Parallel Differential Evolution Algorithm with Multiple Trial Vectors to Artificial Neural Network Training

HtetThazin Tike Thein, Khin Mo MoTun
University of Computer Studies, Yangon
htetthazintikethein@ucsy.edu.mm, khinmo2tun@gmail.com

Abstract

In this paper, parallel differential evolution algorithm with multiple trial vectors for training artificial neural networks (ANNs) is presented. The proposed method is PDEA, which is a DE-ANN+ modified by adding island model. Within PDEA, an island model is designed to cooperatively search for the global optima in search space. By combining the strengths of the differential evolution algorithm with multiple trial vectors and island model, PDEA greatly improves the optimization performance. PDEA algorithm is used for ANN training to classify the parity-p problem. Results obtained using proposed algorithm has been compared to the results obtained using other evolutionary algorithms.

***Keywords**: Artificial intelligence, artificial neural network, differential evolution algorithm, multiple trial vectors, training method, island model.*

1. Introduction

Artificial Neural Networks with feed-forward structure (ANNs) are widely used in regression, prediction, and classification. The problem of ANN training is formulated as the minimization of an error function in the space of connection weights. Typical ANN training methods e.g. back-propagation and conjugate gradient algorithms are based on gradient descent. Algorithms based on gradient methods can easily get stuck in local minima. To avoid this problem, it is possible to use the technique of a global optimization, like for example the differential evolution algorithm [1], [2], which is one of variations of evolutionary algorithms [3], [4], [5]. Differential evolution algorithm has been introduced recently (in the year 1997), and is a heuristic algorithm for global optimization. It advantages are as follows: a possibility of finding the global optimum 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 [6].

The interest of the Evolutionary Computation community in more complex problems requires the use of advanced models of Evolutionary Algorithms (EAs), because standard models are not powerful enough. One family of such models are island models, in which individuals are split into sub-populations (islands), evolving on their own, and, from time to time exchanging individuals by migrations. Increasing pressure to solve real world complex problems has led to the development of Parallel Evolutionary Algorithms (PEAs) which exploit the intrinsically parallel nature of EAs. An extensive review of parallel evolutionary models, parallel implementations, and pressing theoretical issues can be found in [9]. Parallelization of an evolutionary algorithm can be done at any of the following levels: objective function evaluation level (master-slave) model, population level (island model or migration model) and elements level (cellular level). The first two lead to coarse grained parallelization while the cellular model leads to fine grained parallelization. In this paper, PDEA algorithm is used for ANN training to classify the parity-p problem. The results from the obtained algorithm have beencompared with results from the following algorithms: an evolutionary algorithm, a DE algorithm with multiple trial vectors, a DE algorithm without multiple trial vectors,gradient training methods, such as error back-propagation, andtheLevenberg-Marquardt method.

2. Related Work

Since 1997, the DE algorithm has been modified to increase its effectiveness. The introduction of adaptive selection of control parameters in the DE algorithm means that better results can be obtained in the same period of time, and the algorithm is less sensitive to dimensionality changes in the task being optimized [7]. Also, in 2007, the concept ofmultiple trial vectors [8] was introduced to the DE algorithm. This approach is based on the generation of a higher number of mutated individuals around the existing individuals (solutions). Because of this, the probability of generating a better solution is increased [8].Many applications have shown that parallel evolutionary algorithms can speed up computation and find better solutions, compared to a sequential evolutionary algorithm. Over the past few years, considerable amount of work has been done on parallelization using island model (IM) strategy [10], [11], [12], [13], [14].

In this paper, parallel differential evolution algorithm with multiple trial vectors is proposed. PDEA is designed for improving the optimization performance of the component algorithm. The proposed method is a modified DE-ANNT+ method [15] with the island model added. DE-ANNT+ [15] with the multiple trial vectors techniques is also used for ANN training to classify the party-p problem.The results obtained from using the proposed method have been compared with the results obtained from using the error back-propagation algorithm [16], [17], the Evolutionary Algorithm-NeuralNetwork Training (EA-NNT) method [18], the Levenberg-Marquardt (LM) algorithm [19], the DE-ANNT method [20] and the DE-ANNT+ method [15].This paper is an extension of [15], in which an ANN training algorithm based on DE+ algorithm was presented.

