Training of Artificial Neural Networks Using Differential Evolution Algorithm

Adam Slowik, and Michal Bialko
Department of Electronics and Computer Science, Koszalin University of Technology, Koszalin, Poland
aslowik@ie.tu.koszalin.pl

Abstract — **In the paper an application of differential evolution algorithm to training of artificial neural networks is presented. The adaptive selection of control parameters has been introduced in the algorithm; due to this property only one parameter is set at the start of proposed algorithm. The artificial neural networks to classification of parity-p problem have been trained using proposed algorithm. Results obtained using proposed algorithm have been compared to the results obtained using other evolutionary method, and gradient training methods such as: error back-propagation, and Levenberg-Marquardt method. It has been shown in this paper that application of differential evolution algorithm to artificial neural networks training can be an alternative to other training methods.**

Keywords — **artificial intelligence, artificial neural network, differential evolution algorithm, training method.**

I. Introduction

To apply an artificial neural network to any problem solving, it is first necessary to train the network. The training depends on adaptation of free network parameters, that is, on the proper selection of the neural weight values [1], [2]. Specialized learning algorithms are used for adaptation of these weight values [3]. Among those algorithms the most popular is the error back-propagation method (EBP) [2], [4].

This algorithm based on gradient method permits for efficient neural network training to solve difficult problems, which often refer to non-separable data [4], [15], and is a fundamental supervised training algorithm for multi-layer feed-forward neural networks.

However unfortunately, the EBP algorithm possesses a few disadvantages. A huge number of iterations required to obtain satisfactory results, and sensitivity on local minima of error function are mostly enumerated in literature as a disadvantages of this algorithm. Its operation also depends on the value of learning coefficient. When the value of the learning coefficient is chosen too small, then it leads to long operation of the algorithm; but when this value is too high, then it can cause the algorithm oscillation. A few modifications named momentum have been introduced to error back-propagation algorithm for improving its convergence. These modifications depend on addition of extra components, which values are proportional to the value of weight changes in the previous step, to the values of neural network weights.

The other neural network training algorithm is a Levenberg-Marquardt algorithm (LM) [14]. This algorithm modifies values of weights in a group way, after the application of all training vectors. It is one of the most effective training algorithms for the feed-forward neural networks. But also this algorithm possesses some disadvantages. Its inconvenience is mainly tied with computing of error function jacobian and jacobian inversion, to obtain a matrix having dimension equals to the number of all weights in the neural network, therefore the requirement for the memory is very high. This algorithm is also local and there is no guarantee to find a global minimum of the objective function. In the case when the algorithm converges to the local minimum, there is no way of escape, and the obtained solution is sub-optimal.

The objective function describing the artificial neural networks training problem is a multi-modal function, therefore the algorithms based on gradient methods can easily stuck in local extremes. In order to avoid this problem it is possible to use the technique of a global optimization, like for example the differential evolution algorithm [10], [11], which is one of variations of evolutionary algorithms [5], [6], [7]. Differential evolution algorithm has been introduced recently (in the year 1997), and is a heuristic algorithm for global optimization. Its advantages are as follows: a possibility of finding the global minimum of a multi-modal function regardless of initial values of its parameters, quick convergence and a small number of parameters to set up at the start of the algorithm operation [12].

In this paper, an application of differential evolution algorithm for training the feed-forward flat artificial neural networks [15] to classification of parity-p problem is described. Results obtained using differential evolution algorithm have been compared with results obtained using the error back-propagation algorithm [8], [9], evolutionary EA-NNT method [13], and Levenberg-Marquardt algorithm [14]. In this article, the adaptive selection of the control parameter values in differential evolution algorithm has been introduced; due to this only one

1-4244-1543-8/08/$25.00 ©2008 IEEE

parameter, the number of individuals in the population, is required. The method proposed in this paper has been named DE-ANNT (Differential Evolution – Artificial Neural Network Training).

