e-ISSN 1607-3274 Радіоелектроніка, інформатика, управління. 2019. № 4
p-ISSN 2313-688X Radio Electronics, Computer Science, Control. 2019. № 4

UDC 004.89

MODIFICATION AND PARALLELIZATION OF GENETIC ALGORITHM FOR SYNTHESIS OF ARTIFICIAL NEURAL NETWORKS

Leoshchenko S. D. – Postgraduate student of the Department of Software Tools, National University "Zaporizhzhia Polytechnic", Zaporizhzhia, Ukraine.

Oliinyk A. O. – PhD, Associate Professor, Associate Professor of the Department of Software Tools, National University "Zaporizhzhia Polytechnic", Zaporizhzhia, Ukraine.

Subbotin S. A. – Dr. Sc., Professor, Head of the Department of Software Tools, National University "Zaporizhzhia Polytechnic", Zaporizhzhia, Ukraine.

Lytvyn V. A. – Postgraduate student of the Department of Software Tools, National University "Zaporizhzhia Polytechnic", Zaporizhzhia, Ukraine.

Shkarupylo V. V. – PhD, Associate Professor, Associate Professor of Computer Systems and Networks National University of Life and Environmental Sciences, Ukraine.

ABSTRACT

Context. The problem of automation synthesis of artificial neural networks for further use in diagnosing, forecasting and pattern recognition is solved. The object of the study was the process of synthesis of ANN using a modified genetic algorithm.

Objective. The goals of the work are the reducing the synthesis time and improve the accuracy of the resulting neural network.

Method. The method of synthesis of artificial neural networks on the basis of the modified genetic algorithm which can be implementing sequentially and parallel using MIMD – and SIMD-systems is proposed. The use of a high probability of mutation can increase diversity within the population and prevent premature convergence of the method. The choice of a new best specimen, as opposed to a complete restart of the algorithm, significantly saves system resources and ensures the exit from the area of local extrema. The use of new criteria for adaptive selection of mutations, firstly, does not limit the number of hidden neurons, and, secondly, prevents the immeasurable increase in the network. The use of uniform crossover significantly increases the efficiency, as well as allows emulating other crossover operators without problems. Moreover, the use of uniform crossover increases the flexibility of the genetic algorithm. The parallel approach significantly reduces the number of iterations and significantly speedup the synthesis of artificial neural networks.

Results. The software which implements the proposed method of synthesis of artificial neural networks and allows to perform the synthesis of networks in sequentially and in parallel on the cores of the CPU or GPU.

Conclusions. The experiments have confirmed the efficiency of the proposed method of synthesis of artificial neural networks and allow us to recommend it for use in practice in the processing of data sets for further diagnosis, prediction or pattern recognition. Prospects for further research may consist in the introduction of the possibility of using genetic information of several parents to form a new individual and modification of synthesis methods for recurrent network architectures for big data processing.

KEYWORDS: data sample, synthesis, artificial neural network, genetic algorithm, neuroevolution, mutation.

ABBREVIATIONS

ANN is an artificial neural net;

EA is an evolutionary algorithm;

ESP is an enforced subpopulations;

MGA is a modified genetic algorithm;

NEAT is a neural evolution through augmenting topologies;

PMG is a parallel modified genetic algorithm;

RAM is a random access memory;

RV is a random value;

SANE is a symbiotic adaptive neuroevolution.

NOMENCLATURE

FB is a feedback connection between neurons;

$f_{comp.diff}$ is a criterion which characterizes the conditional complexity of the network;

f_{con} is a criterion which characterizes the degree of connectedness of neurons in the network;

$f_{fitness}$ is a fitness function of the individual;

$f_{top.diff}$ is a criterion which characterizes the complexity of the network;

G is a generation of the individuals;

g_{Ind} is a genes (genetic information) of the individual;

Ind is an individual from the population (generation);

N_h is a hidden neuron;

N_i is an input neuron;

N_o is an output neuron;

NN is a neural net or individual from the population (generation);

P is a population of the individuals (neural nets);

$p_{convergenœ}$ is a probability of early convergence of the method;

p_{mut} is a probability of mutation;

w is a connection between neurons.

INTRODUCTION

The choosing of topology and configuration the weights of connections of the ANN are the most important stages in the use of neural network technologies for solving practical problems [1–7]. From these stages de-

DOI 10.15588/1607-3274-2019-4-7

pends on the quality (adequacy) of the obtained neuronet models, control systems, etc.

Synthesis of ANNs by the traditional method is performed, in fact, through trial and error. The researcher sets the number of layers of neurons, as well as the structure of connections between them (the presence/absence of recurrent connections), and then analyzes the result. That is, ANNs is trained using any method, and then tested on a test sample. If the results of the synthesis meet the specified criteria, the task of building the ANN is considered to be completed successfully; otherwise, the process is repeated with other values of the output parameters [8–15].