3. Background

3.1. Differential Evolution Algorithm

The differential evolution algorithm has been proposed by Price and Stron [1]. 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:
        $r_1,r_2,r_3 \in$ [1;PopSize], where $r_1 \neq r_2 \neq r_3 \neq i$
        For each $j^{th}$ gene in $i^{th}$ individual ($j \in$ [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 thanindividual $x_i$thenreplaceindividual$x_i$by child $u_i$ individual
    End;
End;
```

The individual x_i is better than individual u_i when the solution represented by it has a lower value of the objective function (regarding minimization tasks) or a higher value (regarding maximization tasks) than the solution stored in individual u_i. The algorithm shown in the pseudo-code optimizes the problem with n decision variables. The F parameter scales 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 the j^{th} decision variable stored in the i^{th} individual in the population. This algorithm is a heuristic algorithm for global optimization and is operated by using decision variables in a real number form. The individuals occurring in this algorithm are represented by real number strings. Its searching space must be continuous [1], [6]. By computing the difference between two individuals chosen randomly from the population, the DE algorithm determines a function gradient within a given area (not at a single point). Therefore, the DE algorithm prevents thesolution sticking at a local extreme of the optimized function [1], [6]. Another important property of this algorithm is a local limitation of the selection operator to only two individuals (parent (x_i) and child (u_i)), and, owing to this property, the selection operator is more effective and faster [6]. Also, to accelerate the 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.

3.2.Island Model Strategy

Independent runs suffer from obvious drawbacks: once a run reaches a situation where its population has become stuck in a difficult local optimum, it will most likely remain stuck forever. This is unfortunate since other runs might reach more promising regions of the search space at the same time. It makes more sense to establish some form of communication between the different runs to coordinate search, so that runs that have reached low-quality solutions can join in on the search in more promising regions.

In island models, also called distributed EAs, coarse-grained model, or multi-deme model, the population of each run is regarded as an island. One often speaks of islands as subpopulations that together form the population of the whole island model. Island evolves independently as in the independent run model, for most of the time. But periodically solutions are exchanged between islands in a process called migration.

The idea is to have a migration topology, a directed graph with islands as its nodes and directed edges connecting two islands. At certain points of time selected individuals from each island are sent off to neighboring islands, i.e., islands that can be reached by a directed edge in the topology. These individuals are called migrants and they are included in the target island after a further selection process. This way, islands can communicate and compete with one another. Islands that got stuck in low-fitness regions of the search space can be taken over by individuals from more successful islands. This helps to coordinate search, focus on themost promising regions of the search space, and use the available resources effectively.

In the island model approach, each island executes a standard sequential evolutionary algorithm. The communication between sub-population is assured by a migration process. Some randomly selected individuals (migration size) migrate from one island to another after every certain number of generations (migration interval) depending upon a communication topology (migration topology). The two basic and most sensitive parameters of island model strategy are: migration size, which indicates the number of individuals migrating and controls the quantitative aspect of migration; and migration interval denoting the frequency of migration.