II. Differential Evolution Algorithm

The differential evolution algorithm has been proposed by Price and Storn [10]. Its pseudo-code form is as follows:

Create an initial population consisting of *PopSize* individuals
While (termination criterion is not satisfied)
Do Begin
For each *i*-th individual in the population
Begin
Randomly generate three integer numbers
r*1, *r*2, *r*3 ∈ [1; *PopSize*], where *r*1≠*r*2≠*r*3≠*i
For each *j*-th gene in *i*-th individual (*j*∈[1; *n*])
Begin
$v_{i,j}:=x_{r1,j}+F\cdot(x_{r2,j}-x_{r3,j})$
Randomly generate one real number
$rand_j \in [0; 1)$
If $rand_j < CR$ **then** $u_{i,j}:=v_{i,j}$
Else $u_{i,j}:=x_{i,j}$;
End;
If individual u_i is better than individual x_i then
Replace individual x_i by child u_i individual
End;
End;

The individual x_i is better than individual u_i when the solution represented by it has lower value of objective function (regarding to minimization tasks), or higher - (regarding to maximization tasks) than the solution stored in individual u_i.

The algorithm shown in the pseudo-code optimizes the problem having n decision variables. The F parameter is scaling the values added to the particular decision variables, and the CR parameter represents the crossover rate. The parameters $F \in (0; 2]$, and $CR \in [0; 1)$ are determined by the user, and $x_{i,j}$ is the value of j - th decision variable stored in i - th individual in the population. This algorithm is a heuristic algorithm for global optimization, and is operating with decision variables in real number form. The individuals occurring in this algorithm are represented by real number strings. Its searching space must be continuous [10], [12].

The differential evolution algorithm, by computation of difference between two randomly chosen individuals from the population, determines a function gradient in a given area (not in a single point), and therefore prevents sticking the solution in a local extremum of optimized function [10], [12]. The other important property of this algorithm is a local limitation of selection operator only to the two individuals: parent (x_i) and child (u_i), and due to this property the selection operator is more effective and faster [12].

Also, to accelerate convergence of the algorithm, it is assumed that the index r_1 (occurring in the algorithm pseudo-code) points to the best individual in the population. In this paper such version of differential evolution algorithm has been used in experiments.

III. DE-ANNT Method

Proposed DE-ANNT method operates in several following steps:

In the first step, a population of individuals is randomly created. The number of individuals in the population is stored in parameter *PopSize*. Each individual x_i consists of k genes (where k represents number of weights in trained artificial neural network). In Figure 1a a part of artificial neural network with neurons from n to m are shown. Additionally, in Figure 1b the coding scheme for weights in an individual x_i connected to neurons from Figure 1a, is shown.

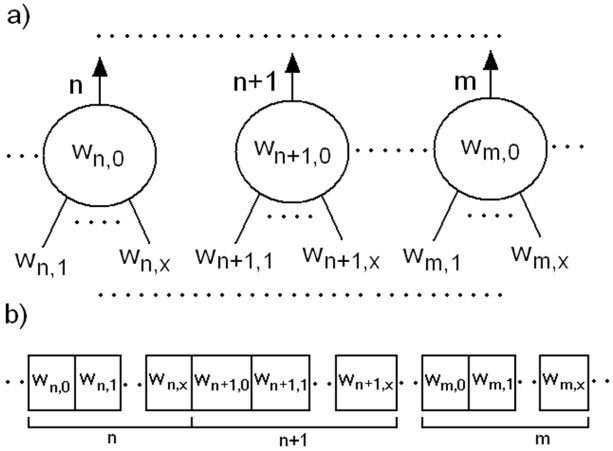

Fig. 1 – Part of artificial neural network (a), corresponding to it chromosome containing the weight values (b); weights $w_{i,0}$ represent bias weights

Each j-th ($j \in [1, k]$) gene of individual x_i can have values from determined range of variability (closed double-sided) from min_j to max_j. In proposed method the values min_j=-1, and max_j=1 are assumed.