Of course, the rapid development of the theory and practice of the using of genetic algorithms, forced researchers to look for ways to apply them to the problem of searching optimal structure of ANN (the evolution of neural networks or neuroevolution). This decision becomes even more logical if drawing a parallel with the real world, that is, if the idea of ANNs is borrowed from the nature, then the evolution of the nervous system with the subsequent formation and development of the brain is an example of solving such a problem [16, 17].

The object of study is the process of synthesis of ANN using a modified genetic algorithm.

For the decision of tasks of choosing the topology of ANN, and settings of the weights of all neurons and rigorous methods to date do not exist. The proposed solutions are aimed at solving local problems, through which the structure of the ANN is unsatisfactory, and the training time is large. In this case, it must to create a network and make calculations again. Even less attention is paid to the construction of multilayer asymmetric ANNs, characterized by complexity and multivariance.

The subject of study is the sequential and parallel method of synthesis of ANNs.

To date, there are several methods for the synthesis of ANNs, based on the use of evolutionary algorithms, however, it should be noted that most of these methods have similar disadvantages: a considerable time and a highlyiterative nature. Therefore, the paper proposes two approaches: a MGA and PMGA for the synthesis of ANNs.

The purpose of the work is to reduce the synthesis time and improve the accuracy of the resulting ANN. Additionally, determine the feasibility of using parallel implementation of MGA.

1 PROBLEM STATEMENT

The basis of ANNs are neurons with a structure similar to biological analogues. Each neuron can be represented as a microprocessor with several inputs and one output. When neurons are joined together, a structure is formed, which calls a neural network [18]. Vertically aligned neurons form layers: input, hidden and output. The number of layers determines the complexity and, at the same time, the functionality of the network, which is not fully investigated.

For researchers, the first stage of creating a network is the most difficult task. The following recommendations are given in the literature [10].

1) the number of neurons in the hidden layer is determined empirically, but in most cases used the rule $N_h \le N_i + N_o$;

2) increasing the number of inputs and outputs of the network leads to the need to increase the number of neurons in the hidden layer;

3) for the ANNs modeling multistage processes required additional hidden layer, but, on the other hand, the addition of hidden layers may lead to overwriting and the wrong decision at the output of the network.

Based on this, we present the problem as follows: for the synthesis of ANN (NN) it is necessary to determine the set of neurons $N = \{N_i, N_o, N_h\}$, what consists of subsets of input $N_i = \{N_{i_1}, N_{i_2}, ..., N_{i_n}\}, n = 1,2,...,|N_i|$, output $N_o = \{N_{o_1}, N_{o_2}, ..., N_{o_n}\}, n = 1,2,...,|N_o|$ and hidden neurons $N_h = \{N_{h_1}, N_{h_2}, ..., N_{h_n}\}, n = 1,2,...,|N_h|$ and a lot of weights of connections between neurons $w = \{w_i\}$. Having determined the values of the elements of the sets, we can consider the synthesis of ANN – complete.

2 REVIEW OF THE LITERATURE

The combination of ANNs and EA makes it possible to combine the flexibility of setting ANNs and adaptability EA, which allows to implement a largely unified approach to solving a wide range of problems of classification, approximation and modeling [19–29].

The first work on the use of EA for training and setting up ANN's, appeared about 20 years ago [30–31]. Research in this area is usually associated with the following tasks:

– searching for the values of weights of connections ANNs with a fixed structure;

– setting the structure of the ANN without first finding the weights of connections;

– setting the parameters of the training algorithm;

– setting parameters of neuronal activation functions;

– filtering training data;

– various combinations of the above tasks.

Neuroevolution approach for simultaneous solution of two main tasks of the synthesis of ANNs: setting the weights of connections and the structure of ANN, allows compensating to some extent the disadvantages, inherent in each of them separately and combining their advantages [32–35]. On the other hand, the price of it is a huge searching space, as well as combining a number of disadvantages caused by the using of the evolutionary approach. Summing up, we list the advantages and disadvantages.

The advantages include:

1) independence from the structure of ANN and characteristics of neuronal activation functions;

2) the ability to automatically search for the ANN topology and obtain a more accurate neural network model.

DOI 10.15588/1607-3274-2019-4-7

As noted, the synchronous solution of two problems avoids some difficulties. So the appearance of individuals in the population, which correspond to ANNs with different topologies, reduces the importance of the problem of competing solutions, and the availability of information about the weights of connections allows to bypass the problem of subjective assessment of the structure of ANN [33], due to the fact that the structure of the neural network is not estimated, but the entire ANN completely.

