Applying Bacterial Memetic Algorithm for Training Feedforward and Fuzzy Flip-Flop based Neural Networks

László Gál[1,2] János Botzheim[3,4] László T. Kóczy[1,4] Antonio E. Ruano[5]

1 Institute of Information Technology and Electrical Engineering,
Széchenyi István University, Győr, Hungary
2 Department of Technology, Informatics and Economy,
University of West Hungary Szombathely, Hungary
3 Department of Automation Széchenyi István University Győr, Hungary
4 Department of Telecommunication and Media Informatics
Budapest University of Technology and Economics, Budapest, Hungary
5 Centre for Intelligent Systems, FCT, University of Algarve, Portugal
Email: gallaci@ttmk.nyme.hu, {botzheim, koczy}@{sze, tmit.bme}.hu, aruano@ualg.pt

Abstract*—In our previous work we proposed some extensions of the Levenberg-Marquardt algorithm; the Bacterial Memetic Algorithm and the Bacterial Memetic Algorithm with Modified Operator Execution Order for fuzzy rule base extraction from input-output data. Furthermore, we have investigated fuzzy flip-flop based feedforward neural networks. In this paper we introduce the adaptation of the Bacterial Memetic Algorithm with Modified Operator Execution Order for training feedforward and fuzzy flip-flop based neural networks. We found that training these types of neural networks with the adaptation of the method we had used to train fuzzy rule bases had advantages over the conventional earlier methods.*

Keywords— Bacterial Memetic Algorithm, Fuzzy Flip-Flop, Levenberg-Marquardt method, Neural Network.

1 Introduction

Bacterial type evolutionary algorithms are inspired by the biological bacterial cell model [1,2]. The *Bacterial Memetic Algorithm* (BMA) is a recent method for fuzzy rule base extraction from input-output data for a certain system [7]. We have investigated its properties intensely and found some points where its performance in the fuzzy rule base identification could be improved. The recent bacterial type algorithms we proposed were named *Bacterial Memetic Algorithm with Modified Operator Execution Order* (BMAM), *Improved Bacterial Memetic Algorithm* (IBMA) and *Modified Bacterial Memetic Algorithm* (MBMA) [3,4]. They are both memetic algorithms and apply alternatively global and local search for identifying fuzzy rule bases from input-output data automatically when no human expert to define the rules is available.

Neural Networks belong to the Soft Computing area like Fuzzy Systems and Evolutionary Computing. They can be used for modeling a certain system where input-output data pairs exist. The neural networks are inspired by biological phenomena: the brain itself and other parts of the neural system.

Fuzzy Flip-Flops are extended forms of the binary flip-flops that are widely used in digital technics [5]. They use fuzzy logic operations instead of Boolean logic ones and require fuzzy inputs, furthermore they produce fuzzy outputs instead of digital values.

Our previous works were developing the *Bacterial Memetic Algorithm* applied for *fuzzy rule base identification* (FRBI) and investigating various types of *Fuzzy Flip-Flops* (F^3) used in *feedforward neural networks* (FFNN) as replacements of the neurons [6]. We trained the *Fuzzy Flip-Flop based Neural Networks* (FNN) with the *Levenberg-Marquardt* (LM) based training method as it is a widely used and accepted one. However, we faced the same problems with the LM based feedforward neural network training as in the fuzzy rule base identification. Therefore we have adopted the *Bacterial Memetic Algorithm with Modified Operator Execution Order* (BMAM) for training Neural Networks. Our goal was to improve the learning capabilities of feedforward neural networks with a bacterial type evolutionary approach.

In this paper we propose the adaptation of the BMAM for training feedforward neural networks, and we study and evaluate the respective results. From another aspect another paper was proposed here where we report on the findings of our investigations of the properties of different types of FNNs trained with BMAM [14].

