm with such a Bacterial Evolutionary Algorithm (BEA) which is similar to the Bacterial Memetic Algorithm (BMA) however it does not contain the Levenberg-Marquardt method. The performance specifications for the candidate with the lowest MSE value over all sessions in case of constant rule number are shown in Table 3. The parameters of the algorithms were set as follows: population size: 10, number of clones: 8, number of infections: 4, and number of rules: 3. In the Bacterial Memetic Algorithm, the Levenberg-Marquardt step runs in 10 iterations in every generation. Both algorithms were set to run for 10 sessions, the number of generations was 20 in each session. For every study case, the BMA gives the lowest MSE value fuzzy model not only for the training data but for the validation pattern set, too. Figure 2 shows MSE evolution lines for a particular simulation sessions for the six dimensional problem. The improvement caused by the Levenberg-Marquardt procedure in the memetic technique can be seen well from the simulations.

The other, more general version of the algorithm automatically sets the optimal size of the rule base, too. The performance specifications for the best candidate can be seen in Table 4. The maximum allowed bacterium length (number of rules) is 10. The number of rules can be seen in the table, too. The obtained results show the same as above, namely that the BMA provides significantly better result.

Statement 3.2. *I have developed a computer program realising the bacterial memetic algorithm, and I have applied it for optimising the rules of Mamdani type fuzzy rule base using trapezoidal shaped membership functions.*

Table 3: *Specifications for the fuzzy model with the lowest MSE value found after all sessions*

specifications	BEA	BMA	problem
MSE-training	$4.5 \cdot 10^{-5}$	$1.5 \cdot 10^{-5}$	pH
MSE-test	$7.6 \cdot 10^{-3}$	$3 \cdot 10^{-5}$	pH
MSE-training	$3.4 \cdot 10^{-1}$	$8.5 \cdot 10^{-2}$	ICT
MSE-test	$2.0 \cdot 10^{-1}$	$2.0 \cdot 10^{-2}$	ICT
MSE-training	$2.5 \cdot 10^{1}$	$2.0 \cdot 10^{1}$	6 dim.
MSE-test	$2.6 \cdot 10^{1}$	$2.3 \cdot 10^{1}$	6 dim.

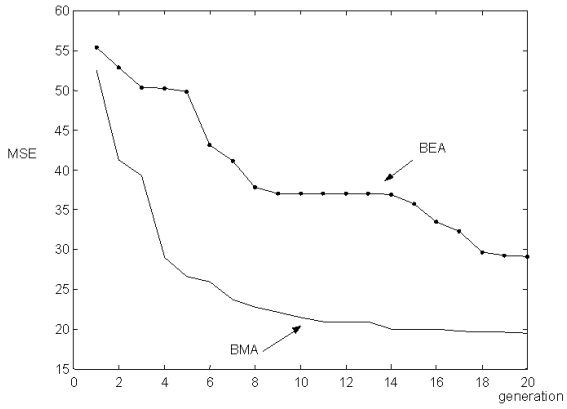

Figure 2: *MSE evolution lines for the six dimensional problem*

Table 4: *Best candidates in case of changing rule number*

specifications	pH		ICT		6 dim.	
	BEA	BMA	BEA	BMA	BEA	BMA
MSE-training	$2.73 \cdot 10^{-3}$	$4.7 \cdot 10^{-7}$	$8.42 \cdot 10^{-1}$	$1.04 \cdot 10^{-2}$	2.07	$4.10 \cdot 10^{-1}$
MSE-test	$5.12 \cdot 10^{-3}$	$6.07 \cdot 10^{-7}$	1.26	$2.60 \cdot 10^{-3}$	2.66	$9.9 \cdot 10^{-1}$
number of rules	9	8	2	5	7	10

Statement 3.3. *I have performed investigations by applying simulations for modelling reference problems obtained from the literature. I found that the bacterial memetic algorithm thus developed had given (in general much) better convergence results than the bacterial evolutionary algorithm.*

3.4 Optimising the parameters of Takagi-Sugeno type fuzzy systems

In the previous subsections I proposed different methods for identifying Mamdani type fuzzy systems. In this subsection I briefly formulate similar ideas for Takagi-Sugeno type systems.

The structure of the parameters in the rules of Takagi-Sugeno systems is different in the antecedent and in the consequent parts, thus they can be handled in different way. The structure of the antecedent part is the same as in case of Mamdani type systems, thus for the antecedent parameters the combination of the bacterial and the Levenberg-Marquardt approach can be used, and least-square techniques for the linear rule consequent parameters. The latter is applied due to the fact that it delivers a global optimum of linear parameters in the least squares sense and hence is favourable among algorithms which can be stuck in local minima.