However, there are other disadvantages:

1) the complexity of fine-tuning the connections weights in the later stages of evolutionary search;

2) large, compared with gradient algorithms, the requirements for the amount of RAM through the use of the population of ANNs;

3) the complexity of the organization of search topology ANN.

Despite the fact that in most of the works devoted to the neuroevolutionary approach, only a theoretical approach to solving the problems of neural network optimization is proposed, several methods can be found that are recognized as promising and worthy of attention [32–39].

From the early works of noteworthy cellular Frederick Gruau method [40–42] uses a special grammar for the representation of neural network structures. One individual represented an entire neural network, with each neuron considered as a biological cell, and the growth of the network was determined through the mechanisms of sequential and parallel "division" of neurons i.e. cells. However, this method involves the implementation of a large number of specific operators that provide simulation of cell activity.

The SANE [43, 44] method uses a different approach. It is consider the development of two independent populations, one of which individuals are separate neurons, and the other contains information about the structures of an artificial neural network. The disadvantages of this method include the fact that the number of hidden neurons and connections is limited.

The ESP method [45, 46] is a development of the sane method. Its main difference is that the network structure is fixed and is given a priori. The population of neurons is divided into subpopulations, in each of which the evolution is independent. Due to parallelization of the solution search, as well as simplification of the problem due to the rejection of the evolution of the artificial neural network structure, ESP works much faster than SANE, sometimes by an order of magnitude, but for the successful operation of the method it is required to choose the appropriate structure of the neural network [47].

One of the most potentially successful attempts to get rid of the disadvantages of direct coding while preserving all its advantages is the method proposed in 2002, called NEAT [48, 49]. Designed by Kenneth Stanley, the NEAT method allows customizing the structure of the network, and without restrictions on its complexity. The solution proposed by the authors is based on the biological concept of homologous genes (alleles), as well as on the existence in nature of the synapsis process – the alignment of homologous genes before the crossover. The technique assumes that two genes (in two different individuals) are homologous if they are the result of the same mutation in the past. In other words, with each structural mutation (gene addition), a new gene is assigned a unique number, which then does not change during evolution. The method uses a number of techniques, such as historical labels and specialization of individuals, to make the process of evolution significantly more efficient [50].

Summing up, it can be noted that the joint use of evolutionary methods and artificial neural networks allows us to solve the problems of configuration and training of artificial neural networks both individually and simultaneously. One of the advantages of this synthesized approach is largely a unified approach to solving a variety of problems of classification, approximation, control and modeling. The use of qualitative evaluation of the functioning of artificial neural networks allows the use of neuroevolutionary methods to solve the problems of the study of adaptive behavior of intelligent agents, the search for game strategies, signal processing. Despite the fact that the number of problems and open questions concerning the development and application of neuroevolutionary methods (coding methods, genetic operators, methods of analysis, etc.) is large, often for the successful solution of the problem with the use of neuroevolutionary method adequate understanding of the problem and neuroevolutionary approach, as evidenced by a large number of interesting and successful works in this direction [33–36].

3 MATERIALS AND METHODS

The paper proposes a consistent implementation of MGA for the synthesis of ANN.

In the method, which is proposed to find a solution using a population of neural networks: $P = \{NN_1, NN_2, ..., NN_n\}$, that is, each individual is a separate ANN $Ind_i \to NN_i$ [51]. During initialization population divided into two halves, the genes $g_{Ind_i} = \{g_1, g_2, ..., g_n\}$ of the first half of the individuals is randomly assigned $g_{Ind_i} = \{g_1 = \text{Rand}, g_2 = \text{Rand}, ..., g_n = \text{Rand}\}$. Genes of the second half of the population are defined as the inversion of genes of the first half $g_{Ind_i} = \overline{\{g_1 = \text{Rand}, g_2 = \text{Rand}, ..., g_n = \text{Rand}\}}$. This allows for a uniform distribution of single and zero bits in the population to minimize the probability of early convergence of the method: $p_{convergenc\ e} \to \min$.

After initialization, all individuals have coded networks in their genes without N_h, and all N_i are connected to each N_o. That is, at first, all the presented ANNs differ only in the weights of the interneuron connection w_i. In the process of evaluation, based on the genetic information of the individual under consideration, a neural network is first built, and then its performance is checked, which determines the $f_{fitness}$ of the individual [51–53]. After evaluation, all individuals are sorted in order of reduced fitness, and a more successful half of the

DOI 10.15588/1607-3274-2019-4-7

sorted population is allowed to cross, with the best individual immediately moving to the next generation. In the process of reproduction, each individual is crossed with a randomly selected individual from among those selected for crossing. The resulting two descend-ants are added to the new generation: $G = P` = \{Ind_1, Ind_2, ..., Ind_n\}$. Once a new generation is formed the mutation operator starts working. However, it is important to note that the selection of the truncation significantly reduces the diversity within the population, leading to an early convergence of the algorithm, so the probability of mutation is chosen to be rather large: $p_{mut} = 15 - 25\%$ [51].