2 Bacterial Memetic Algorithm with Modified Operator Execution Order (BMAM)

2.1 Bacterial Memetic Algorithm (BMA)

Bacterial Memetic Algorithm (BMA) is a very recent approach used for fuzzy rule base identification (FRBI) [7]. It combines global and local search. For the global search it uses bacterial type evolutionary approach and for the local search the *Levenberg-Marquardt* method is deployed. Previous work confirmed that the *Pseudo-Bacterial Genetic Algorithm* (PBGA) and the *Bacterial Evolutionary Algorithm* (BEA) were rather more successful in this area than the conventional genetic algorithms [1,2].

2.1.1 Bacterial mutation

PBGA is a special kind of *Genetic Algorithm* (GA) [8], it introduces a new "genetic" operation called *bacterial mutation*. For the algorithm, the first step is to determine how the problem can be encoded in a *bacterium* (*chromosome*). In case of modelling fuzzy systems the parameters of the model – all the breakpoints of the rule base – have to be encoded in the *chromosome*. The next step is to generate initial *bacteria* randomly. Then an optimization process is started utilizing mainly the *bacterial mutation*, in order to refine the model parameters.

The *bacterial mutation* operation tries to improve the parts of the *chromosomes*. Therefore each individual (*bacterium)* is selected one by one and a number of copies of the selected individual (*clones*) are created. Then the same part or parts are randomly chosen from all clones and it (they) is (are) mutated (except one single clone that remains unchanged). Mutation means to replace the part with a random value in a specified range.

After the mutations all the clones are evaluated (SSE, MSE, BIC) and the best clone is selected whose mutated part or parts are transferred to the other clones. Theoretically, this operation copies just a few parameters from one clone to the other clones (*gene transfer*), but in practice, the other clones will not differ from the best clone at the end at all. So, this operation can be done with discharging all the clones except the best one and then cloning further the best clone.

After the selection of the best clone and transferring its mutated part or parts to the other clones the above procedure is repeated until all the parts are mutated exactly once. The final best clone is remaining in the population and all the other clones are destroyed. The *bacterial mutation* cycle is done on the other individuals in the population e.g. in a parallel processing way.

At the end of the *complete bacterial mutation cycle* a new generation of bacteria is arisen.

The *Bacterial Evolutionary Algorithm* (BEA) is based on the PBGA supported by a new genetic operation called *gene transfer* operation. This operation can play an important role in the FRBI process as it establishes relationships among the individuals of the population (useful in somewhat changing environment) and is able to increase or decrease the number of the rules in a fuzzy rule base (useful in determining the appropriate size of the fuzzy rule base). Because this behaviour is not exploited in our investigations when training neural networks this operation is not described in detail.

2.1.2 The Levenberg-Marquardt method (LM)

The *Levenberg-Marquardt* (LM) method [9] is a gradient based iterative procedure. It is used for least squares curve fitting for a given set of empirical data, minimizing the *sum of squared error* function (SSE). It can be used for fuzzy rule extraction alone, but it generates only locally optimal rule base in the neighbourhood of the initial rules.

The *Error Back Propagation* algorithm (EBP or BP) was a great improvement in neural network research, but it has weak convergence rate. The LM algorithm is more complex and requires more computational effort than the BP, but it has much better convergence rate properties. Therefore the LM algorithm is one of the most popular training functions for feedforward back propagation networks.

2.1.3 Bacterial Memetic Algorithm (BMA)

Memetic algorithms combine evolutionary and local search methods [10]. The evolutionary part is able to find the global optimum region, but is not suitable to find the accurate minimum in practice. The gradient based part is able to reach the accurate optimum, but is very sensitive to the initial position in the search space and is unable to avoid the local optimum. Combining global and local search is expected to be beneficial.

Bacterial Memetic Algorithm (BMA) combines the *Bacterial Evolutionary Algorithm* (BEA) and the *Levenberg-Marquardt* (LM) method. In the past we used it for fuzzy rule base extraction, among others.

The BMA integrates its two components, the BEA and the LM method in the following way:

1. *Bacterial Mutation* operation for each individual,
2. *Levenberg-Marquardt* method for each individual,
3. *Gene Transfer* operation for a partial population.