Statement group 4. *I have proposed a new optimisation algorithm for determining the parameters of Takagi-Sugeno type fuzzy rule base using trapezoidal and Gaussian shaped mem-*

bership functions. I have shown by applying simulations that the method has more favourable properties than other techniques known from the literature in the investigated cases. Based on these results and on theoretical considerations I expect that the advantages of my proposed new method will appear in other specific applications, too.

My publication related to this statement can be found under [B10].

The goal is now to train non-linear antecedent parameters as well as linear consequent parameters in a hybrid and iterative manner, in order to gain good approximation accuracy with reasonable model complexity. In the encoding of the optimisation problem the bacterium contains only the antecedents fuzzy sets of the rules. I investigate not only trapezoidal but Gaussian shaped membership functions, too. The parameters of the rule antecedent parts are the center and widths of the Gaussian fuzzy sets in case of Gaussian shaped membership functions. The length of the individuals (the number of rules) is constant. Within the initialisation component only the antecedent parameters of the fuzzy systems are initialised, i.e. in order to start the antecedent learning scheme, the consequent parameters have to be trained for this initial setting. This is because in the antecedent learning the consequent parameters are used for the evaluation of the optimisation function. The antecedent and consequent parameters are optimised cyclically, one after the other. After the stopping criterion is fulfilled, consequent learning is carried out once more, as an eventual last change in the antecedent parts should be compensated.

The antecedent learning is done by bacterial memetic algorithm. For the Levenberg-Marquardt step in BMA the determination of the elements of the Jacobian matrix is needed, thus the partial derivatives need to be calculated for Gaussian and trapezoidal shaped membership functions.

Statement 4.1. *I have determined the partial derivatives of the output of the Takagi-Sugeno type fuzzy models using trapezoidal and Gaussian shaped membership functions according to the antecedent parameters of the model, and I have also determined the whole Jacobian matrix based on these derivatives.*

For consequent learning the Least Squares Approach [12] is applied. This is feasible as the consequent parameters are linear and therefore the mean squared error optimisation problem can be solved analytically and globally. Principally there are two possibilities for linear consequent learning: global learning, where the complete parameter vector representing all linear consequent parameters in all rules is optimised, and local learning, where each rule is treated separately. From various analytical and empirical examinations local learning turned out to be superior to global one in a lot of aspects, so I propose the local learning approach for optimising the consequent parameters.

I used the six dimensional problem for testing the algorithm. The method provided good results comparing with other approaches.

Statement 4.2. *I have developed a new computer program for separated optimisation of the antecedent and consequent parameters of Takagi-Sugeno type fuzzy systems.*

Statement 4.3. *I have performed investigations by applying simulations. I found that the new method thus developed had given better results in terms of mean square error than other techniques known from the literature.*

3.5 The application of bacterial programming for the identification of B-spline neural networks

The design process of B-spline neural network involves six design phases. The determination of the two last phases can be envisaged as a non-linear least-squares problem, and therefore completely supervised training algorithms can be employed for their determination. The Levenberg-Marquardt method is suitable for this purpose [18]. The first four phases constitute a very complex combinatorial problem. There are different constructive algorithms to help in this task. Among others the genetic programming technique was proposed [2]. In this subsection I propose a new method for B-spline neural networks design, namely the Bacterial Programming approach which is a fusion of the principles of Genetic Programming [10] and Bacterial Evolutionary Algorithm.

Statement group 5. *I have introduced a new optimisation method, namely the bacterial programming technique. I have applied the method for optimising the structure of B-spline neural networks. I found by concrete simulations that the method provided better results than other techniques known from the literature.*

My publications related to this statement can be found under [B4, B12, B13, B14].

3.5.1 The proposed method

In bacterial programming like in genetic programming, the individuals are represented by expression tree, but bacterial programming uses not the operators of genetic programming known from the literature (crossover, mutation), but the operators of bacterial evolutionary algorithm (bacterial mutation, gene transfer).