In the second step, the mutated individual v_i (vector) is created for each individual x_i in the population according to the formula:

$$v_i = x_{r1} + F \cdot \left(x_{r2} - x_{r3}\right) \tag{1}$$

where: $F \in (0, 2]$, and $r1$, $r2$, $r3$, $i \in [1, PopSize]$ fulfill constraint:

$$r1 \neq r2 \neq r3 \neq i \tag{2}$$

The indices $r2$, and $r3$ point at individuals randomly chosen from the population. Index $r1$ points at the best individual in the population, which has the lowest value of the training error function $ERR(.)$. This function is described as follows:

$$ERR = \frac{1}{2} \cdot \sum_{i=1}^{T} \left(Correct_i - Answer_i\right)^2 \tag{3}$$

where: i – actual number of training vector, T – number of all training vectors, $Correct_i$ – required correct answer for i-th training vector, $Answer_i$ – answer generated by the neural network for i-th training vector applied to its input. The DE-ANNT method is minimizing the value of the objective function $ERR(.)$.

In the third step, all individuals x_i are crossed-over with mutated individuals v_i. As a result of this crossover operation an individual u_i is created. The crossover operates as follows: for chosen individual $x_i = (x_{i1}, x_{i2}, x_{i3}, ..., x_{ij})$, and individual $v_i = (v_{i1}, v_{i2}, v_{i3}, ..., v_{ij})$; for each gene $j \in [1; k]$ of individual x_i, randomly generate a number $rand_j$ from the range [0; 1), and use the following rule:

$$\begin{aligned} &\text{if } rand_j < CR \text{ then} \\ &\quad u_{ij} = v_{ij} \\ &\text{else} \\ &\quad u_{ij} = x_{ij} \end{aligned} \tag{4}$$

where: $CR \in [0; 1)$.

In this paper the adaptive selection of control parameter values F and CR are introduced according to the formulas:

$$F = 2 \cdot A \cdot random \tag{5}$$

$$CR = A \cdot random \tag{6}$$

where:

$$A = \frac{TheBest_i}{TheBest_{i-1}} \tag{7}$$

$random$ - random number with uniform distribution in the range [0; 1), $TheBest_i$ - the value of objective function for the best solution in i-th generation, $TheBest_{i-1}$ - the value of objective function for the best solution in i-1-th generation.

From formulas (5), and (6) we can see that in the case of stagnation (lack of changes of the best solution) the F parameter is taking random values from the range [0; 2), and CR parameter takes random values from the range [0; 1). In such a case, the searching of the solutions space has more global character, and differential evolution algorithm can more easily "get out" from the local extreme causing its stagnation. However, in the case when results obtained by differential evolution algorithm are improving in subsequent generations, then the F parameter is taking random values from the range [0; $2 \cdot A$), and CR parameter takes random values from the range [0; A). Obviously, the value of coefficient A is lower when a higher improvement of obtained results has been occurred between two successive generation. In this case the searching of the solution space has more local character, and can lead to "fine-tuning" of the best solution to the optimal value.

In the fourth step, a selection of individuals to the new population is performed according to following rule:

$$\begin{aligned} &\text{if } ERR(u_i) < ERR(x_i) \text{ then} \\ &\quad \text{Replace } x_i \text{ by } u_i \text{ in the new population} \\ &\text{else} \\ &\quad \text{Leave } x_i \text{ in the new population} \end{aligned} \tag{8}$$

In the fifth step, it is checked , whether the value of $ERR(x_{r1}) < \varepsilon$, or the algorithm has reached the prescribed number of generations (index $r1$ points the best individual with the lowest value of objective function $ERR(.)$ in the population). If yes, then the algorithm is stopped, and result stored in individual x_{r1} is returned. Otherwise, the algorithm jumps to the second step.

IV. The Structure of Assumed Artificial Neural Network and Neuron Model

The proposed DE-ANNT method has been tested by training of feed-forward flat artificial neural network, which structure is shown in Figure 2 (AF – activation function, WS – weighting sum).

