World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:3, 2008

A New Self-Adaptive EP Approach for ANN Weights Training

Kristina Davoian, and Wolfram-M. Lippe

***Abstract*—**Evolutionary Programming (EP) represents a methodology of Evolutionary Algorithms (EA) in which mutation is considered as a main reproduction operator. This paper presents a novel EP approach for Artificial Neural Networks (ANN) learning. The proposed strategy consists of two components: the self-adaptive, which contains phenotype information and the dynamic, which is described by genotype. Self-adaptation is achieved by the addition of a value, called the *network weight*, which depends on a total number of hidden layers and an average number of neurons in hidden layers. The dynamic component changes its value depending on the fitness of a chromosome, exposed to mutation. Thus, the mutation step size is controlled by two components, encapsulated in the algorithm, which adjust it according to the characteristics of a predefined ANN architecture and the fitness of a particular chromosome. The comparative analysis of the proposed approach and the classical EP (Gaussian mutation) showed, that that the significant acceleration of the evolution process is achieved by using both phenotype and genotype information in the mutation strategy.

***Keywords*—**Artificial Neural Networks (ANN), Learning Theory, Evolutionary Programming (EP), Mutation, Self-Adaptation.

I. Introduction

MUTATION-BASED Evolutionary Algorithms, also known as Evolutionary Programming, have been successfully applied not only to combinatorial optimization problems but also to other areas of Artificial Intelligence (AI), where they are used as subsidiary evolutionary approaches for some parameters optimization, e.g. for ANNs connection weights training [1-13]. The classical strategy used in EP is Gaussian mutation operator, denoted as the classical EP (CEP), based on a standard normal distribution [1, 5-9]. The main disadvantage of this approach is its slow convergence to a near-optimal solution. To overcome this drawback research has been devoted towards the development of new mutation strategies. Yao and Liu [15, 16] have proposed an alternative mutation technique, called the Fast Evolutionary Programming (FEP), which is based on the Cauchy distribution function. Their comparative study of both mutation-based algorithms showed that solving high-dimensional optimization problems, the FEP converges quicker to minima than the CEP. Furthermore, Yao *et al.* [17] have developed an improved strategy of FEP, called the improved Fast Evolutionary Programming (IFEP).

K. Davoian and W.-M. Lippe are with the Department of Computer Science, University of Münster, Einsteinstr. 62, 48149 Münster, Germany (phone: +49-2518333796, fax: +49-2518333755, e-mail: kristina.davoian@uni-muenster.de, lippe@math.uni-muenster.de).

Detailed analysis of existing approaches, presented in [18] showed that they utilize only genotype information, i.e. depend on a total number of connection weights between neurons. We have assumed, that the efficiency of a mutation step size may depend not only on the genotype, but also on the phenotype (according to the definition given by [1, 2] the actual ANN architecture (a total number of neurons and layers) is called *phenotype* and connection weights (their values and a total number) represent *genotype*). In this paper we propose a new self-adaptive mutation strategy based on a uniform distribution for ANNs training. In comparison with existing EP strategies, it contains not only genotype, but includes also the phenotype information The phenotype information in the proposed strategy is incorporated in a value, called the *network weight (NW)*, which adapts the mutation operator to a given ANN architecture. The network weight value depends on the number of hidden layers and the average number of neurons in hidden layers and thus contains information about an ANN's "internal" structure. This relationship is determined by the Fermi-Dirac-like function.

The dynamic part of the proposed strategy is determined by the fitness function, which is defined by the individual's mean square error (MSE) between the expected and the actual outputs over all examples of a considered task. This involves a dynamic component in the mutation operator, which changes its value during run time and adjusts the mutation strength proportional to the fitness of a particular chromosome. These two components together (the NW and the MSE) encapsulate all necessary information about phenotype and genotype of a given problem, which enables to adapt the mutation strength according to the characteristics of ANN's "internal" architecture and fitness of a particular chromosome; this increases the efficiency of reproduction (percentage of successful mutations) and leads to a quick convergence to an optimum.