4.Proposed PDEA Method

The proposed PDEA method is based on the previously elaborated DE-ANNT+ method [15] and operates according to the 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 the number of weights in the trained ANN). In Fig 1. (a), a part of an ANN with neurons from n to m is shown. Additionally, in Fig 1(b), the coding scheme for weights in an individual x_i connected to neurons from Fig 1. (a) is shown.

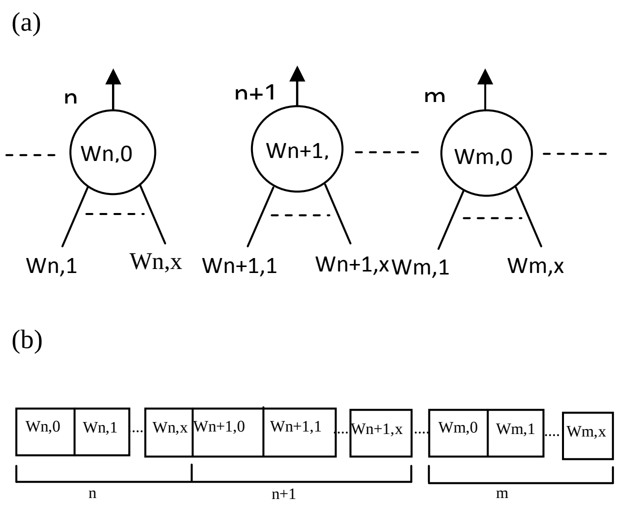

Figure 1.Part of (a) ANN, corresponding to its (b) chromosome containing the weight values; w*i,0* represent bias weights [15].

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

In the second step, the NT (number of trial vectors) mutated individuals (trial vectors) $V_{i,m}$($m \in [1,NT]$) are created for each individual x_i in the population, according to the formula

$$V_{i,m} = x_{r1} + F(x_{r2} - x_{r3}) \quad (1)$$

where $F \in [0,2)$, and $r_1, r_2, r_3, i \in [1, PopSize]$ fulfill the constraint

$$r_1 \neq r_2 \neq r_3 \neq i \quad (2)$$

Indexes r_2 and r_3 point to individuals randomly chosen from the population. Index r_1 points to the best individual in the population, which has the lowest value of the training error function, ERR (.). This function is described as follows:

$$ERR = \frac{1}{2}\sum_{i=1}^{T}(Correct_i - Answer_i)^2 \quad (3)$$

where*i* is the actual number of training vector, T is the number of all training vectors; $Correct_i$ is the required correct answer for the *i*th training vector, and $Answer_i$is the answer generated by the neural network for the *i*th training vector applied to its input. The DE-ANNT+ method minimizes the value of the objective function ERR (.). From the created set of mutated vectors $V_{i,m}$, only one vector $V_{i,m}$ (individual),having the lowest value of the objective function ERR (.), is chosen for each individual x_i, and it is assigned as vector v_i.

In the third step, all individuals x_i are crossed over with their 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 $x_i = (x_{i,1},x_{i,2},x_{i,3},...,x_{i,j})$ and individual $v_i = (v_{i,1},v_{i,2},v_{i,3},...,v_{i,j})$; 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:

If $rand_j < CR$ then $u_{i,j} = v_{i,j}$
Else $u_{i,j} = x_{i,j}$

where CR ∈ [0; 1).

In this paper, an adaptive selection of control parameter values F and CR is introduced (similarly as in [7]) according to the formulas

$$A = \frac{TheBest_i}{TheBest_{i-1}} \tag{4}$$

$$F = 2A(random) \tag{5}$$

$$CR = A(random) \tag{6}$$

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

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

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

If ERR (u_i) < ERR (x_i) then
Replace x_i by u_i in the new population
Else leave x_i in the new population

In the fifth step, it is checked whether the value of ERR (x_{r1}) <*e*, or if the algorithm has reached the prescribed number of generations (indexr1points to the best individual with the lowest value of the objective function, ERR, in the population). If this is the case, the result stored in individual x_{r1} is returned and the algorithm goes to next step. Otherwise, the algorithm jumps to the second step.

In the sixth step, select individuals from parallel differential evolution algorithm according to migration policy.

In the seventh step, migrate and replace individuals according to migration topology.

In the eighth step, stop if the stop criterion is satisfied; otherwise, go to second step. In the proposed system, number of iterations and limited error are used as criteria.

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

The proposed PDEA method has been tested by training of feed-forward flat artificial neural network. Fig 2 shows the typical neural network. (*AF* – activation function: *WS* – weighting sum).

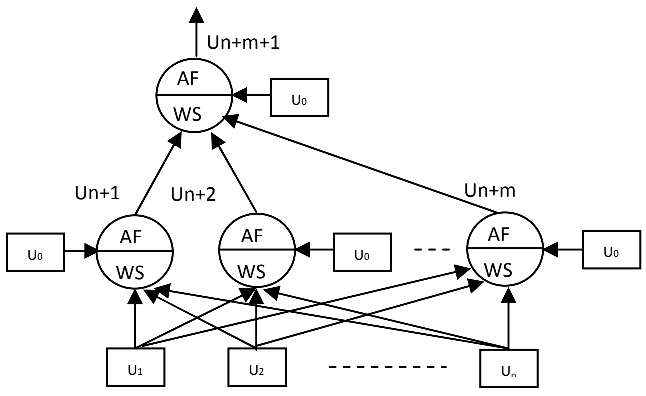

Figure 2.Structure of artificial neural network

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

$$WSj = \sum_{i=0}^{p} w_{j,i} U_i \tag{7}$$

where p – the number of inputs in the jth neuron; $w_{j,i}$– the value of weight representing the connection between the *j*th neuron and its input; U_i– the value of the *i*th input of the *j*th neuron.