If the best individual in the population does not change within a certain number of generations (by default, it is proposed to set this number at seven), a new best individual is selected from the queue. This approach significantly saves time and resources of the system, in contrast to the complete restart of the method, but also allows implementing the exit from the area of local extrema due to the relief of the objective function, as well as a large degree of reliability of individuals in one generation.

It should be noted that the number of hidden neurons is theoretically unlimited. To regulate the size of the resulting networks, three criteria are used: the criteria for regulating the size and direction of development of the network, allowing at the stage of mutation to adaptively choose which type of structure transformation is more suitable for this network.

Obviously, the chosen method of coding requires special genetic operators that implement crossover and mutation.

The uniform crossover operator is one of the most efficient recombination operators in the standard genetic algorithm [54–56].

Uniform crossover is performed according to a randomly selected pattern that indicates which genes should be inherited from the first father (other genes are taken from the second parent). That is, the General rule of uniform crossing can be represented as follows:

$$\begin{aligned} &Crossover(Ind_1, Ind_2, DataofCros) = Ind_3 \\ &g_{Ind_3} = \{g_1 = Rand\left(g_{Ind_1}, g_{Ind_2}\right), \\ &g_2 = Rand\left(g_{Ind_1}, g_{Ind_2}\right), ..., \\ &g_i = Rand\left(g_{Ind_1}, g_{Ind_2}\right)\}. \end{aligned} \quad (1)$$

An example of a uniform crossover is shown in Fig. 1.

$$\begin{cases} parent\,1: [0011001110\ 10] \\ parent\,2: [1010110110\ 11] \end{cases} \xrightarrow{\text{crossover}} \begin{matrix} [1011011110\ \ 10]: child\ 1 \\ [0010100110\ \ 11]: child\ 2 \end{matrix}$$

locus :	1	2	3	4	5	6	7	8	9	10	11	12
etalon :	0	1	0	1	1	0	1	1	1	0	1	1

Figure 1 – Example of a uniform crossover

It has long been known that setting the probability of transmission of the parent gene to the offspring in uniform crossing can significantly improve its efficiency [54, 55], and also allows you to emulate other crossing operators (single-point, two-point). It is also known that the use of the operator of uniform crossover allows the use of the so-called multi-parents recombination, when more than two parents are used to generate one offspring. Despite this, most studies use only two parents and a fixed probability of gene transfer is 0.5 [54].

Uniform crossover gives more flexibility when combining strings, which is an important advantage when working with genetic algorithms.

When using the proposed method, such types of mutation operator can be used:

1) adding a hidden neuron with an index assignment $[N_h - 1]$. The new neuron is added along with the input and output connections. In this case, the output connection of the neuron can not bind it to the input neuron;

2) removal of a randomly selected hidden neuron along with all input and output connections. In this case, if a gap is formed in the remaining indices of neurons, the correction of indices in accordance with the above algorithm. The input and output neurons of the network cannot be removed;

3) adding a connection. Randomly determine the starting and ending indexes of the neurons in ANN submitted by mutating individual. In this case the connection can't end the input to the neuron. The link weight is also determined randomly with: $[N_h - 1]$. If the ins already has a connection with similar input and output neurons, its weight is replaced by a random;

4) delete a randomly selected connection. In this case, a situation may arise when the last connection in the hidden neuron is removed. In this case, the neuron is removed, and, if necessary, the correction of neuronal indices of the network;

5) changing the weight of a randomly selected connection to a random value from the range [–0.5; 0.5].

Thus, using mutations of points it is possible to change the parameters of the structure of ANN.

Chaotic addition (removal) of neurons and connections can lead to situations where, for example, the network has many neurons and few connections. More logical would be to use different types of mutations depending on the characteristics of the network architecture presented mutouch individual. For this purpose, three criteria were introduced that regulate the size and direction of the network development [57, 58].

The first of them characterizes the degree of connectedness [57] of neurons in the network and is calculated by the formula:

$$f_{con} = \frac{N_c}{2^{FB-1}\left[N_s(N_s-1) - N_i(N_i-1) - (1-FB)N_o(N_o-1)\right]}. \quad (2)$$

It is worth noting that connections from hidden neurons to the output can appear in any case. Thus, the smaller the more likely it is that a new connection will be added as a result of the mutation.