Thus, the evolution of ANN architectures and connection weights is driven by two components in the mutation algorithm: the MSE, which changes its value during run time and adjusts the mutation strength proportional to the fitness of a particular chromosome, and the NW, which increases the efficiency of reproduction by adapting the mutation step size according to the size of the given ANN topology.

We provided experimental study of a NW-based EP for the purpose of evaluating the performance of algorithm. In order to determine the speed of algorithm's work for ANNs of

different complexity, we considered predefined ANNs and evolved only the connection weights. The results were compared with those, obtained by the CEP (Gaussian mutation). To study the generalization ability of ANNs evolved by the NW-based EP strategy, the algorithm has been applied to the simultaneous evolution of connection weights and architectures and tested on the problem of predicting Mackey-Glass chaotic time series. The results of these simulations were compared with those, obtained by EPNet [19] and FNT [20] algorithms.

This paper is organized as follows: section II describes the NW-based EP. Section III and IV present and analyze the experimental results. Section V concludes this paper.

II. The NW-based Evolutionary Programming

Let us consider an initial population consisting of M randomly generated chromosomes. Each chromosome $S_i = (s_1, s_2, \ldots, s_m)$, $\forall\, i \in \{1, \ldots, M\}$, represents one possible set of connection weights, where m is a total number of connections between neurons and $s_j \in [-1.0, 1.0]$ for $j \in \{1, \ldots, m\}$. The fitness of a particular chromosome depends on the considered task and is assigned by a MSE. All chromosomes of parental population take part in new individuals' creation, i.e. expose to mutation. In the described algorithm each new chromosome is formed by applying mutation to one gene, randomly chosen out of a parental individual, which is changed by the following formula:

$$s'_j = s_j\left(1.0 + N_w(l,n) N_{dyn} N_{rand}\right) \tag{1}$$

where s_j is a gene chosen out of a chromosome S, and mutated; N_{dyn} – the MSE of S_i and N_{rand} is a uniformly distributed random value ($N_{rand} \in [-1.0, 1.0]$). The NW value $N_w(l, n)$ is defined by the function (2), which depends on the number of hidden layers l and the average number of neurons on hidden layers n. For each ANN the quantity N_w is calculated only once and does not change its value during the evolution, if we consider the evolution of connection weights in the environment determined by an ANN architecture; in case of simultaneous evolution of architectures and connection weights it becomes a new value every time when ANN's architecture is changed. We provided extensive empirical tests, considering predefined ANNs (in order to find the best values of the NW with high precision, which significantly increase the mutation step size), and studied their dependency on the ANN architectures. The tests were provided for ANNs with 1-5 hidden layers and 2-6 neurons on each hidden layer, i.e. the simplest ANN had 1 hidden layer with 2 neurons and the most complex ANN had 5 hidden layers with 6 neurons on each layer. The following eight global minimisation problems, cited by [15], have been used to determine NW values: high-dimensional unimodal f_1, f_2 and multimodal f_9, f_{10} functions (dimension 30); and low-dimensional functions f_{15} (dimension 4), f_{16}, f_{17} and f_{18} (dimension 2) with only a few local minima. We considered eight functions of different complexity (the most difficult are multimodal functions where the number of local minima increases exponentially with the problem dimension) with the purpose of obtaining the generalised NW values, which increase the average improvement of the population independent of requirements of any given task, i.e. do not contain knowledge of a problem. The empirical results showed that the distribution of the optimal NW values is similar for all considered functions and depends on the total number of hidden layers and the average number of neurons on hidden layers, i.e. NW is related only with ANNs' "internal" architecture. This relationship is defined by the Fermi-Dirac-like function and is calculated according to the following formula:

$$N_w(l,n) = A_1 + \frac{l}{2} + \frac{B_1 - \frac{l}{2}}{1 + \exp\left(\frac{n-\mu}{T_1}\right)} \tag{2}$$