The hierarchical structure of B-spline neural network can be represented better by expression tree than by a string encoded into a chromosome. Figure 3 shows a sample expression tree for B-spline neural network. In B-splines design, sub-models must be *added* $(+)$, sub-models of higher dimensionality must be *created* from smaller sub-models $(\times)$, and sub-models of higher dimensionality must be *split* into lower dimensional sub-models $(/)$. This is the set of primitive functions that is defined for B-splines design. The node terminals do not represent only one input variable, but also the spline order, the number of interior knots, and their locations. In case of the B-spline network shown in Figure 3 for instance the left-most leaf of the tree means that this terminal node contains the input variable 3, its spline order is 1, the number of its interior knots is 2, and they are located in positions 0 and 0.4. The output of this model can be computed as the addition of three sub-models, so that:

$$y(\mathbf{X}) = f(\mathbf{X_3} \times \mathbf{X_2}) + f(\mathbf{X_1} \times \mathbf{X_2}) + f(\mathbf{X_1}),$$

where $\mathbf{X_i}$ denotes the input variable i, so the model contains two bi-dimensional and one one-dimensional sub-models.

The evolutionary process is the same as in case of bacterial evolutionary algorithm but here the individuals are trees. The initial population contains randomly created trees. The number of individuals is N_{ind}. After this there is an evolutionary cycle which applies the bacterial mutation and the gene transfer operators. The terminal criterion is usually the maximum number of generations (N_{gen}).

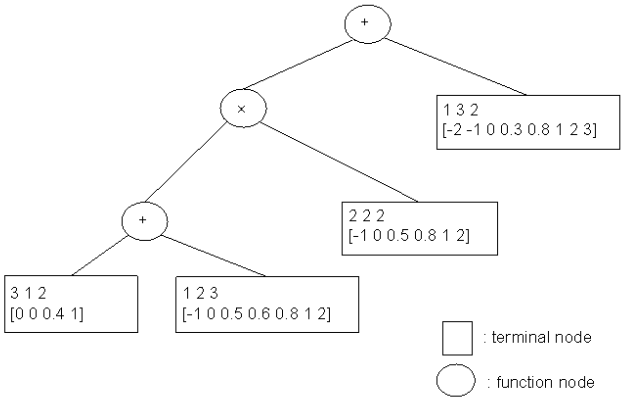

Figure 3: *A sample expression tree for B-spline networks*

The bacterial mutation is applied to each bacterium one by one. First, N_{clones} copies (clones) of the bacterium are generated. Then, a certain part of the bacterium is randomly selected and the parameters of this selected part are randomly changed in each clone (mutation). In the new method, because coding is given by an expression tree, there are two types of parts: function and terminal parts. In the function mutation at a node, the whole sub-tree beneath this node is replaced by a new randomly generated one. However, terminal mutation affects only the given node. Next, all the clones and the original bacterium are evaluated by an error criterion. The best individual transfers the mutated part into the other individuals. This cycle is repeated for the remaining parts until all of the parts of the bacterium have been mutated and tested. Mutation is applied once to the selected part, meaning that neither the selected nodes nor the sub-tree in case of function mutation, will be chosen once again for mutation, in the same generation. At the end, the best bacterium is kept and the remaining N_{clones} are discharged.

The gene transfer is similar as before. After splitting the population one bacterium is randomly chosen from the superior sub-population (source bacterium) and one from the inferior sub-population (destination bacterium). A randomly selected sub-tree of the source bacterium will overwrite a randomly selected sub-tree of the destination bacterium. There is no limitation for the size of the selected sub-tree, it can be even a sub-tree containing only one terminal node. This three step process (splitting, selection of source and destination bacterium, gene transfer) is repeated for N_{inf} times.

Bayesian Information Criterion (18) is used for evaluating the bacteria.

Statement 5.1. *I have developed the bacterial programming technique.*

Table 5: *Model structure for the lowest Bayesian Information Criterion value found after all sessions for the ICT problem*

	GP	BP
sub-models	$(1 \times 2)(1)(2)$	$(2 \times 1)(1)$
complexity	106	106
BIC	-1533.9	-2048.3
MSE	$1.3 \cdot 10^{-8}$	$8.8 \cdot 10^{-11}$

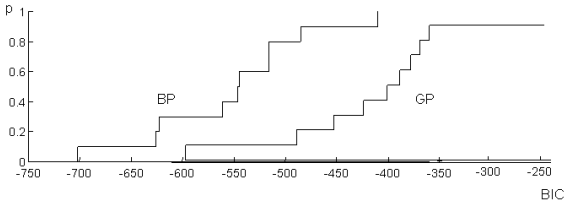

Figure 4: *Empirical probability distribution function for the generic six dimensional function*

3.5.2 The application of the method

Genetic programming solves the problem with more favourable results than the previously applied constructive algorithms. Thus, I compared the bacterial programming (BP) with the genetic programming (GP).