The use of the second coefficient is based on the assumption that the more elements in the sum of the input and output vectors of the training sample (the more the total number of input and output neurons), which is likely the more complex should be the ANN required to solve

DOI 10.15588/1607-3274-2019-4-7

the problem [57]. The second coefficient is calculated by the formula:

$$f_{top.diff} = \frac{N_i + N_o}{N_s}. \quad (3)$$

That is, the more neurons in the network, the lower the value of the criterion $f_{top.diff}$ and the less likely the mutation will be chosen, which adds a new hidden neuron.

The third criterion is also based on the assumption that a more complex network should be used to solve more complex problems. However, this criterion characterizes the conditional complexity of the network. This criterion is based on the concept of cyclomatic complexity [59], [60].

$$f_{comp.diff} = \frac{N_i + N_o}{N_s}. \quad (4)$$

For any of the described cases, a ligament $f_{con} \cdot f_{top.diff} \cdot f_{comp.diff}$ is used in the method, since the degree of connectedness of existing neurons must be taken into account for use.

Removing connections in ANN gives a side effect: may appear hanging neurons that have no incoming connections, as well as dead-end neurons, i.e. neurons without output connections. In cases where the function of neuronal activation is such that at zero weighted sum of inputs its value is not equal to zero, the presence of hanging neurons makes it possible to adjust the neural displacement. It is worth noting that, in addition to ensuring the diversity of the population, the removal of connections can contribute to the removal of some of the uninformative and lowinformative input features.

In the developed method, it is proposed to use an adaptive mutation mechanism [57, 59, 60], which provides for the choice of the mutation type depending on the values of the criteria f_{con}, $f_{top.diff}$ and $f_{comp.diff}$.

The choice of mutation type is determined based on the value of the multiplication $f_{con} \cdot f_{top.diff} \cdot f_{comp.diff}$. This approach on the one hand does not limit the number of hidden neurons, on the other, prevents the immeasurable increase in the network, because the addition of each new neuron in the network will be less likely. A mutation of the weight of a randomly existing bond occurs for all mutating individuals with a probability of 0.5.

Fig. 2 shows a schematic representation of the mutation type selection process.

Given the features of the proposed MGA synthesis of neural networks, its parallel form can be represented as in Fig. 3. All stages of the method can be divided into 3 stages, separated by points of barrier synchronization. At the first stage, the main core initializes the population P, and adjusts the initial parameters of the method, namely: the stopping criterion, the population size, the criteria for adaptive selection of mutations. Next, the distribution of equal parts of the population (subpopulations) and initial parameters to the cores of the computer system is performed. Initialization of the initial population cannot be carried out in parallel on the cores of the system, because the generated independent populations intersect thus increasing the search for solutions. The second stage of the proposed method is performed in parallel by the cores of the system. All cores perform the same sequence of operations on their initial population. After the barriersynchronization, the main core receives the best solutions