The value μ has the same physical sense as a chemical potential in thermodynamics and depends on the number of hidden layers (the original Fermi-Dirac function and included in it chemical potential μ are cited by [21]):

$$\mu = A_2 + \frac{B_2}{1 + \exp\left(\frac{l - B_2}{T_2}\right)} \tag{3}$$

where the coefficients (constants) $A_1 = 3.0$, $B_1 = 2.0$, $T_1 = 0.4$, $A_2 = 1.2$, $B_2 = 3.2$, $T_2 = 0.6$ (the original Fermi-Dirac function depends on temperature T [21]).

As it is apparent from the equation (2), the behavior of its fractional part (right hand expression) is changed with the increment of hidden layers: up to 4 layers it becomes positive, and when the number of hidden layers is 4 it becomes zero (the exceptional case). This means, therefore, that the NW is independent of the average number of neurons and has optimal value 5.0. With more than 4 layers the fractional part of the expression (2) becomes negative. In spite of these characteristics of the equation (2), which are conditioned by the adjustment of the optimal mutation step size to a particular ANN architecture, the NW values are always positive.

The component N_{dyn} in equation (1) represents the genotype information, that is, the MSE which is dynamic component in the sense that it is different for every mutated chromosome. It enables the control of a randomly generated value and the adjustment of the mutation strength to an individual depending on its fitness, i.e. the higher the error of a chromosome, the higher the step size.

After mutation process the fitness values of parental and offspring chromosomes are compared and the best individual reaches the offspring population. The evolution cycle is repeated until certain halting criteria are satisfied.

III. The XOR Simulations

The first set of experiments was conducted in order to compare the NW-based EP and the CEP approaches. In these experiments we tested the algorithm on the simple problem – XOR function, and evolved the connection weights, considering the same ANN architectures as those used for the determination of the NW values (Section II). The initial population of chromosomes was randomly generated and consisted of 50 individuals. For each considered ANN, 1000 runs of the algorithm with NW values, according to the equation (2), were made. The terminating criterion was the precision of the best individual's mean square error equal to 1.0e-3.0To make a fair comparison, both algorithms started their evolution with the same initial conditions (size of a population, ANN structure, absence of other genetic operators). The purpose of this comparative analysis was to evaluate the performance of both mutation techniques for ANNs of different complexity. For different ANN architectures we provided 1000 runs of both algorithms and as a determined precision between the desired and actual outputs was reached, the algorithms stopped their work. As a quality measure the average number of iterations was used, at which the optimal solutions were found. In Fig. 1 the comparative results of both algorithms are presented. Table I presents the average time in *ms*, needed to find the optimal solutions.

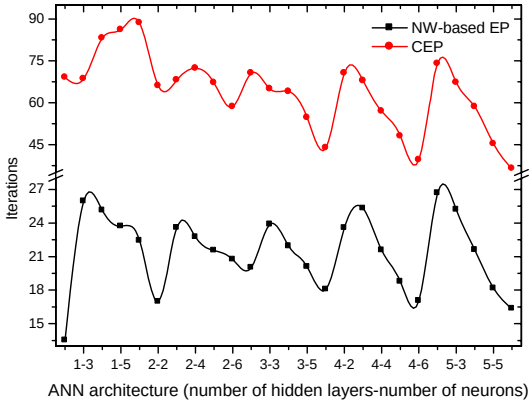

Fig. 1 Average number of iterations needed to find an optimal set of connection weights for NW-based EP and CEP

Simulations were also carried out to determine the percentage of successful mutations (percent of mutated chromosomes that reached the offspring population). The results showed that the average rate of successfully mutated chromosomes increases with the complexity of an ANN architecture. In comparison to other self-adaptive strategies, which have limited average improvement, determined by the 1/5 success rule [22] (the maximal rate of improved chromosomes), the proposed NW-strategy adjusts the percentage of successful mutations depending on the complexity of a given topology. In other words the rate of mutated chromosomes increases with the increment of hidden layers and neurons on them. This together with the optimal step size ensures fast population improvement. For the most complex ANN (with 5 hidden layers and 6 neurons per hidden layer) considered, approximately 24% of all chromosomes undergoing mutation improved their values at each stage of evolution.