In the first experiments results were generated in order to decide the best values for the BP parameters. After this I compared the two approaches by applying the three reference problems previously used.

Results obtained for the ICT problem can be seen in Table 5. The table shows the result for the best individual, which has the lowest BIC value found after all sessions.

The main advantage of the bacterial approach can be deduced from the results of the six dimensional problem. These results show that the bacterial method gives good performance for problems with larger input dimension. I confirmed the obtained results by statistical analysis, too. Figure 4 shows empirical probability distribution function for the six dimensional problem, using both algorithms. While the best individuals have only BIC values with GP between about -600 and -500, they have the same values between -700 and -630 with the BP.

Statement 5.2. *I have developed a computer program realising bacterial programming technique for optimising the structure of B-spline neural network.*

Statement 5.3. *I have performed investigations by applying simulations for modelling reference problems obtained from the literature. I found that the bacterial programming technique thus developed had given better results in terms of all criteria than the genetic programming technique.*

3.6 Feature selection by bacterial evolutionary algorithms

An increase in the dimensionality of the input space increases the complexity of the learning problem. When features with only minor or no relation at all to the output space are involved, the resulting function might tend to overfit the training data. It is often efficient to reduce the number of features in advance. Feature selection can be described as the task of identifying an optimal subset of m out of the available n features.

Statement group 6. *I have applied the bacterial evolutionary algorithm for the feature selection task and I have shown by applying simulations that the algorithm combined with different regression methods is able to determine the optimal feature set of high dimensional problems.*

My publications related to this statement can be found under [B5, B15].

I propose two versions of the algorithm. In the first version the goal is to select a predefined number of features, while in the other version the number of features is free, it has to be optimised by the algorithm as well. The bacterium consists of a vector of integers, where these integers are the identifiers of the features.

The features encoded into the bacterium and the training pattern set are used to evaluate the bacteria. In the second version of the method the length of the bacterium has to be considered also in the evaluation.

The evolutionary process is same as before. However, in the bacterial mutation and gene transfer operator, when a part of a bacterium is changed (e.g. in a clone in case of the bacterial mutation or in a destination bacterium in case of the gene transfer), we must take care that the new part is unique within the selected bacterium. In the second version of the algorithm the operators were improved similarly as in the third subsection allowing the change of the length of the individuals. There are new parameters in the operators, in the bacterial mutation the length of the segment selected for mutation is also a parameter and the maximal allowed changing in the clones is a parameter, too. Similarly in the gene transfer a new parameter is the length of the segment obtained from the source bacterium, and the maximal allowed changing in the length of the destination bacterium is another new parameter, too.

Statement 6.1. *I have developed an encoding method to bacterial evolutionary algorithm applied for the feature selection task and I have improved the bacterial operators fitting to the task.*

I have performed simulations for testing the method. In the first version of the algorithm three regression problems were used. The three regression methods used for analysing the behaviour of the algorithm are as follows: the first is a simple least-square optimiser, the other two are fuzzy rule base identification models. The goal for all of the regression methods is to find a solution for three high-dimensional problems. The first test function was defined over a 20-dimensional data, the other two were defined over a 50-dimensional space. The goal was the selection of predefined number of variables. The bacterial evolutionary algorithm solved the task successfully.

After this I tested the improved version of the algorithm on an 80-dimensional function. The algorithm converged to the optimal solution in each running, and only 40 generations were needed for that. Already in the tenth generation good solutions were found.

Statement 6.2. *I have implemented a computer program for solving the feature selection task.*

Statement 6.3. *I have used the algorithm combined with three different regression methods, and I have performed simulations for different high dimensional test functions to determine the optimal variable set of the functions.*

4 Summary and outlook

The goal of the thesis was to develop identification methods based on numerical data that can produce results better in terms of quality criteria (e.g. mean square error) relevant for the given applications than other techniques known from the literature. The thesis discusses mostly the identification of fuzzy rule based systems. The aim of the identification is to create an optimal rule set based on numerical patterns. In the first five statements I dealt with such methods which create suitable fuzzy or neural model in the state space of some variables in terms of a given criterion. In the sixth statement I dealt with the preliminary step of the model identification process which goal is to find the minimal set of variables describing the given problem properly. I showed in the statements that the bacterial approach and the Levenberg-Marquardt method provide better solution for the identification, optimisation task than other methods known from the literature. I used some evaluation criteria for testing the methods. In the improved versions of the algorithms I proposed not only the optimisation of constant number of rules, or in case of the feature selection task, constant number of dimensions, but the optimisation of the size of the rule base, too. For this purpose I showed novel rule reduction operators in the first statement, and I proposed such kind of modification of the bacterial operators in the third and sixth statements, which allow changing the length of the individual in the evolutionary process.