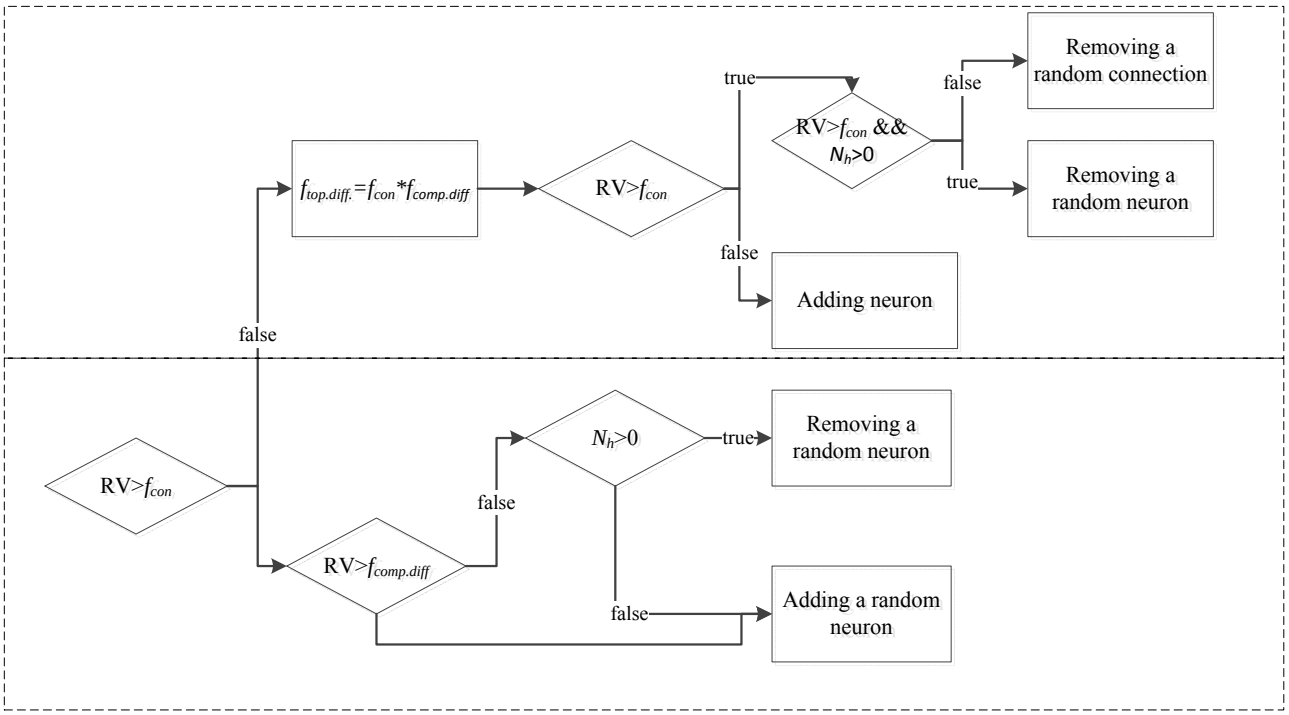

Figure 2 – The choice of the type of mutation

DOI 10.15588/1607-3274-2019-4-7

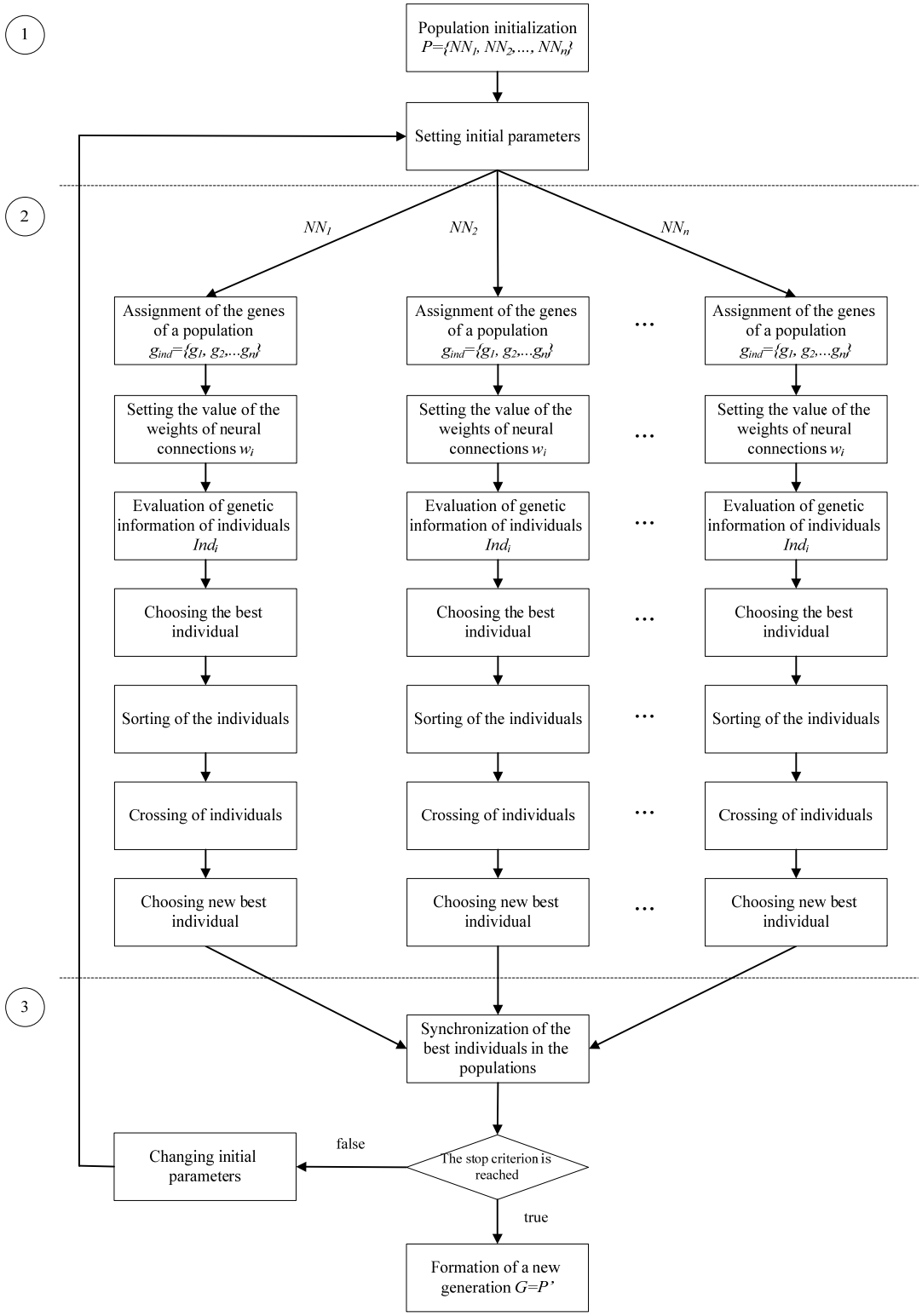

Figure 3 – Schematic representation of PMGA

from the other cores and checks the stopping criterion. If it is, then the next G is formed. Otherwise, after changing the initial parameters, allowing the cores of the system getting the other solutions, return to the distribution of the initial parameters to the cores on the system is performed. And then the cores perform parallel calculations according to the second stage of the method.

