BACTERIAL MEMETIC ALGORITHM FOR FUZZY RULE BASE OPTIMIZATION

Cristiano Cabrita, Centre for Intelligent Systems, FCT, University of Algarve, Portugal, ccabrita@ualg.pt
János Botzheim, Dept. of Telecommunication and Mediainformatics, Budapest University of Technology and Economics, Hungary & Dept. of Computer Science, The Australian National University, Australia, botzheim@tmit.bme.hu
Tamás (Tom) D. Gedeon, Dept. of Computer Science, The Australian National University, Australia, tom.gedeon@anu.edu.au
António E. Ruano, Centre for Intelligent Systems, FCT, University of Algarve, Portugal, aruano@ualg.pt
László T. Kóczy, Dept. of Telecommunication and Mediainformatics, Budapest University of Technology and Economics & Széchenyi István University, Győr, Hungary, koczy@tmit.bme.hu
Carlos Fonseca, Centre for Intelligent Systems, FCT, University of Algarve, Portugal, cmfonsec@ualg.pt

ABSTRACT

In our previous works model identification methods were discussed. The bacterial evolutionary algorithm for extracting a fuzzy rule base from a training set was presented. The Levenberg-Marquardt method was also proposed for determining membership functions in fuzzy systems. The combination of evolutionary and gradient-based learning techniques – the bacterial memetic algorithm – was also introduced. In this paper an improvement of the bacterial memetic algorithm is shown for fuzzy rule extraction. The new method can optimize not only the rules, but can also find the optimal size of the rule base.

KEYWORDS: fuzzy rule base, bacterial algorithm, Levenberg-Marquardt method, memetic algorithm.

1. INTRODUCTION

A key issue in constructing fuzzy rule based models is finding good rule bases, even in circumstances where no human expert is available. The Bacterial Evolutionary Algorithm (BEA) [3, 6] is a recent approach which can be used to automatically find optimal or quasi-optimal solutions, using bacterial mutation and gene transfer operators inspired by evolutionary processes found in nature. Neural networks are often considered complementary to fuzzy systems as a number of such learning algorithms can be applied to fuzzy systems where the objective is to tune the membership functions in the fuzzy system so that the system performs a desired input to output mapping. These gradient descent / hill climbing algorithms perform local searches of the error space, and can only find local optima. They are effective in improving the performance of existing rule bases in combined with global search evolutionary algorithms, fine tuning the global optimum found. The combination of the evolutionary and the local-search methods is called a memetic algorithm [7]. In previous work, we presented a new kind of memetic algorithm, where the bacterial algorithm is hybridized with the Levenberg-Marquardt technique [2]. In this paper, the method is improved, and is also able to optimize the size of the rule base, eliminating the requirement to predefine the number of rules. Section 2 describes the structure of the fuzzy

systems used in this paper. The steps of the proposed algorithm are described in Section 3. Simulation results are shown in Section 4. Section 5 draws some conclusions.

2. FUZZY SYSTEMS

It is assumed that the model consists of a rule base:

$$R = \{R_i\} \tag{1}$$

where

$$R_i\text{: IF } (x_1 \text{ is } A_{i1}) \text{ AND} \ldots \text{AND } (x_n \text{ is } A_{in}) \text{ THEN } (y \text{ is } B_i) \tag{2}$$

R_i are the fuzzy rules, A_{ij} the fuzzy sets of the j^{th} antecedent in the i^{th} rule, B_i are the fuzzy sets of the i^{th} consequence, x_j is the input variable in the j^{th} dimension and y is the output. The fuzzy system can compute output y for an input vector x. The task is to create the most suitable fuzzy rule base for a pattern set. The class of membership functions is restricted to trapezoidal ones, as these are general enough and widely used. They are described by four parameters with the four breakpoints of the trapezoid. The membership functions are identified by indices i and j. Thus, an antecedent membership function A_{ij} belongs to the j^{th} input variable in the i^{th} rule and has four parameters: $a_{ij},b_{ij},c_{ij},d_{ij}$. A consequent membership function B_i belongs to the i^{th} rule and its parameters are: a_i,b_i,c_i,d_i. The relative importance of the j^{th} fuzzy variable in the i^{th} rule is:

$$A_{ij}(x_j) = \begin{cases} \dfrac{x_j - a_{ij}}{b_{ij} - a_{ij}}, & if \;\; a_{ij} < x_j < b_{ij} \\ 1, & if \;\; b_{ij} \le x_j < c_{ij} \\ \dfrac{d_{ij} - x_j}{d_{ij} - c_{ij}}, & if \;\; c_{ij} \le x_j < d_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $a_{ij} \le b_{ij} \le c_{ij} \le d_{ij}$ must hold.

Besides the structure of the fuzzy rule base, the inference method must also be discussed [10]. The activation degree of the i^{th} rule (the t-norm is the minimum) is:

$$w_i = \min_{j=1}^{n} A_{ij}(x_j) \tag{4}$$

where n is the number of input dimensions. w_i is the importance of the i^{th} rule if the input vector is x and $A_{i,j}(x_j)$ is the j^{th} membership function in the i^{th} rule. The i^{th} output is cut at the height w_i. The COG method is then used for defuzzification.

3. THE ALGORITHM

There are various successful evolutionary optimization algorithms known from the literature. The advantage of these algorithms is their ability to solve and quasi-optimize problems with non-linear, high-dimensional, multimodal, and discontinuous character. The original genetic algorithm (GA) was developed by Holland [4] and was based on the process of evolution of biological organisms. These processes can be easily applied in optimization problems where one individual corresponds to one solution of the problem. A more recent evolutionary technique is called the bacterial evolutionary algorithm [3, 6]. This mimics the microbial evolution phenomenon. Bacteria can transfer genes to other bacteria. This mechanism is used in the bacterial mutation and in the gene transfer operation. For the bacterial algorithm, the first step is to determine how the problem can be encoded in a bacterium (chromosome). Our 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. For example, the encoding method of a fuzzy system with two inputs and one output can be seen in Fig. 1.

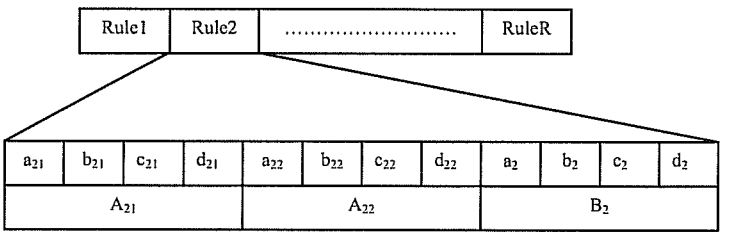

Figure 1. Encoding of the fuzzy rules

Neural networks training algorithms can also be used for fuzzy rule optimization [1]. They result in an accurate local optimum. Incorporating the neural network training algorithm with the bacterial approach, the advantages of both methods can be utilized in the optimization process. The hybridization of these two methods leads to a new kind of memetic algorithm, because the bacterial technique is used instead of the classical genetic algorithm, and the Levenberg-Marquardt as the local searcher. This method is the Bacterial Memetic Algorithm (BMA) [2]. The flowchart of the algorithm can be seen in Fig. 2.

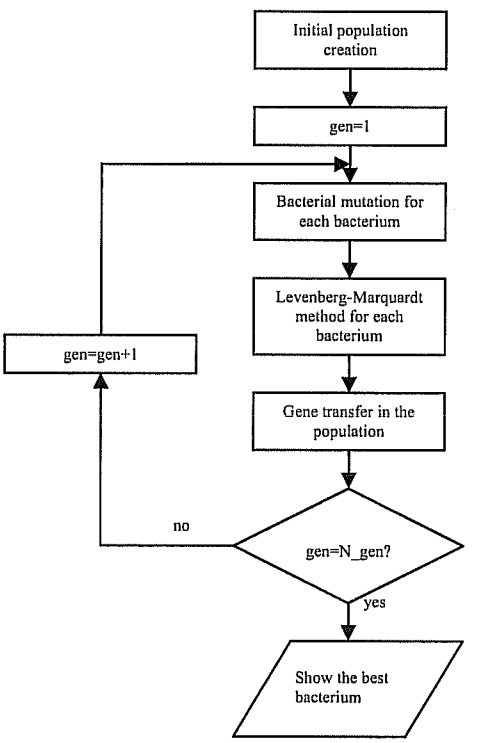

Figure 2. Flowchart of the algorithm

The difference between the BEA and the BMA is that the latter contains a local searcher step, the Levenberg-Marquardt procedure.