A bipolar sigmoidal activation function has been assumed in the form

$$U_j = f(WS_j) = \frac{1-\exp(-\lambda WS_j)}{1+\exp(-\lambda WS_j)} \quad (8)$$

whereU_j – the value of the *j*th neuron output; λ - the nonlinearity coefficient of the activation function (assumed λ =1).

6.Description of Experiments

The proposed parallel differential evolution algorithm with multiple trial vectors (PDEA) to classify the parity-p problem. There were five islands used to realize PDEA. The differential evolution algorithm has been adapted for solving and assigned to every island in PDEA. Island model used different subpopulation with each own island. Each island operates its own execution as like in DE algorithm. Each island initializes the population at the start of the algorithm or replace the subpopulation migrates from other neighbor. Mutation, crossover and selection are performed on the individual chromosome. If the migration interval is not fired, the next iteration begin within island, otherwise, a portion of its own population and neighbor is selected for migration. If the migration occurs, island sends sub-population to neighbor island. Neighbor island replaces the sub-population send by its neighbor and replace with its portion of population and algorithm continue. To classify the parity-p problem ($p \in [3; 6]$), ANNs having structures shown in Fig. 2, were trained using the proposed method and other methods for comparison. The parity-p problem is described as follows: if p presents the number of inputs, and each input can accept values “1” or “-1”, then, in the output of the network, “1” occurs if and only if the number of “1” in the inputs of the ANN is odd. Otherwise “-1” occurs in the output of the ANN.

The example training set was equal to the testing set and contained 2^p vectors. The following values of parameters were assumed: PopSize = 100 and e = 0.0001. Comparative results obtained using the algorithms DE-ANNT+ [15], DE-ANNT [20], EA-NNT [18], the EBP algorithm [16], [17], and the LM algorithm [19], were taken from [15].

In the experiments we used identical islands, i.e, islands with same parameters. We used a ring topology for our experiments. We used five islands because no significant change was noticed in the nature of the algorithm with a change in the number of islands. The policy of migration used was best-random policy in which best string from an island replaces any other random string of another island based on the ring topology. Each island executed the standard DE algorithm with the total population size of 20 individuals per island. The PDEA algorithm presented in this paper was stopped when the training error value of the ANN was lower than e =0.0001 or when operation time exceeded the maximal computation time for each parity-p problem.

In table 1-3, the symbols used are as follows: NT-the number of trial vectors; ME-the training method chosen: NI-the number of iterations; CC-the correct classification (%); FC-the false classification (%). The values representing the correct CC and false FC classifications were computed as follows:

$$CC = \left(\frac{\sum_{i=1}^{M} C_i}{2^p}\right)100\% \quad (9)$$

$$FC = 100\% - CC \quad (10)$$

where CC—the correct classification (%); M—the number of testing vectors ($M \in [1,2^p]$); p—the number of inputs in the ANN; C_i—the coefficient representing the correctness of the classification of the ith training vector which is determined as follows:

$$C_i = \begin{cases} 1, when\ U_{out} > \varphi\ for\ B_i = 1 \\ 1, when\ U_{out} < -\varphi\ for\ B_i = -1 \\ 0, otherwise \end{cases} \quad (11)$$

where $U_{out} = f(S_{out})$—the value of the output signal of the ANN after the application of the *i*th testing vector to its input; φ —the threshold of the training correctness; B_i—the value expected for the output of the ANN. Artificial neural networks were trained using the proposed PDEA method for different values of NT ∈ [1; 10] and parameter φ =0.99.

Table 1.Average Values for Different Values of NT and φ = 0.99

	Problem Parity – 3		Problem Parity - 4	