The main goal of my further research is the identification of hierarchical fuzzy rule bases. Beside the hierarchical structure, the application of sparse rule bases is also practical, the complexity of the model can be further decreased by applying this kind of rule bases. One of the most important tasks in the further research will be the automatic identification of hierarchical interpolative fuzzy systems used in applications with high complexity. These new research directions may significantly increase the efficient applications of intelligent computational models both in the fields of scientific research and industrial utilisation.

References

[1] M. Brown and C. Harris. *Neurofuzzy Adaptive Modelling and Control*. Prentice-Hall, 1994.

[2] C. Cabrita, A. E. Ruano, and C. M. Fonseca. Single and multi-objective genetic programming design for B-spline neural networks and neuro-fuzzy systems. In *Proceedings of the IFAC Workshop on Advanced Fuzzy-Neural Control 2001, AFNC01*, pages 75–80, Valencia, Spain, Oct. 2001.

[3] R. Fletcher. *Practical Methods of Optimization*. Wiley, 2000.

[4] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, 1995.

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

[6] R. Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, 1990.

[7] J. H. Holland. *Adaption in Natural and Artificial Systems*. The MIT Press, Cambridge, Massachusetts, 1992.

[8] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic. Theory and Applications*. Prentice Hall, Upper Saddle River, New Jersey, 1995.

[9] L. T. Kóczy and D. Tikk. *Fuzzy rendszerek*. TypoTEX, Budapest, 2000.

[10] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[11] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.*, 2(2):164–168, 1944.

[12] L. Ljung. *System Identification: Theory for the User*. Prentice Hall PTR, Prentic Hall Inc., Upper Saddle River, New Jersey 07458, 1999.

[13] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Man-Mach. Stud.*, 7:1–13, 1975.

[14] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11(2):431–441, Jun. 1963.

[15] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.

[16] N. E. Nawa and T. Furuhashi. Fuzzy system parameters discovery by bacterial evolutionary algorithm. *IEEE Transactions on Fuzzy Systems*, 7(5):608–616, Oct. 1999.

[17] N. E. Nawa, T. Hashiyama, T. Furuhashi, and Y. Uchikawa. Fuzzy logic controllers generated by pseudo-bacterial genetic algorithm. In *Proceedings of the IEEE Int. Conf. Neural Networks (ICNN97)*, pages 2408–2413, Houston, 1997.

[18] A. E. Ruano, C. Cabrita, J. V. Oliveira, and L. T. Kóczy. Supervised training algorithms for B-spline neural networks and neuro-fuzzy systems. *International Journal of Systems Science*, 33(8):689–711, 2002.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[20] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Syst. Man Cybern.*, 15(1):116–132, 1985.

[21] L. A. Zadeh. Fuzzy sets. *Inf. Control*, 8:338–353, 1965.

[22] J. M. Zurada. *Introduction to Artificial Neural Systems*. West Publishing Co., St. Paul, 1992.

Publications

[B1] J. Botzheim, C. Cabrita, L. T. Kóczy, and A. E. Ruano. Fuzzy rule extraction by bacterial memetic algorithms. *International Journal of Intelligent Systems*. Accepted.

[B2] J. Botzheim, C. Cabrita, L. T. Kóczy, and A. E. Ruano. Estimating fuzzy membership functions parameters by the Levenberg-Marquardt algorithm. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2004*, pages 1667–1672, Budapest, Hungary, July 2004.

[B3] J. Botzheim, C. Cabrita, L. T. Kóczy, and A. E. Ruano. Fuzzy rule extraction by bacterial memetic algorithms. In *Proceedings of the 11th World Congress of International Fuzzy Systems Association, IFSA 2005*, pages 1563–1568, Beijing, China, July 2005.

[B4] 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*, 11(2):220–231, February 2007.

[B5] J. Botzheim, M. Drobics, and L. T. Kóczy. Feature selection using bacterial optimization. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, IPMU 2004*, pages 797–804, Perugia, Italy, July 2004.