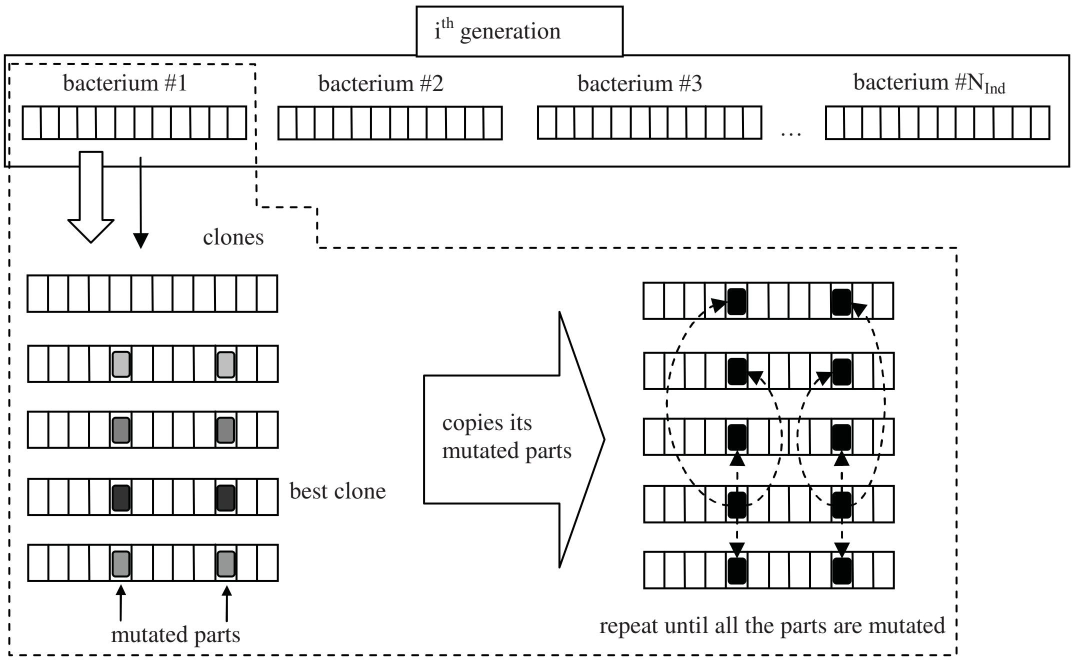

Figure 1: Bacterial mutation (one individual)

ISBN: 978-989-95079-6-8

This way the LM method is nested into the BEA, so that local search is done for every global search cycle.

2.2 Bacterial Memetic Algorithm with Modified Operator Execution Order (BMAM)

Although BMA provides a very good speed of convegrenve towards the optimal model parameters there are some points of the algorithm where the performance could be increased. We proposed new techniques to improve its performance. Some of them contain modifications that are not useful in training FFNNs (handling the *knot order violation* that can occur in applying LM for FRBI) (IBMA, MBMA) [3]. Another improvement to BMA is the *Bacterial Memetic Algorithm with Modified Operator Execution Order* [4] which exploits the *Levenberg-Marquardt* method more efficiently.

Instead of applying the LM cycle *after* the *bacterial mutation* as a separate step, the modified algorithm executes several LM cycles *during* the *bacterial mutation* after *each mutational* step.

The *bacterial mutation* operation changes one or more parameters of the modeled system randomly, and then it checks whether the model obtained by this way performs better than the previous models or the models that have been changed concurrently this way in the other clones. The mutation test cycle is repeated until all the parameters of the model have gone through the bacterial mutation.

In the mutational cycle it is possible to gain a temporary model that has an instantaneous fitness value that is worse than the one in the previous or the concurrent models. However, it is potentially better than those, because it is located in such a region of the search space that has a better local optimum than the other models do. In accordance to this, if some Levenberg-Marquardt iterations are executed after *each* bacterial mutational step, the test step is able to choose some potentially valued clones that could be lost otherwise.