3.1 Initial population creation

First the initial (random) bacteria population is created. The population consists of N_{ind} chromosomes (bacteria). This means that all membership functions in the chromosomes must be randomly initialized. The number of rules in the chromosomes can be different. The length of the chromosome is a random value.

3.2 Bacterial mutation

The bacterial mutation is applied to each chromosome one by one. First, N_{clones} clones of the rule base are generated. Then a part (rule) of the chromosome is randomly selected and the parameters of this selected part are randomly changed in each clone (mutation). Because we also want to find the optimal number of rules, we must allow changes to the length of the bacterium during this operation. So, we use a random value, which determines that after the mutation the length will increase, decrease or remain the same. If it will increase, then besides changing the parameters of the selected rule, we add a new rule to the bacterium, with random parameters. If it will decrease, then a rule is deleted. If the length remains the same, then only the parameters are changed. Next all the clones and the original bacterium are evaluated by an error criterion. The error criterion must contain two important components. How accurate the rule base is, and how large its complexity. The best individual transfers its mutated part into other individuals. This cycle is repeated for the remaining parts, until all parts of the chromosome have been mutated and tested. At the end the best rule base is kept and the remaining N_{clones} are eliminated.

3.3 Levenberg-Marquardt method

After the bacterial mutation step, the Levenberg-Marquardt algorithm [5] is applied for each bacterium. In this step a minimization criterion has to be employed also, which is related to the quality of the fitting. The training criterion that will be employed is the usual Sum-of-Square-of-Errors (SSE):

$$\Omega = \frac{\left\| \underline{t} - \underline{y} \right\|^2}{2} = \frac{\left\| \underline{e}[k] \right\|^2}{2} \tag{5}$$

where t stands for the target vector, y for the output vector, e for the error vector, and $\| \ \|$ denotes the 2-norm. It will be assumed that there are m patterns in the training set.

The most commonly used method to minimize Eq. (5) is the Error-Back-Propagation (BP) [9] algorithm, which is a steepest descent algorithm. The BP algorithm is a first-order method as it only uses derivatives of the first order. If no line-search is used, then it has no guarantee of convergence and the convergence rate obtained is usually very slow. If a second-order method is to be employed, the best to use is the Levenberg-Marquardt (LM) algorithm [5], which explicitly exploits the underlying structure (sum-of-squares) of the optimization problem on hand.

Denoting by J the Jacobian matrix :

$$\underline{\underline{J}}[k] = \left[\frac{\partial y(\underline{x}^{(p)})[k]}{\partial \underline{par}[k]} \right] \tag{6}$$

where the vector $\underline{par}$ contains all membership functions' parameters (all breakpoints in the membership functions), and k is the iteration variable. The new parameter values can be obtained by the update rule of the LM method:

$$\underline{par}[k+1] = \underline{par}[k] - \left[\underline{\underline{J}}^T[k] \underline{\underline{J}}[k] + \alpha \underline{\underline{I}} \right]^{-1} \underline{\underline{J}}^T[k] \underline{e}[k] \tag{7}$$

In Eq. (7), α is a regularization parameter, which controls the 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. This is dependent on how well the actual criterion agrees with a quadratic function, in a particular neighborhood. The good results presented by the LM method (compared with other second-order methods such as the quasi-Newton and conjugate gradient methods) are due to the explicit exploitation of the underlying characteristics of the optimization problem (a sum-of-square of errors) by the training algorithm. The Jacobian matrix with respect to the parameters in the bacterium must be computed. This can be done on a pattern by pattern basis. The computation of the Jacobian matrix is described in [1].