[B6] J. Botzheim, B. Hámori, and L. T. Kóczy. Extracting trapezoidal membership functions of a fuzzy rule system by bacterial algorithm. In B. Reusch, editor, *Computational Intelligence, Theory and Applications*, volume 2206 of *Lecture Notes in Computer Science*, pages 218–227. Springer-Verlag, Berlin-Heidelberg, 2001.

[B7] J. Botzheim, B. Hámori, L. T. Kóczy, and A. E. Ruano. Bacterial algorithm applied for fuzzy rule extraction. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, IPMU 2002*, pages 1021–1026, Annecy, France, July 2002.

[B8] J. Botzheim and L. T. Kóczy. Model identification by bacterial optimization. In *Proceedings of the 5th International Symposium of Hungarian Researchers on Computational Intelligence*, pages 91–102, Budapest, Hungary, November 2004.

[B9] J. Botzheim, L. T. Kóczy, and A. E. Ruano. Extension of the Levenberg-Marquardt algorithm for the extraction of trapezoidal and general piecewise linear fuzzy rules. In *Proceedings of the 2002 IEEE World Congress on Computational Intelligence, WCCI 2002*, pages 815–819, Honolulu, Hawaii, May 2002.

[B10] J. Botzheim, E. Lughofer, E. P. Klement, L. T. Kóczy, and T. D. Gedeon. Separated antecedent and consequent learning for Takagi-Sugeno fuzzy systems. In *Proceedings of the 2006 IEEE World Congress on Computational Intelligence, WCCI 2006*, pages 10478–10484, Vancouver, Canada, July 2006.

[B11] C. Cabrita, J. Botzheim, T. D. Gedeon, A. E. Ruano, L. T. Kóczy, and C. M. Fonseca. Bacterial memetic algorithm for fuzzy rule base optimization. In *Proceedings of the World Automation Congress, WAC 2006*, Budapest, Hungary, July 2006.

[B12] C. Cabrita, J. Botzheim, A. E. Ruano, and L. T. Kóczy. Genetic programming and bacterial algorithm for neural networks and fuzzy systems design. In *Proceedings of the IFAC International Conference on Intelligent Control Systems and Signal Processing, ICONS 2003*, pages 500–505, Faro, Portugal, April 2003.

[B13] C. Cabrita, J. Botzheim, A. E. Ruano, and L. T. Kóczy. Design of B-spline neural networks using a bacterial programming approach. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2004*, pages 2313–2318, Budapest, Hungary, July 2004.

[B14] C. Cabrita, J. Botzheim, A. E. Ruano, and L. T. Kóczy. A hybrid training method for B-spline neural networks. In *Proceedings of the IEEE International Symposium on Intelligent Signal Processing, WISP 2005*, pages 165–170, Faro, Portugal, September 2005.

[B15] M. Drobics and J. Botzheim. A bacterial evolutionary algorithm for feature selection. FLLL/SCCH Master and PhD Seminar, Johannes Kepler University, Linz, Austria, June 2005.

[B16] L. T. Kóczy and J. Botzheim. Fuzzy rule base model identification techniques. In *Proceedings of the 3rd International Symposium of Hungarian Researchers on Computational Intelligence*, pages 13–24, Budapest, Hungary, November 2002.

[B17] L. T. Kóczy and J. Botzheim. Hierarchical interpolative fuzzy model identification. In *Proceedings of the 18th Hungarian-Korean Joint Seminar*, pages 17–27, Budapest, Hungary, October 2002.

[B18] L. T. Kóczy and J. Botzheim. Fuzzy models, identification and applications. In *Proceedings of the IEEE 3rd International Conference on Computational Cybernetics, ICCC 2005*, pages 13–19, Mauritius, April 2005.

[B19] L. T. Kóczy, J. Botzheim, and T. D. Gedeon. Fuzzy models and interpolation. In M. Nikravesh, J. Kacprzyk, and L. A. Zadeh, editors, *Forging New Frontiers: Fuzzy*

Pioneers I & II, volume 217 of *Studies in Fuzziness and Soft Computing*, pages 111–131. Springer, Berlin-Heidelberg, 2007.

[B20] L. T. Kóczy, J. Botzheim, A. E. Ruano, A. Chong, and T. D. Gedeon. Fuzzy rule extraction from input/output data. In P. Sincak, J. Vascak, and K. Hirota, editors, *Machine Intelligence Quo Vadis?*, volume 21 of *Advances in Fuzzy Systems - Applications and Theory*, pages 199–216. World Scientific, Singapore, 2004.