The proposed parallel method for ANN synthesis can be applied both on MIMD-systems [61] (clusters and supercomputers) and on SIMD (for example, GPU programmed with CUDA technology).

DOI 10.15588/1607-3274-2019-4-7

4 EXPERIMENTS

The proposed methods for MGA and PMGA were compared with existing analogues: ESP, SANE and NEAT.

Also note that testing the MGA, ESP, SANE and NEAT will occur using the following hardware and software: the computing system of the Department of software tools of National University "Zaporizhzhia Polytechnic" (NUZP), Zaporizhzhia: Xeon processor E5-2660 v4 (14 cores), RAM 4x16 GB DDR4, the programming model of Java threads.

The experimental verification of the proposed PMGA will additionally be performed with the additional use of the Nvidia GTX 960 GPU with 1024 cores, which are programmed using CUDA technology.

This testing technology will further compare the speed and performance of the PMGA on the MIMD-systems and SIMD.

For testing it used a training sample of Physical Unclonable Functions Data Set from the open repository UCI Machine Learning Repository [62, 63]. General information about the sample are given in table 1.

Table 1 – General information about data set

Criterion	Characteristic
Data Set Characteristics:	Multivariate
Attribute Characteristics:	Integer
Number of Instances:	6000000
Number of Attributes	129

For ANN training, 5 million instances were used, and testing of the resulting ANN occurred on 1 million instances from the sample.

5 RESULTS

Table 2 presents the overall results of the proposed MGA in comparison with the results of existing analogues. Particular attention is paid to the determination of the time needed for the synthesis of ANN, the value of the average error in training stage and the value of the average error when working in test mode.

Tables 3–5 show the results of testing PMGA using different hardware and using different number of CPU and GPU cores during operation.

Table 3 shows how the time spent on the synthesis of ANN changes when the number of CPU or GPU cores is increased.

Table 4 shows the speedup changes depending on the number of CPU or GPU cores used.

Table 5 shows the changing the communication overhead with the increase in used CPU or GPU cores.

For more clarity, the dependence of the speedup on the number of CPU cores used in the form of a diagram is shown in Fig. 4, for the GPU at the Fig. 5.

Fig. 6 shows the efficiency graph of NUZP computing systems when executing the proposed method.

Table 2 – General results of the testing

Method	Training time (time for synthesis ANN), s	Average error in training stage	Average error when in test mode
ESP	30923.87	2.72	2.88
SANE	53498.30	3.02	3.39
NEAT	99506.83	1.71	2.09
МГА	86132.26	1.01	1.47

Table 3 – Dependence the execution time of the proposed method to the number of involved cores

Using CPU					
Number of involved cores	1	2	4	8	16
Execution time	86132,26	46660,08	24587,93	13830,56	8053,90
Using GPU					
Number of involved cores	60	120	240	480	960
Execution time	86764,77	52492,69	33857,78	24039,03	20192,78

Table 4 –Dependence the speedup to the number of involved cores

Using CPU					
Number of involved cores	1	2	4	8	16
Speedup	1,00	1,85	3,50	6,23	10,69
Uding GPU					
Number of involved cores	60	120	240	480	960
Speedup	0,99	1,64	2,54	3,58	4,27

Table 5 – Dependence the communication overhead to the number of involved cores

Using CPU					
Number of involved cores	1	2	4	8	16
Communication overhead	0.00	0.08	0.14	0.24	0.41
Using GPU					
Number of involved cores	60	120	240	480	960
Communication overhead	0.16	0.21	0.29	0.42	0.68