3.4 Gene transfer

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 (rule) from the source bacterium is chosen and this part will either overwrite a rule of the destination bacterium or will be added to the destination bacterium as a new rule. Between the overwriting and the addition we choose randomly. This cycle is repeated for N_{inf} times, where N_{inf} is the number of "infections" per generation.

3.5 Stop condition

If the population satisfies a stop condition or the maximum number of generation N_{gen} is reached then the algorithm ends, otherwise it returns to the bacterial mutation step.

4. SIMULATION RESULTS

In this work three examples have been used to test the algorithm. We compared the BMA with the BEA. We used a one-dimensional problem (pH problem), a two-dimensional one (ICT), and a six-dimensional benchmark problem. For the complete description of this problems please refer to [3, 8]. Both algorithms were set to run for 10 sessions of 20 generations each. The training and validation sets contained the same number of patterns and some of the patterns were identical (for the ICT problem only). 101 patterns were used for the pH problem, 110 patterns for the ICT problem and 200 patterns for the six-dimensional function. The parameters of the algorithms were set as follows: population size: 10, number of clones: 8, number of infections: 4. In the BMA, the Levenberg-Marquardt step runs for 10 iterations in every generation and the maximum allowed bacterium length is 10.

For the evaluation of the bacteria, the BIC criterion is used in the following form:

$$BIC=m*\ln(MSE)+n*\ln(m) \quad (8)$$

where m is the number of patterns in the training set, n is the number of rules, MSE is the Mean-Square-Error. The mean values obtained by the algorithm can be seen in Table 1.

Specification	pH		ICT		Six-dimensional	
	BEA	BMA	BEA	BMA	BEA	BMA
MSE	$6{,}28\text{x}10^{-3}$	$2{,}3\text{x}10^{-6}$	$8{,}69\text{x}10^{-1}$	$3{,}67\text{x}10^{-2}$	3.21	1.29
MSRE	$4{,}06\text{x}10^{4}$	$1{,}02\text{x}10^{1}$	$5{,}63\text{x}10^{13}$	$1{,}80\text{x}10^{13}$	$6{,}20\text{x}10^{-2}$	$3{,}07\text{x}10^{-2}$
MREP	$2{,}05\text{x}10^{3}$	$2{,}69\text{x}10^{1}$	$1{,}77\text{x}10^{8}$	$1{,}08\text{x}10^{8}$	$1{,}86\text{x}10^{1}$	$1{,}21\text{x}10^{1}$
MSEv	$7{,}76\text{x}10^{-3}$	$2{,}08\text{x}10^{-6}$	1,30	$1{,}12\text{x}10^{-2}$	4,29	2,22
MSREv	$1{,}63\text{x}10^{5}$	$4{,}18\text{x}10^{1}$	$1{,}92\text{x}10^{-1}$	$1{,}62\text{x}10^{-3}$	$6{,}50\text{x}10^{-2}$	$3{,}37\text{x}10^{-2}$
MREPv	$4{,}12\text{x}10^{3}$	$5{,}46\text{x}10^{1}$	$2{,}56\text{x}10^{1}$	3,04	$1{,}91\text{x}10^{1}$	$1{,}32\text{x}10^{1}$
#rules	4,90	7,40	2,20	5,00	6,50	7,30

Table 1. Mean values using BEA and BMA.

MSRE means the Mean Square of the Relative Error, MREP means the Mean Relative Error Percentage, and v subscript is used for the validation data. It can be seen from the table that the BMA found the lowest value for every problem, and for every error definition. Besides the mean values it is also important to compare the best models given by the two algorithms. The performance specifications for the best candidate can be seen in Table 2. For every study case, the bacterial memetic algorithm gives the lowest MSE valued fuzzy model. These models also represent the best validation specifications. Using the Levenberg-Marquardt method as local searcher, the performance of the bacterial algorithm can be improved.