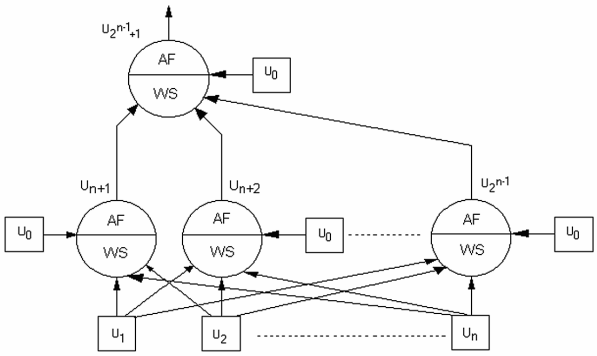

Fig. 2 – Structure of assumed artificial neural network

The classic model of neuron including the adder of input values multiplied by corresponding values of weights - i.e. weighted sum, has been taken as a model of artificial neuron. The weighted sum WS_j of j-th neuron is defined as follows:

$$WS_j = \sum_{i=0}^{p} w_{j,i} \cdot U_i \tag{9}$$

where: p – number of inputs in j-th neuron, $w_{j,i}$ – value of weight representing the connection between j-th neuron and its i-th input, U_i – the value occurring on i-th neuron input.

A bipolar sigmoidal activation function has been assumed in the form:

$$U_j = f(WS_j) = \frac{1 - \exp(-\lambda \cdot WS_j)}{1 + \exp(-\lambda \cdot WS_j)} \tag{10}$$

where: U_j – the value of j-th neuron output, λ - the non-linearity coefficient of activation function; (assumed λ=1).

V. Description of Experiments

The artificial neural networks having structures shown in Figure 2, to classification of parity-p problem ($p \in [3; 7]$) have been trained using proposed and other methods. The example training set was equal to the testing set, and contains 2^p vectors. The following values of parameters have been assumed: PopSize=100, ε=0.0001. Each of algorithms: DE-ANNT (DE), EA-NNT (EA) [13], error back-propagation algorithm (EBP) [8], [9], and Levenbeg-Marquardt algorithm (LM) [14] have been executed 10-fold, and the average values of results are presented in Tables 1-3. The identical termination criterion such as in DE-ANNT method described in the section III of this paper, have been assumed for all other algorithms. The learning coefficient ρ equals to 0.2 has been assumed in the EBP algorithm. Preliminary investigation has been performed before beginning of experiments. The main goal of preliminary investigation was a determination of maximal computation time of algorithms related to the time necessary for convergence of DE-ANNT algorithm. For parity-3 problem this time equals to 1 [s], for parity-4 problem 3 [s], for parity-5 problem 9 [s], for parity-6 problem 60 [s], and for parity-7 problem 200 [s]. Each algorithm have been stopped when the value of training error of artificial neural network had lower value than ε (ε=0.0001) or when operation time exceeded the maximal computation time for each parity problem

.

In Tables 1-3, the symbols used are as follows: *ME* – chosen training method, *NI* – number of iterations, *CC* – correct classification [%], *FC* – false classification [%]. The values representing the correct *CC*, and false *FC* classification were computed as follows:

$$CC = \frac{\sum_{i=1}^{M} C_i}{2^p} \cdot 100\% \qquad (11)$$
$$FC = 100\% - CC$$

where: *CC* – correct classification [%], *M* – number of testing vectors ($M \in [1, 2^p]$), *p* – number of inputs in artificial neural network, C_i – coefficient representing correctness of classification of *i*-th training vector which is determined as follows:

$$C_i = \begin{cases} 1, & when\ U_{out} > \varphi\ for\ B_i = 1 \\ 1, & when\ U_{out} < -\varphi\ for\ B_i = -1 \\ 0, & otherwise \end{cases} \qquad (12)$$

where: $U_{out}=f(S_{out})$ – value of the output signal of artificial neural network after application of *i*-th testing vector to its input, φ - threshold of training correctness, B_i – value expected at the output of the artificial neural network.

In Figures 3-5, the graphical representation of φ parameter for its successive values: φ=0 (Figure 3), φ=0.9 (Figure 4), φ=0.99 (Figure 5) is presented. In all Figures, the dashed line represents assumed sigmoidal activation function.