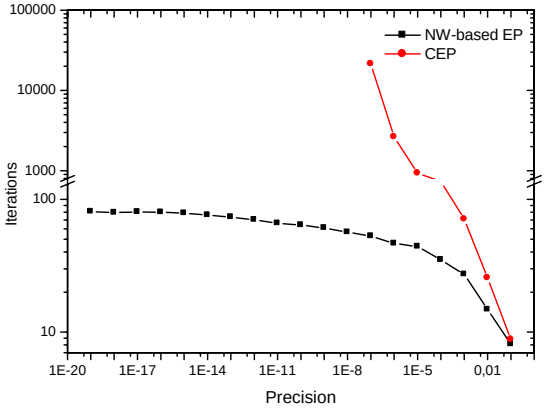

Fig. 2 Algorithms resistance to a premature convergence to a local optimum: ability to achieve a machine precision

TABLE I
AVERAGE TIME, NEEDED TO ALGORITHMS CONVERGENCE

ANN architecture		Average time, ms	
Hidden layers	Hidden neurons (average)	CEP	NW-based EP
1	2	8.8	1.7
1	3	12.6	4.8
1	4	19.4	5.9
1	5	29.1	8.0
1	6	40.9	10.4
2	2	10.3	2.6
2	3	17.8	6.2
2	4	24.7	7.8
2	5	34.3	11.0
2	6	39.1	13.8
3	2	13.5	3.8
3	3	22.0	8.1
3	4	28.2	9.7
3	5	36.0	13.2
3	6	40.6	16.7
4	2	15.0	5.0
4	3	27.2	10.1
4	4	30.8	11.7
4	5	40.6	15.8
4	6	45.7	19.7
5	2	18.5	6.6
5	3	33.1	12.4
5	4	36.8	13.5
5	5	45.7	18.3
5	6	50.9	22.8

In the described experiments above the algorithms stopped, when the MSE of the best chromosome in the population achieved the predefined precision equal to 1.0e-3.0 (the algorithm found a set of connection weights could solve the problem with the precision 1.0e-3.0). To study both

algorithms' ability to find solutions with the high precision (and thus to extend the search space and investigate algorithms resistance to convergence to a local instead of a global optimum) we provided a set of runs with the stopping condition equal to a machine precision. The tests showed that the NW-based algorithm is able to find solutions of better quality and is likely to be resistant to a premature convergence to local minima, while the self-adaptive Gaussian mutation is unable to find solutions with a machine precision. Fig. 2 presents an example of both algorithms' convergence for the ANN with 1 hidden layer and 3 neurons on it.

IV. The Mackey-Glass Chaotic Time Series Prediction

In the second set of experiments we applied the NW-mutation strategy to the simultaneous evolution of connection weights and architectures of ANN and tested it on the problem of predicting the Mackey-Glass chaotic time series [23], which is generated by the following differential equation:

$$\frac{dx}{dt} = \beta x(t) + \frac{\alpha x(t-\tau)}{1 + x^{10}(t-\tau)}$$

where $\alpha = 0.2$, $\beta = -0.1$, $\tau = 17$ (as mentioned by [19, 23], the system shows chaotic behavior when $\tau > 16.8$). The input consists of four variables $x(t)$, $x(t-6)$, $x(t-12)$ and $x(t-18)$, the predicting value is $x(t+6)$ and the time step is one. Fourth-order Runge-Kutta method with initial conditions $x(0) = 1.2$, $x(t-\tau) = 0$ for $0 \le t \le \tau$ was used to generate data for Mackey-Glass time series. 500 patterns (of point 118 to 617) were considered as training data, the following 500 samples being used as test data. The values of training and testing errors were rescaled linearly to between 0.1 and 0.9. The experiments were made in order to determine the best and the average prediction errors for small and large ANNs. In order to compare the results of NW-based algorithm with the existing approaches, we used the normalized root-mean-square (RMS) error to evaluate the performance of the NW-based mutation strategy. The RMS is determined by the RMS values of the absolute prediction error $\Delta t = 6$, divided by the standard deviation of $x(t)$.

$$error = \frac{\left\langle \left[x_{pred}(t, \Delta t) - x(t + \Delta t) \right]^2 \right\rangle^{\frac{1}{2}}}{\left\langle (x - \langle x \rangle)^2 \right\rangle^{\frac{1}{2}}}$$

As mentioned by [21], the prediction is perfect if RMS = 0; if RMS = 1 the prediction is not better than a constant predictor. The following parameters have been used in experiments to evolve ANNs: the population size 50, the maximum number of generations 150, the number of hidden nodes for each individual in the initial population was generated uniformly at random between 8 and 16, and the number of mutated hidden nodes 1. Two types of tests have been made for the purpose of 1) studying algorithm's generalization ability, i.e. ability to produce compact ANNs with low training and testing errors and 2) obtaining the smallest prediction error independent of ANN's complexity. The results of the first set of experiments were compared with those, obtained by the EPNet algorithm [19], which produces compact ANNs with sufficiently small prediction errors in comparison with BP (back-propagation) and CC (cascade-correlation) learning techniques, considered in [19]. Unfortunately, the EPNet does not provide the smallest prediction error of large networks (which usually have higher precision of prediction than small networks), since it searches a large space and requires long computation time [19]. The results of the second set of tests were compared with those evolved by the algorithm, based on the flexible neural tree model (FNT) [20] (though the comparison is not fair, since FNT algorithm uses adapted to this problem exponential instead of sigmoid function to transform neurons' input signals, which increases precision of prediction, but takes long computation time to achieve it). Table II shows the best and the average results of the NW-based strategy for first and second sets of tests over 50 runs of the algorithm. Fig. 3 demonstrates the work of the algorithm: the evolution of mean of the average normalized RMSE. Table III presents the generalization results comparison among the NW-based, EPNet and some other learning algorithms. Table IV presents the best training errors of large ANNs and the number of generations at which they were found for NW-based and FNT algorithms.

TABLE II
The Best and the Average Results produced by the NW-Based Strategy for the Mackey-Glass Chaotic Time Series Problem

	1st experiments	2nd experiments
Number of Connections	96	202
Aver. Num. of Connect.	98.21	202.5
Best Training Error	0.01203	0.00728
Aver. Training Error	0.01302	0.00742
Best Testing Error	0.01218	0.00735
Average Testing Error	0.01230	0.00751
Number of Generation	64	126
Aver. Num. of Gener.	78	144

TABLE III
Comparative Results of Algorithms for Small Networks for the Mackey-Glass Chaotic Time Series Problem

Algorithm	Number of connections	Testing Error
NW-based EP	96	0.01 (0.01218)
EPNet	103	0.02 (0.0152)
BP	540	0.02
CC Learning	693	0.06

TABLE IV
Best Prediction Errors of NW-Based and FNT Algorithms for the Mackey-Glass Chaotic Time Series Problem

Algorithm	Best RMSE (Training)	Generation
NW-based EP	0.00728	126
FNT	0.006901	135

The following observations can be made from the obtained results. First, they demonstrate that the NW-based algorithm evolves much more compact ANNs with smaller prediction

error than other considered algorithms and indicated insignificant distinctions between the best training and testing errors. Second, the NW-based strategy is able to produce ANNs with good generalization ability. The smallest ANN produced by NW-based strategy used 96 connections to achieve the smallest prediction error 0.01203, while the average number of connections was 98.21. This means, therefore, that the NW-based algorithm can achieve low error and compact ANN. Finally, the proposed algorithm was able to evolve ANNs with the comparable precision of prediction and showed quick convergence of evolving large networks – the best prediction error 0.00728 was indicated by the ANN with 202 connections; this result was achieved on the 126th generation.