NT	**NI**	**CC (%)**	**NI**	**CC(%)**
1	22.4	98.75	374	76.875
2	22.8	98.75	233.6	83.125
3	22.3	98.75	186.6	83.750
4	22.4	100	162.3	85
5	29.3	93.75	126.1	85
6	28.6	97.50	129.4	78.125
7	20.2	98.75	116.2	80
8	17.9	98.75	100.9	82.5
9	17.4	98.75	91.5	81.25
10	9.0	98.75	82.4	82.4
	Problem Parity - 5		**Problem Parity - 6**	
NT	**NI**	**CC (%)**	**NI**	**CC(%)**
1	461.7	86.5625	1433.6	80.625
2	203	89.0625	974.8	75.15625
3	233.1	84.375	772.8	80.9375
4	178.6	78.4375	631.9	80.9375
5	169.7	73.125	569.3	75.78125
6	147.7	82.625	493.7	72.1875
7	144.5	79.375	442.7	60.15625
8	123.4	80.3125	401.7	49.53125
9	110.8	83.125	368	50.15625
10	103.6	77.1875	336.1	70.15625

In Table 1, the average values of the results were taken from [15]. It can be seen from Table 1 that the best results (the highest values of CC) are obtained for values NT ∈ [1; 4]. Therefore, during the next experiment, the value NT equal to 3 was assumed. The PDEA algorithm was executed tenfold, and the average values of the results obtained for φ=0.90 and φ=0.99 are presented in Table 2 (φ=0.90) and in Table 3 (φ=0.99). The comparative results in both tables are taken from [15]. The φ values were chosen experimentally according to the author’s previous experience.

Table 2.Average Values of Results Obtained After Tenfold Repetition of PDEA Algorithm (φ= 0.90)

Problem Parity-3				Problem Parity-4		
ME	**NI**	**CC (%)**	**FC (%)**	**NI**	**CC (%)**	**FC (%)**
PDEA	15	100	0	30	98.685	1.315
DE+	27.3	100	0	200.3	88.125	11.875
DE	24.9	100	0	442.4	87.5	12.5
EA	56.4	81.25	18.75	154.5	72.5	27.5
EBP	300	36.25	63.75	800	69.375	30.625
LM	19	100	0	49.9	97.5	2.5
Problem Parity-5				**Problem Parity-6**		
ME	**NI**	**CC (%)**	**FC (%)**	**NI**	**CC (%)**	**FC (%)**
PDEA	100	97.489	2.511	500	98.9035	1.0965
DE+	211.9	91.5625	8.4375	737.9	93.28125	6.71875
DE	420.3	96.5625	3.4375	2058.4	89.84375	10.15625
EA	351	67.8125	32.1875	1505.4	78.75	21.25
EBP	2250	69.6875	30.3125	10800	85.78125	14.21875
LM	161.7	96.5625	3.4375	1044.2	97.8125	2.1875

Table 3. Average Values of Results Obtained After Tenfold Repetition of PDEA Algorithm (φ = 0.99)

Problem Parity-3				Problem Parity-4		
ME	**NI**	**CC (%)**	**FC (%)**	**NI**	**CC (%)**	**FC (%)**
PDEA	20	100	0	31	90.556	9.444
DE+	33.7	98.75	1.25	198.4	83.75	16.25
DE	25.3	96.25	3.75	458.4	81.25	18.75
EA	55.2	65	35	153.3	60.625	39.375
EBP	300	6.25	93.75	800	2.5	97.5
LM	21.7	71.25	28.75	33.9	81.875	18.125
Problem Parity-5				**Problem Parity-6**		
ME	**NI**	**CC (%)**	**FC (%)**	**NI**	**CC (%)**	**FC (%)**
PDEA	49	91.671	8.329	103	93.474	6.526
DE+	234.6	86.875	13.125	773.4	80.9375	19.0625
DE	640.4	86.875	13.125	2025.3	83.59375	16.40625
EA	350.4	60.3125	39.6875	1498.9	65	35
EBP	2250	17.1875	82.8125	10800	41.09375	58.90625
LM	57.7	84.375	15.625	154.6	85.9375	14.0625

From Tables 2 and 3, it can be seen that, for the threshold of training correctness values φ=0.90 and φ=0.99, the application of the proposed PDEA method caused an increasein the correct classification of data as compared to the DE-ANNT+ method. For all cases, better results (having a higher percentage of correctly classified data) were obtained using the proposed method than by using the DE-ANNT+ method. Also, results obtained using the PDEA algorithm are better than the results obtained by using DE-ANNT, EBP, EA, and LM algorithms.

Also, it can be seen from Tables 2 and 3 that the number of iterations (NI) of EBP increases as the maximal time increases, but the NI of LM does not. This is caused by the fact that, for the EBP method, the

ANN training error value was not lower than e=0.0001 after the maximal time. Therefore, in all cases, the EBP method was stopped after the same number of iterations, but with the LM algorithm, the computations were often stopped before the maximal time was reached.

7. Conclusion