Specification	pH		ICT		Six-dimensional	
	BEA	BMA	BEA	BMA	BEA	BMA
MSE	$2{,}73\text{x}10^{-3}$	$4{,}7\text{x}10^{-7}$	$8{,}42\text{x}10^{-1}$	$1{,}04\text{x}10^{-2}$	2.07	$4{,}10\text{x}10^{-1}$
MSRE	$3{,}71\text{x}10^{4}$	3,63	$5{,}32\text{x}10^{13}$	$7{,}09\text{x}10^{12}$	$4{,}78\text{x}10^{-2}$	$9{,}56\text{x}10^{-3}$
MREP	$1{,}97\text{x}10^{3}$	$1{,}92\text{x}10^{1}$	$2{,}03\text{x}10^{8}$	$6{,}39\text{x}10^{7}$	$1{,}59\text{x}10^{1}$	7,06
MSEv	$5{,}12\text{x}10^{-3}$	$6{,}07\text{x}10^{-7}$	1,26	$2{,}60\text{x}10^{-3}$	2,66	$9{,}9\text{x}10^{-1}$
MSREv	$1{,}48\text{x}10^{5}$	$1{,}53\text{x}10^{1}$	$1{,}87\text{x}10^{-1}$	$3{,}74\text{x}10^{-4}$	$4{,}28\text{x}10^{-2}$	$1{,}5\text{x}10^{-2}$
MREPv	$3{,}96\text{x}10^{3}$	$3{,}93\text{x}10^{1}$	$2{,}34\text{x}10^{1}$	1,48	$1{,}47\text{x}10^{1}$	9,28
#rules	9	8	2	5	7	10

Table 2. Best candidates using BEA and BMA.

5. CONCLUSIONS

An improved version of the bacterial memetic algorithm was introduced in this paper. The algorithm is a combination of the bacterial evolutionary algorithm and the Levenberg-Marquardt method. This approach gives very good results in optimization of fuzzy rule base. It shows significant improvement on the bacterial evolutionary algorithm. With the improved bacterial operators, we need not to predefine the number of rules.

6. ACKNOWLEDGEMENTS

Research supported by the Australian Research Council, the Széchenyi University Main Research Direction Grant 2005, a National Scientific Research Fund Grant OTKA T048832 and the Hungarian-Portuguese Bilateral Intergovernmental S&T Cooperation Grant P-21/03.

7. REFERENCES

[1] J. Botzheim, C. Cabrita, L. T. Kóczy, and A. E. Ruano, "Estimating Fuzzy Membership Functions Parameters by the Levenberg-Marquardt Algorithm," FUZZ-IEEE 2004, Budapest, Hungary, 2004, pp. 1667-1672.

[2] J. Botzheim, C. Cabrita, L. T. Kóczy, and A. E. Ruano, "Fuzzy rule extraction by bacterial memetic algorithm," IFSA 2005, Beijing, China, 2005, pp.1563-1568.

[3] J. Botzheim, B. Hámori, L. T. Kóczy, and A. E. Ruano, "Bacterial algorithm applied for fuzzy rule extraction," IPMU 2002, Annecy, France, 2002, pp. 1021-1026.

[4] J. H. Holland, Adaptation in Nature and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, The MIT Press, Cambridge, MA, 1992.

[5] D. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," SIAM J. Appl. Math., 11, 1963, pp. 431-441.

[6] N.E. Nawa and T. Furuhashi, "Fuzzy Systems Parameters Discovery by Bacterial Evolutionary Algorithms," IEEE Tr. Fuzzy Systems 7 (1999), pp. 608-616.

[7] Y. S Ong and A. J. Keane, "Meta-Lamarckian Learning in Memetic Algorithms," IEEE Trans. on Evolutionary Computation, Vol 8. No. 2. 2004, pp. 99-110.

[8] A. E. Ruano, C. Cabrita, J. V. Oliveira, L. T. Kóczy, and D. Tikk, "Supervised Training Algorithms for B-Spline Neural Networks and Fuzzy Systems," Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, Canada, 2001.

[9] P. Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, PhD. Dissertation, Appl. Math., Harvard University, USA, 1974.

[10] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," IEEE Tr. Systems, Man and Cybernetics 3 (1973), pp. 28-44.