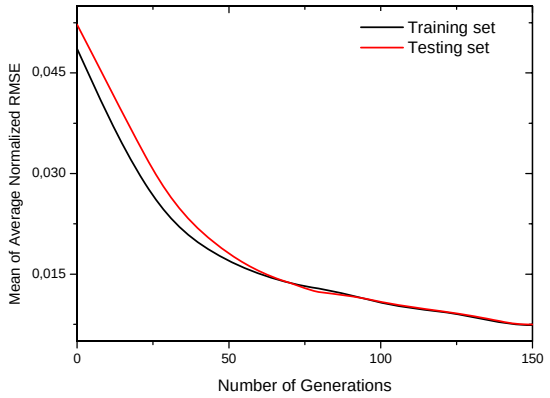

Fig. 3 Algorithm's work: the evolution of the mean RMSE

V. Conclusion

In this paper we described a mutation-based EP approach for evolving feed-forward ANNs, which allows the simultaneous adaptation of the mutation strength to an ANN's structure and a particular chromosome. In comparison with existing mutation algorithms it utilizes not only genotype information, but also phenotype information. Phenotype information is included in the novel self-adaptive control parameter called the *network weight (NW)* (see equation (2)), which depends on a total number of hidden layers and an average number of neurons on hidden layers, and thus explicitly describes an ANN's "internal" architecture. The experiments on 4 high- and 4 low-dimensional minimization functions (provided in Section II) showed that the NW value is defined by the Fermi-Dirac-like function (see equation (2)). Genotype information is represented by the MSE of every individual.

The comparative analysis to the classical EP (CEP) algorithm showed that the NW-based EP enables faster convergence to optimal solution as well as is more resistant to the premature convergence to local instead of global optima (Section III Figs. 1, 2, Table I). The results of second experiments showed that the proposed algorithm produces ANNs with good generalization ability: it was able to evolve much more compact ANNs and showed very competitive prediction errors for both large and small networks (Section IV, Tables II, III, IV).

By adapting the step size according to the information derived from the phenotype and the genotype, the NW-based EP strategy enables the algorithm to increase the mutation step size and percentage of successful mutations. This ensures a significant improvement of the population at each stage of evolution, which allows the finding of solutions of good quality and leads to a rapid convergence of the algorithm to optima. Another advantage of the described strategy is that it does not contain *a priori* knowledge of the problem domain and is independent of the ANN's "external" architecture – the number of input and output neurons, conditioned by a given task, which makes it widely applicable.

Acknowledgment

The authors would like to thank Alexander Reichel for his help with the mathematical representation of the NW values' dependency on ANN's "internal" structure (the Fermi-Dirac-like function). We would also like to thank Angelos Molfetas for the constructive comments and valuable discussions, which have helped to improve this manuscript.

References

[1] X. Yao, "Evolutionary artificial neural networks", in *Encyclopedia of Computer Science and Technology*, Vol. 33, New York: Marcel Dekker, pp. 137–170, 1995

[2] X. Yao, "Evolving Artificial Neural Networks", in *Proc. of the IEEE*, 87 (9), pp. 1423-1447, 1999

[3] D. B. Fogel, "Evolving Neural Networks: Selected Medical Applications and the Effects of Variation Operators", *Modeling and Simulation: Theory and Practice – A Memorial Volume for Professor Walter J. Karplus*, Kluwer Academic Press, Boston, MA, pp. 217-248, 2003