This paper presents parallel differential evolution algorithm with multiple trial vectors (PDEA) for artificial neural network training to classify the parity-p problem. Based on the results shown in Tables 2 and 3, it can be seen that training of artificial neural networks by using the PDEA algorithm increases the efficiency of the data classification in the same period when compared with the EA, EBP, DE-ANNT, DE-ANNT+, or LM algorithms. Therefore, one can say, that using proposed PDEA algorithm better trained artificial neural network can be obtained at the presumed time, than using EA, EBP, DE-ANNT, DE-ANNT+, or LM algorithms (more data are correctly classified for increasing values of parameter φ). Additionally, an introduction of parallel differential evolution algorithm is presented in this paper. Also, it is worth saying that the proposed PDEA algorithm can be used in many industrial electronics applications in which the use of artificial neural network is needed.

References

[1] 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.

[2] K. Price, “An Introduction to Differential Evoluiton,” in David Corne, Marco Dorigo, and Fred Glover, editors, New Ideas in Optimization, pp. 79-108, McGraw-Hill, London, UK, 1999.

[3] D. Goldberg, Genetic Algorithm in Search, Optimization, and Machine Learning. Addison-Wesley Professional, 1989.

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

[5] J. Arabas, Lectures on Evolutinary Algorithms. WNT, Warsaw, 2001, (in Pollish).

[6] R. L. Becerra and C. A. CoelloCoello, “Cultured differential evolutionfor constrained optimization,”Comput. Methods Appl. Mech. Eng., vol. 195, no. 33–36, pp. 4303–4322, Jul. 1, 2006.

[7] A. Slowik and M. Bialko, “Adaptive selection of control parameters in differential evolution algorithms,” inComputational Intelligence: Methods and Applications, L. Zadeh, L. Rutkowski, R. Tadeusiewicz, and J. Zurada, Eds. Warsaw, Poland: EXIT, pp. 244–253, 2008.

[8] E. Mezura-Montes, C. A. CoelloCoello, J. Velázquez-Reyes, and L. Munoz-Dávila, “Multiple trial vectors in differential evolution for engineering design,”Eng. Optim., vol. 39, no. 5, pp. 567–589, Jul. 2007.

[9] E.Alba and M. Tomassini, “Parallelism and Evolutionary Algorithms,” IEEE Trainsactions on Evolutionary Computation, vol. 6, no. 3, pp 373-388, Aug. 1998.

[10] Z. Skolicki and K. De Jong, “Improving evolutionary algorithms with multi-representation island models,” inParallel Problem Solvingfrom Nature - PPSN VIII 8th International Conference, Springer-Verlag, 2004.

[11] D. Whitley, S. Rana and R. B. Heckendorn, “The island model genetic Algorithm: On separability, population size and convergence,”Jnl.ofComputingand Information Technology, vol. 7, no. 1, pp. 33–47, 1999.

[12] E. Cantu-Paz, “Migration policies, selection pressure, and parallel evolutionary algorithms,”Jnl.ofHeuristics, vol. 7, no. 4, pp. 311–334, 2001.

[13] R. Storn, “On the usage of differential evolution for function optimization,” in Proc. Of the 1996 Biennial Conference of the North American Fuzzy Information processing society – NAFIPS, Edited by: M. H. Smith, M. A. Lee, J. Keller nad J. Yen, June 19-22, Berkeley, CA, USA, IEEE Press, New York, pp 519-523.

[14] L. Singh and S. Kumar, “Parallel Evolutionary Asymmetric Subsethood Product Fuzzy-Neural Inference System: An Island Model Approach,” inProceedingsof the InternationalConference on Computing: Theoryand Applications(ICCTA-2007), pp. 282-286, 2007.

[15] A. Slowik, “Application of an Adaptive Differential Evolution Algorithm with Multiple Trial Vectors to Artificial Neural Network Training,” IEEE, Transactions on Industrial Electronics, Vol. 58. No. 8, August, 2011.

[16] L. Fu, Neural Networks in Computer Intelligence.New York: McGraw- ill, 1994.

[17] T. Masters,Practical Neural Network Recipes in C++. New York: Academic, 1993.

[18] A. Slowik and M. Bialko, “Application of evolutionary algorithm to training of feedforward flat artificial neural networks,” (in (in Polish)), inProc.KOWBAN, pp. 35–40, 2007.

[19] A. Osowski, (in Polish), Neural Networks in Algorithmic Use. Warsaw, Poland: WNT, 1996.

[20] A. Slowik and M. Bialko, “Training of artificial neural networks using differential evolution algorithm,” inProc. IEEE Conf. Human Syst. Interaction, Cracow, Poland, pp. 60–65, May 25–27, 2008.

[21] Z. Skolicki and K. De Jong, “The influence of migration sizes and intervals on island models,” in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005), ACM Press, 2001.