In the *Bacterial Memetic Algorithm with Modified Operator Execution Order*, after *each mutational step* of *every single bacterial mutation iteration* several LM iterations are done. Several tests have shown it is enough to run just 3 to 5 of LM iterations per mutation to improve the performance of the whole algorithm. The usual test phase of the *bacterial mutation* operation follows after the LM iterations. After the complete *modified bacterial mutation* follows the LM method that is used in the original BMA, where more, e.g. 10 iterational steps, are done with all the individuals of the population towards reaching the local optimum. After all this the *gene transfer* operation is executed if needed.

3 Fuzzy flip-flops (F^3)

Fuzzy flip-flops are extended forms of binary flip-flops used in the conventional digital technics. We have dealt with the fuzzy extensions (complements) of the binary J-K and D flip-flops. Various types of fuzzy flip-flops are implemented and tested (set, reset type and the general type using the unified equation; J-K, D and Choi type D; based on minmax, algebraic, Yager, Dombi, Hamacher and Frank t-norms and co-norms, resp.) [11]. Because of an interesting property some fuzzy flip-flops can be used for implementing a sigmoid like transfer function and so constructing Multilayer Perceptron Neural Networks.

In our previous works we studied the behavior of various type fuzzy flip-flops, illustrating their characteristics by their respective graphs. We proposed also the concept of *fuzzy flip-flop based neural networks* and investigated their function approximation capabilities [6, 12].

4 Fuzzy flip-flop based feedforward neural networks (FNN)

In our team extensive research was done with the leadership of R. Lovassy in the field of fuzzy flip-flops. As we mentioned it before, various fuzzy norms can be used for building *fuzzy flip-flop based neural networks* (FNNs). The basic idea was to substitute the fuzzy flip-flops with sigmoidal transfer function instead of traditional neurons. The flip-flops are based on various norms, consequently, their transfer functions have different slopes. Fixing the value of the present state Q (in the characteristical equation), often we obtained “good” enough sigmoidal transfer function character [6]. First of all, to train this kind of neural network with a usual training method BP or LM the derivatives of these transfer functions have to be also calculated. Then the FNN can be used and trained in the usual way. We found that the FNNs we created had good approximation properties. [12].

5 Training feedforward neural networks

With an appropriate transfer function and its derivative the *Error Back Propagation* algorithm (BP) can be used for training feedforward neural networks (FFNN). However, it has weak convergence rates.

The LM algorithm is more complex and requires more computational effort than the BP one, but it has much better convergence rates. The LM algorithm is one of the most popular training methods for feedforward neural networks despite of its higher memory requirements and higher complexity.

The training of the FFNNs begins with the random generation of initial weights and biases. Then the training method selected is applied. An update vector is generated that has to be applied for the vector that contains the weights and biases.

When using BP or LM based training methods one faces the drawback of these local searchers described in the next section.

6 Using BMAM in training FFNNs

Although the LM method based training of the neural networks works much more efficiently than the BP based one it has all drawbacks of the local search methods. The training is very sensitive to the (parameter’s) initial position of the search space. An inconveniently generated random parameter set with the initial weights and biases determines a hardly trainable neural network with a weak performance at the end of the LM method based training procedure. This is because the LM method is a local searcher and thus it is unable to avoid the local minima.

We decided to apply bacterial type evolutionary algorithms because they proved to be rather successful in our previous

ISBN: 978-989-95079-6-8

works, better than the other evolutionary approaches. We preferred the *Bacterial Memetic Algorithm with Modified Operator Execution Order* because it converged faster than the original BMA. (And contained not only FRBI related improvements, like IBMA and MBMA do.)
We did not implement here the *gene transfer* operation because it was not useful with the neural network training we have done (in not changing environments, there was no need to change the structure of the NN or the number of the neurons).
The detailed steps of the BMAM used for NN training are described below:

1. Create the initial population – neural networks with two hidden layers – and initialize the neural network's input, layer weights and biases randomly as before a usual LM training procedure. Each individual contains the weights and biases – the parameters of the model – encoded in the chromosome. In a 1-4-3-1 NN the number of the parameters to be encoded are 2*4+4*3+2*3+1 = 27 parameters per individual.
2. Apply the *modified bacterial mutation* for each individual.
    a. Each individual is selected one by one.
    b. N_{Clones} copies of the selected individual are created ("*clones*").
    c. Choose the same part or parts randomly from the clones and mutate it (except one single clone that remains unchanged during this mutation cycle).
    d. Run some conventional LM method based NN training iterations (3-5 epochs).
    e. Select the best clone (simulate and evaluate the NNs) and transfer *all* of its parts to the other clones.
    f. Repeat the part selection-mutation-*LM*-selection-transfer cycle until all the parts are mutated, improved and tested.
    g. The best individual is remaining in the population, all other clones are deleted.
    h. This process is repeated until all the individuals have gone through the *modified bacterial mutation*.
3. Apply the LM method based NN training to each individual (e.g. 10 epochs per individual per generation)
4. Repeat the procedure above from the *modified bacterial mutation* step until a certain termination criterion is satisfied (e.g. maximum number of generations = 20 generation).

The experimental setup was:

- General PC (2GHz), Windows XP, Matlab
- Test function: $f(x)=\dfrac{\sin(c_1\cdot x)\cdot\sin(c_2\cdot x)}{c_3}+c_4$
- $c_1=0.2, c_2=0.07, c_3=2, c_4=0.5$
- Number of individuals in the population: 5
- Number of clones: 5.

We tested the new training algorithm in two ways.

6.1 *Test 1*

In the first test group we applied the new BMAM based NN training. We created a 1-4-3-1 feedforward neural network with the usual sigmoid transfer function and with selected fuzzy flip-flop based neurons. Our goal was to investigate the improvement of BMAM based training over the conventional LM based one so four transfer functions were selected: sigmoid (tansig), Dombi Fuzzy D Flip-Flop (Dombi DF^3), Frank Fuzzy D Flip-Flop (Frank DF^3) and Frank Choi-type Fuzzy D Flip-Flop (Frank CDF^3).
Our goal was here to train a NN that is hard to be trained [12]. The number of neurons (4 and 3 in the hidden layers) was relatively low. It makes possible to recognise the performance improvement (MSE) better, to see that the model complexity may be reduced with the better training, and to avoid overfitting.
We ran 30-30 trainings for each case mentioned above. Then the maximum, minimum, median and mean MSE values calculated.
In our previous work we chose the median to characterize the trainability of the FFF based NNs because in case of training with the LM based way there were several unsuccessful trainings where the final model was unusable and produced too high MSE. That is why we had to analyse 30-100 runs. The best value (minimum MSE) was more or less randomly good so it could not be used as a reliable value for indicate the trainability. In case of the mean value a single one unsuccessful training deteriorates many very successful training results. With using the median this random extreme values could be avoided.
Table 1 and 2 contains the results of these tests. One can see that using BMAM result much more better quality models (lower MSE). Using BMAM results lover maximum MSE values than median or mean values of the LM based training respectively. Furthermore the median and mean MSE values of the BMAM based training are very close to the minimum values of the LM based training respectively.

Table 1: MSE values of LM based training

LM based	Max	Min	Median	Mean
Tansig	0.06452	$1.2\text{x}10^{-5}$	0.00712	0.01891
Dombi DF^3	0.11962	0.04263	0.05732	0.06045
Frank DF^3	0.06644	0.00459	0.04642	0.04159
Frank CDF^3	0.06645	0.00593	0.05486	0.04697

Table 2: MSE values of BMAM based training

BMAM	Max	Min	Median	Mean
Tansig	0.00180	$3.14\text{x}10^{-7}$	$2.38\text{x}10^{-5}$	0.00034
Dombi DF^3	0.03994	0.02342	0.03362	0.03294
Frank DF^3	0.01091	0.00187	0.00680	0.00716
Frank CDF^3	0.02519	0.00460	0.00833	0.00933

Figure 2 to 9 show MSE histograms of 30-30 runs with various transfer functions and training methods. One can see that if using BMAM there is no need to run 30 or 100 complete training cycles to gain an excellent quality model because every BMAM trained model have very low MSE value.