[4] D. G. Landavazo and G. B. Fogel, "Evolved Neural Networks for Quantitative Structure-Activity Relationships of Anti-HIV Compounds", in *Proc. of the IEEE Congress on Evolutionary Computation*, Vol. 1, Honolulu, HI, USA, pp. 199-204, 2002

[5] A. Abraham, "Meta-Learning Evolutionary Artificial Neural Networks", *Neurocomputing Journal*, Elsevier Science, Netherlands, Vol. 56c, pp. 1-38, 2004

[6] A. E. Eiben, R. Hinterding, Z. Michalewicz, "Parameter Control in Evolutionary Algorithms", *IEEE Trans. on Evolutionary Computation*, Vol. 3, pp. 124-141, 2000

[7] R. Hinterding, "Gaussian mutation and self-adaption for numeric genetic algorithms", in *Proc. of the Second IEEE Conference on Evolutionary Computation*, pp. 384-389, 1995

[8] A. Jain, D. Fogel. "Case studies in applying fitness distributions in evolutionary algorithms: I. Simple neural networks and Gaussian mutation", *Applications and Science of Computational Intelligence III*, *Proc. SPIE*, Vol. 4055, pp. 168-175, 2000

[9] P. A. Castillo, J. J. Merelo, V. Rivas, G. Romero, and A. Prieto, "Evolving Multilayer Perceptrons", *Neural Processing Letters* 12(2), pp.115-127, 2000

[10] J. Branke, "Evolutionary approaches to dynamic optimization problems – a survey", *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pp. 134-137, 1999

[11] D. B. Fogel and K. Chellapilla, "Revisiting evolutionary programming", in *SPIE AeroSense'98, Applications and Science of Computational Intelligence*, Orlando, FL, pp. 2-11, 1998

[12] W.-M. Lippe, "Soft-Computing mit Neuronalen Netzen, Fuzzy-Logic und Evolutionären Algorithmen", *Springer-Verlag*, Berlin Heidelberg, 2006

[13] H. Abbass and R. Sarker, "Simultaneous evolution of architectures and connection weights in anns", in *Artificial Neural Networks and Expert Systems Conference*, Dunedin, New Zealand, pp. 16-21, 2001
[14] Lock and C. Giraud-Carrier. "Evolutionary Programming of Near-Optimal Neural Networks", in *Proc. of the Fourth International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA99), Springer-Verlag*, pp. 302-306, 1999
[15] X. Yao and Y. Liu, "Fast Evolutionary Programming", in *Proc. of the Fifth Annual Conference on Evolutionary Programming (EP'96)*, the MIT Press, San Diego, CA, USA, 29/2-2/3/96. pp. 451-460, 1996
[16] X. Yao and Y. Liu, "Fast evolution strategies," *Control and Cybernetics*, 26(3), pp. 467-496, 1997
[17] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster", *IEEE Transactions on Evolutionary Computation*, pp. 82-102, 1999
[18] K. Davoian, A.Reichel, W.-M. Lippe, "Comparison and analysis of mutation-based evolutionary algorithms for ANN parameters optimization", in *Proc. of the 2006 International conference on Data Mining (DMIN'06),*CSREA Press, 2006
[19] X. Yao, Y. Liu, "A new evolutionary system for evolving artificial neural networks", *IEEE Transactions on Neural Networks*, 8(3): 694–713, May 1997
[20] Y. Chen, B. Yang, J. Dong, A. Abraham, "Time series forecasting using flexible neural tree model", *Information Sciences: an International Journal,* Vol 174, pp. 219-235, 2005
[21] W. Greiner, L. Neise, H. Stöcker, "Thermodynamics and statistical mechanics", *Springer-Verlag,* New York [u.a.] 2000
[22] I. Rechenberg, "Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution", *Fromman-Holzboog Verlag*, Stuttgart, Germany, 1973
[23] M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems", *Sci.*, vol. 197, p. 287, 1977