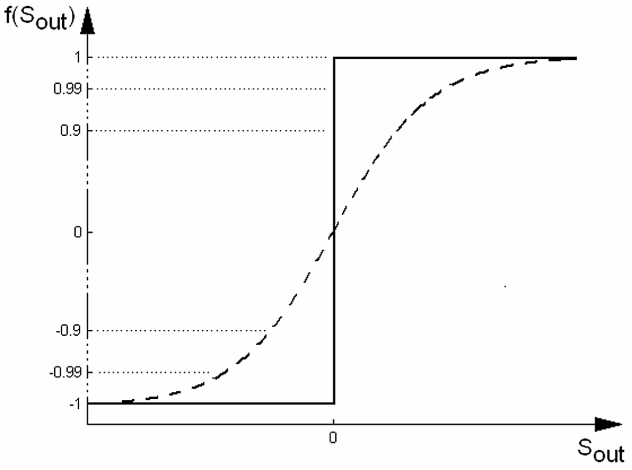

Fig. 3 – Threshold of training correctness for φ=0

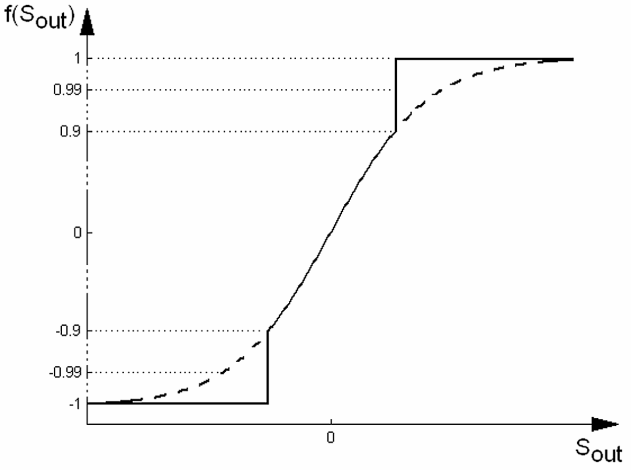

Fig. 4 – Threshold of training correctness for φ=0.9

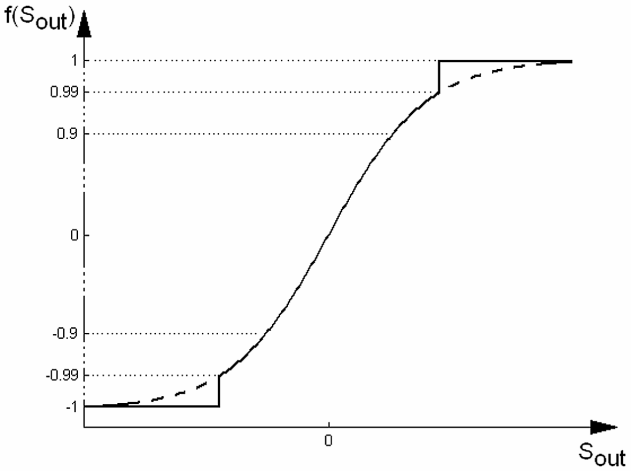

Fig. 5 - Threshold of training correctness for φ=0.99

In the first experiment the value φ=0 has been assumed, and results obtained are shown in Table 1.

TABLE 1: AVERAGE VALUES OF RESULTS OBTAINED AFTER 10-FOLD REPETITION OF EACH ALGORITHM (φ=0)

Parity-3 Problem			
ME	NI	CC [%]	FC [%]
DE	23.5	**100**	0
EA	57.7	95.0	5.0
EBP	300	78.75	21.25
LM	20	**100**	0
Parity-4 Problem			
ME	NI	CC [%]	FC [%]
DE	399.2	96.875	3.125
EA	147.1	92.5	7.5
EBP	800	91.25	8.75
LM	48.1	**97.5**	2.5
Parity-5 Problem			
ME	NI	CC [%]	FC [%]
DE	657.5	96.875	3.125
EA	348.9	92.1875	7.8125
EBP	2250	94.6875	5.3125
LM	64.1	**97.1875**	2.8125
Parity-6 Problem			
ME	NI	CC [%]	FC [%]
DE	1988	95.78125	4.21875
EA	1486.6	94.53125	5.46875
EBP	10800	96.09375	3.90625
LM	391.2	**97.34375**	2.65625
Parity-7 Problem			
ME	NI	CC [%]	FC [%]
DE	2828.3	**98.671875**	1.328125
EA	2716.1	91.64063	8.359375
EBP	31000	96.7969	3.2031
LM	1527.8	95.70314	4.29686