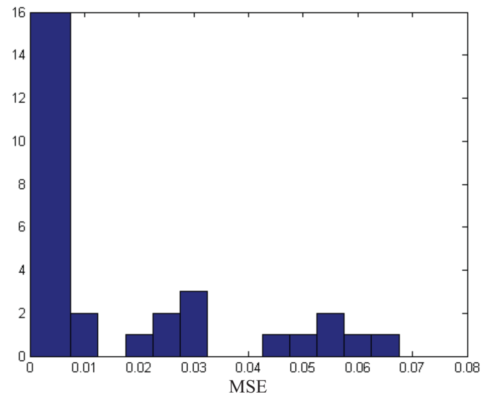

Figure 2: LM – Tansig NN histogram

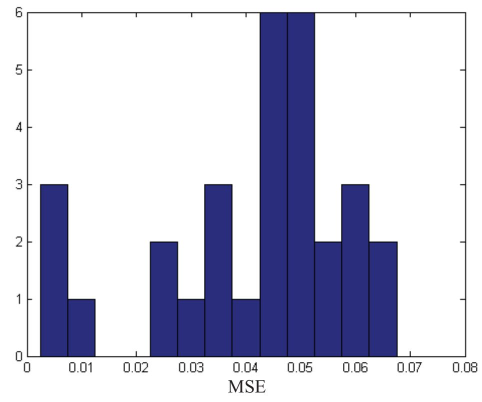

Figure 6: LM – Frank DF^3 NN histogram

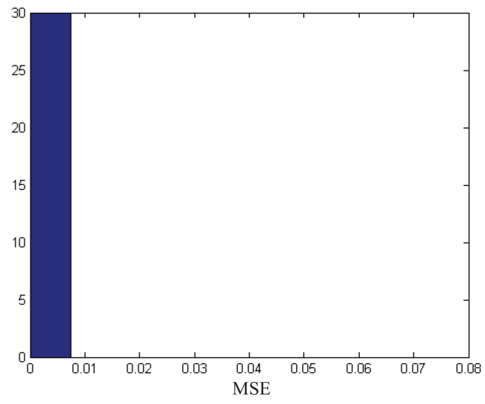

Figure 3: BMAM – Tansig NN histogram

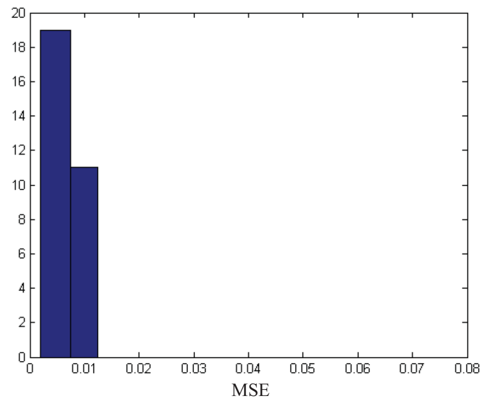

Figure 7: LM – Frank DF^3 histogram

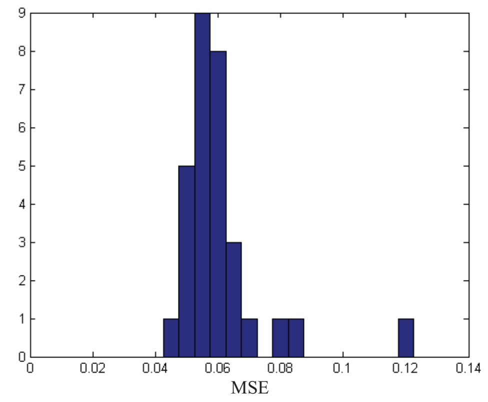

Figure 4: LM – Dombi DF^3 histogram

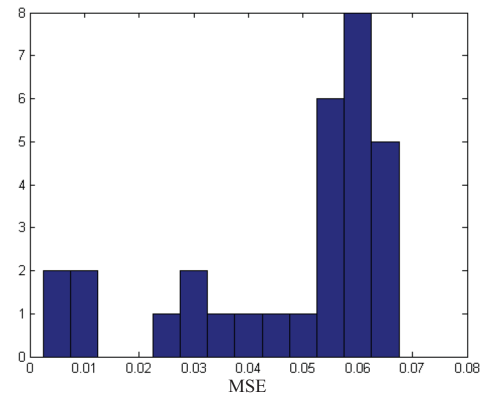

Figure 8: LM – Frank CDF^3 NN histogram

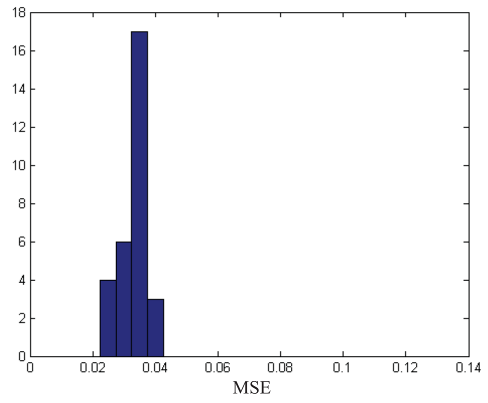

Figure 5: BMAM – Dombi DF^3 NN histogram

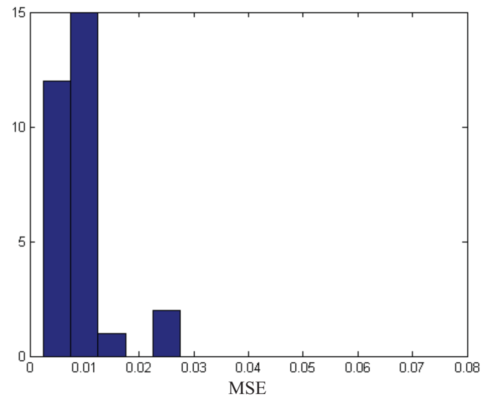

Figure 9: BMAM – Frank CDF^3 NN histogram

ISBN: 978-989-95079-6-8

6.2 Test 2

In the second group of tests we utilized the BMAM to identify quasi optimal parameter values of various types of fuzzy flip-flops used in FNNs. 1-8-8-1 neural networks were created because the good trainability was much more important than the lower complexity of the model here. This way the optimal parameter values are easier to identify.

Therefore we enhanced the capability of the BMAM based training method in a manner that the parameter (internally fixed Q values) of the fuzzy flip-flop used in Fuzzy Flip-Flop Neural Networks (FNN) can be encoded into the *chromosome*. This way it participates in the *bacterial mutation* cycle so the quasi optimal value of this parameter can be identified at the end of the BMAM based training.

Because this parameter is not affected by the LM training we applied two different versions of the *bacterial mutation* especially for this parameter. The first one is the original *bacterial mutation* (generate random values in the range of [0, 1]), while the second one increments or decrements the current fixed Q value with a very fine random step.

Table 2 shows the expected ranges and the quasi-optimal internally fixed Q values of several FNNs identified by the BMAM training method. The expected ranges were derived from our previous work [12].

Table 2: Expected ranges and fixed Q values by BMAM

Type of FNN	Expected range	Fixed Q value identified by BMAM
Algebraic JK FF	0 – 0.4	0.25
Algebraic D FF	~0.1,~0.5, ~0.9	0.91
Algebraic C D FF	<0.15, 0.4 – 0.6, >0.85	0.53
Dombi D FF	<0.1 or >0.9	0.924
Frank D FF	0.25 – 0.45	0.31

Further investigations will be focused on using BMAM based training method to identify of the other variable parameters of Yager, Dombi, Hamacher, Frank norms based FNNs.