It can be seen from Table 1, that results obtained using DE-ANNT algorithm for φ=0 are in four cases better than results obtained using EBP algorithm, in five cases better than results obtained using EA-NNT algorithm, and in two cases comparable or better than results obtained using LM algorithm. It can be also noticed, that DE-ANNT algorithm allows to obtain a value of error function *ERR*(.) lower than ε before reaching the maximal computational time for this algorithm. In comparison, for the EBP algorithm, its termination criterion, following from achieving the ε value by the error function *ERR*(.) in assumed computational time was never reached. The EBP algorithm was always stopped after achieving the maximal computation time. For that reason, the classification criteria has been "sharpened", in order to show that together with an increase of the threshold φ value, the results obtained by proposed DE-ANNT algorithm are successively better compared to results obtained using EBP algorithm.

In the second experiment, similar investigations as in first experiment have been performed, but the value of parameter φ has been changed to φ=0.9, and to φ=0.99 values. The average results obtained using 10-fold repetition for each algorithm are shown in Table 2 (φ=0.9), and in Table 3 (φ=0.99).

TABLE 2: AVERAGE VALUES OF RESULTS OBTAINED AFTER 10-FOLD REPETITION OF EACH ALGORITHM (φ=0.9)

Parity-3 Problem			
ME	NI	CC [%]	FC [%]
DE	24.9	**100**	0
EA	56.4	81.25	18.75
EBP	300	36.25	63.75
LM	19	**100**	0
Parity-4 Problem			
ME	NI	CC [%]	FC [%]
DE	442.4	87.5	12.5
EA	154.5	72.5	27.5
EBP	800	69.375	30.625
LM	49.9	**97.5**	2.5
Parity-5 Problem			
ME	NI	CC [%]	FC [%]
DE	420.3	**96.5625**	3.4375
EA	351	67.8125	32.1875
EBP	2250	69.6875	30.3125
LM	161.7	**96.5625**	3.4375
Parity-6 Problem			
ME	NI	CC [%]	FC [%]
DE	2058.4	89.84375	10.15625
EA	1505.4	78.75	21.25
EBP	10800	85.78125	14.21875
LM	1044.2	**97.8125**	2.1875
Parity-7 Problem			
ME	NI	CC [%]	FC [%]
DE	2414.8	87.265625	12.734375
EA	2752.9	80	20
EBP	31000	86.71877	13.28123
LM	198.9	**94.76563**	5.23437

TABLE 3: AVERAGE VALUES OF RESULTS OBTAINED AFTER 10-FOLD REPETITION OF EACH ALGORITHM (φ=0.99)

Parity-3 Problem			
ME	NI	CC [%]	FC [%]
DE	25.3	**96.25**	3.75
EA	55.2	65	35
EBP	300	6.25	93.75
LM	21.7	71.25	28.75
Parity-4 Problem			
ME	NI	CC [%]	FC [%]
DE	485.4	81.25	18.75
EA	153.3	60.625	39.375
EBP	800	2.5	97.5
LM	33.9	**81.875**	18.125
Parity-5 Problem			
ME	NI	CC [%]	FC [%]
DE	640.4	**86.875**	13.125
EA	350.4	60.3125	39.6875
EBP	2250	17.1875	82.8125
LM	57.7	84.375	15.625
Parity-6 Problem			
ME	NI	CC [%]	FC [%]
DE	2025.3	83.59375	16.40625
EA	1498.9	65	35
EBP	10800	41.09375	58.90625
LM	154.6	**85.9375**	14.0625

TABLE 3: CONTINUATION

Parity-7 Problem			
ME	NI	CC [%]	FC [%]
DE	2845.7	82.1875	17.8125
EA	2739.8	59.53125	40.46875
EBP	31000	51.32815	48.67185
LM	3664.5	**87.4219**	12.5781

It can be seen from Table 2, and Table 3, that differences between results obtained using EBP, EA and DE-ANNT algorithms are growing together with increasing of the threshold of training correctness value. The results obtained using DE-ANNT algorithm are better than results obtained using EBP, and EA algorithms. The results obtained by the DE-ANNT algorithm are better (having higher percentage of correctly classified data) or comparable in four cases on ten possible, compared to the results obtained using LM algorithm.

VI. CONCLUSIONS

Based on the results shown in Tables 1-3, it can be seen, that training artificial neural networks using DE-ANNT algorithm, it is possible to obtain better data classification in the same time period (more number of input vectors are correctly classified) than for the EA or EBP algorithms. It is also worth to notice, that results obtained using DE-ANNT algorithm are less sensitive to changes of the threshold of training correctness φ value in comparison to results obtained using EA, and EBP algorithms. Therefore, one can say, that using proposed DE-ANNT algorithm better trained artificial neural network can be obtained at the presumed time, than using EA, and EBP algorithms (more data are correctly classified for increasing values of parameter φ). In comparison to LM algorithm, the results obtained using DE-ANNT algorithm are promising. It is necessary to point out that presented algorithm can be easily used to training of multi-output artificial neural networks, to training artificial neural networks with non-standard architectures (for example the tower architecture [15]), and non-differentiable neuron activation function for which the application of EBP or LM algorithms is useless. Additionally, an introduction of adaptive changes of F and CR parameter values, in presented in this paper DE-ANNT algorithm, allows for increasing the automation of the training process in relation to other evolutionary methods dedicated to the same problem. The possibilities of described DE-ANNT algorithm will be the goal of a future research.

ACKNOWLEDGMENT

The authors are grateful to the anonymous referees for their valuable comments, which have improved the presentation of this contribution.

REFERENCES

[1] R. Hecht – Nielsen, *Neurocomputing*. Addison – Wesley Publishing Company, 1991.

[2] J. Zurada, *Introduction to Artificial Neural Systems*. West Publishing Company, St. Paul, 1992.

[3] N. Jankowski, *Ontogenic Neural Networks*. Academic Publishing House EXIT, Warsaw, 2003, (in Polish).

[4] M. Bialko, *Artificial Intelligence and Elements of Hybrid Expert Systems*. Publishing House of Koszalin University of Technology, Koszalin, 2005, (in Polish).

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

[6] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1998.

[7] J. Arabas, *Lectures on Evolutionary Algorithms*. WNT, Warsaw, 2001, (in Polish).

[8] L. Fu, *Neural Networks in Computer Intelligence*. McGraw-Hill, Inc., New York, 1994.

[9] T. Masters, *Practical Neural Network Recipes in C++*. Academic Press, New York, 1993.

[10] R. Storn, and K. Price, "Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, Vol. 11, pp. 341-359, 1997.

[11] K. Price, "An Introduction to Differential Evolution," in David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pp. 79-108, McGraw-Hill, London, UK, 1999.

[12] R. L. Becerra, C. A. Coello Coello, "Cultured Differential Evolution for Constrained Optimization," *Computer Methods in Applied Mechanics and Engineering*, Volume 195, Nos. 33-36, pp. 4303-4322, July 1, 2006.

[13] A. Slowik, M. Bialko, "Application of evolutionary algorithm to training of feedforward flat artificial neural networks," *Proceedings of XIV-th National Conference on Computer Application in Scientifics Research*, Wroclaw Scientific Society, pp. 35-40, 2007, (in Polish).

[14] A. Osowski, *Neural networks in algorithmic use*. WNT, Warsaw, 1996, (in Polish).

[15] S. Gallant, *Neural Network Learning and Expert Systems*, MIT Press, Cambridge, MA, 1993.