7 Conclusions

In this paper we introduced the adaptation of *the Bacterial Memetic Algorithm with Modified Operator Execution Order* for *training feedforward neural networks*, especially *neural networks* built from *Fuzzy Flip-Fops* (F^3s).

We applied this new approach to training neural networks and *fuzzy flip-flop based neural networks*. Our goal was to get a quasi-optimal result with only a single one or a very low number of training sequences whose error does not exceed (or very rarely exceeds) an acceptable level. Despite the usual tradeoffs between the complexity and accuracy [13] this way there is no need to run a few hundred of training cycles to get an acceptable model.

Our tests have shown that BMAM used for training FFNNs and fuzzy flip-flop based FFNNs is a very successful tool. Although it requires more computational effort than the conventional training methods it produces a higher quality model (so the complexity of the model can be reduced) with only one training cycle.

Furthermore we enhanced the capability of the BMAM based training method in a manner that the parameter or parameters of the fuzzy flip-flop used in *Fuzzy Flip-Flop based Neural Networks* (FNN) can be encoded into the chromosome. This way it participates in the *bacterial mutation* cycle so the quasi optimal values of these parameters can be identified at the end of the BMAM based training.

Acknowledgment

This paper was supported by the Széchenyi University Main Research Direction Grant 2009, National Scientific Research Fund Grant OTKA T048832 and K75711, SEK Scientific Grant 2009, and the National Office for Research and Technology.

References

[1] Nawa, N. E., Hashiyama, T., Furuhashi, T. and Uchikawa, Y., *A study on fuzzy rules discovery using pseudo-bacterial genetic algorithm with adaptive operator*, Proceedings of IEEE Int. Conf. on Evolutionary Computation, ICEC'97, 1997.

[2] Nawa, N. E. and Furuhashi, T., *Fuzzy Systems Parameters Discovery by Bacterial Evolutionary Algorithms*, IEEE Transactions on Fuzzy Systems 7, 1999, pp. 608-616.

[3] Gál, L., Botzheim, J. and Kóczy, L. T., *Improvements to the Bacterial Memetic Algorithm used for Fuzzy Rule Base Extraction*, Computational Intelligence for Measurement Systems and Applications, CIMSA 2008, Istanbul, Turkey, 2008, pp. 38-43.

[4] Gál, L., Botzheim, J. and Kóczy, L. T., *Modified Bacterial Memetic Algorithm used for Fuzzy Rule Base Extraction*, 5th International Conference on Soft Computing as Transdisciplinary Science and Technology, CSTST 2008, Paris, France, 2008.

[5] K. Ozawa, K. Hirota and L. T. Kóczy, *Fuzzy flip-flop*, In: M. J. Patyra, D. M. Mlynek, eds., Fuzzy Logic. Implementation and Applications, Wiley, Chichester, 1996, pp. 197-236.

[6] R. Lovassy, L. T. Kóczy and L. Gál, *Multilayer Perceptron Implemented by Fuzzy Flip-Flops*, IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, pp. 1683-1688.

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

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

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

[10] Moscato, P., *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.

[11] R. Lovassy, L. T. Kóczy and L. Gál, *Analyzing Fuzzy Flip-Flops Based on Various Fuzzy Operations*, Acta Technica Jaurinensis Series Intelligentia Computatorica vol. 1, no. 3, 2008, pp. 447-465.

[12] R. Lovassy, L. T. Kóczy and L. Gál, *Function Approximation Capability of a Novel Fuzzy Flip-Flop Based Neural Network*, IJCNN 2009 Atlanta - accepted.

[13] L. T. Kóczy and A. Zorat, *Fuzzy systems and approximation*, Fuzzy Sets and Systems, Vol.85, pp. 203-222, 1995.

[14] R. Lovassy, L. T. Kóczy and L. Gál, *Optimizing Fuzzy Flip-Flop Based Neural Networks by Bacterial Memetic Algorithm*, IFSA/EUSFLAT 2009, Lisbon, Portugal, 2009 - accepted