DOI 10.15588/1607-3274-2019-4-7

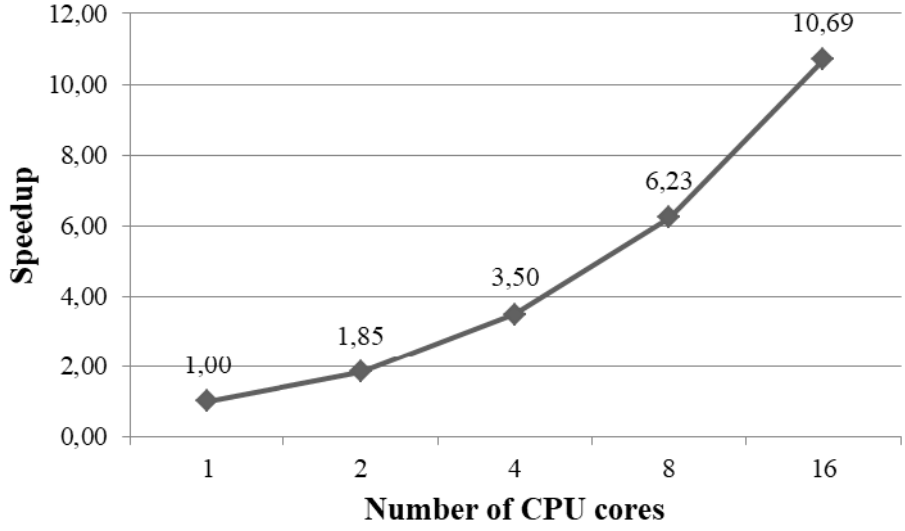

Figure 4 – The speedup graphics of calculations on CPU

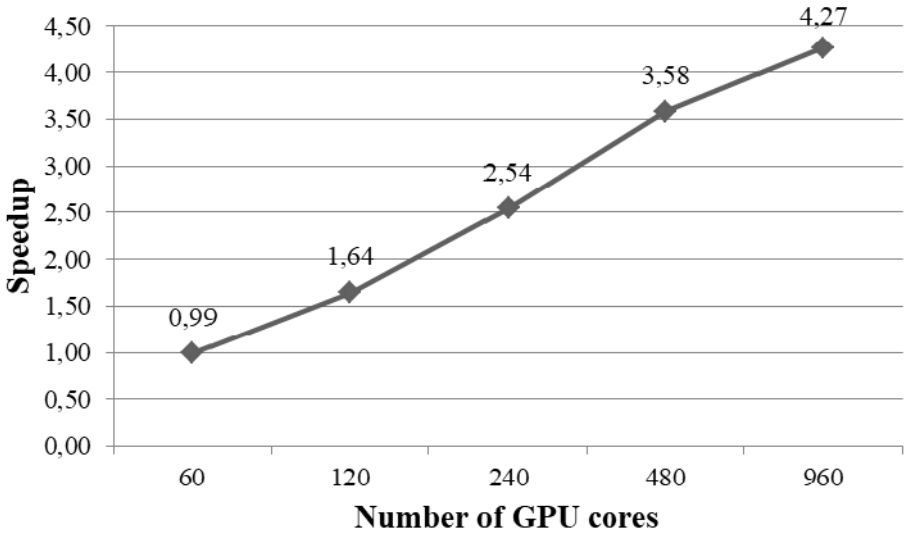

Figure 5 – The speedup graphics of calculations on a GPU

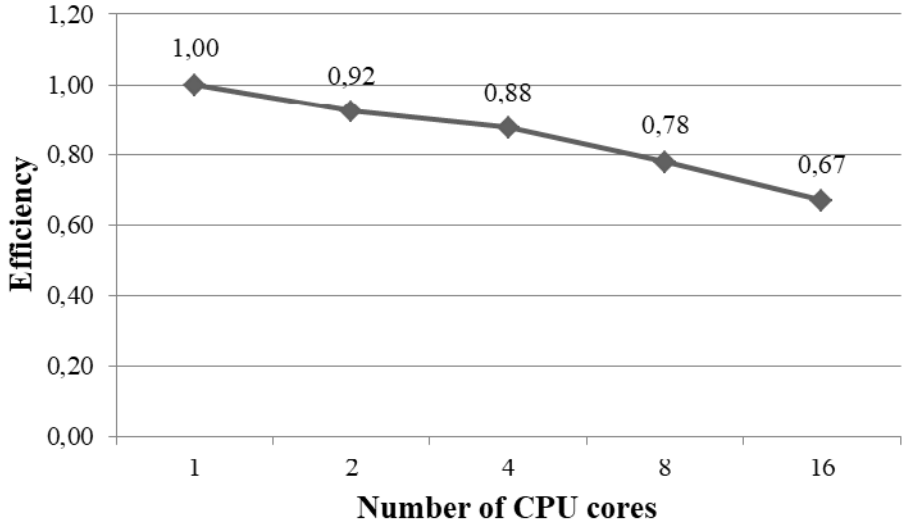

Figure 6 –The efficiency graph NUZP computing systems when executing the proposed method

DOI 10.15588/1607-3274-2019-4-7

6 DISCUSSION

As can be seen from the results in Table 1, consistent implementation MGA at the time of synthesis of ANN to the two existing analogues namely ESP and SANE, but is far ahead NEAT. If we compare the value of the average error in the synthesis of ANN, then using MGA it was possible to minimize it to 1.01%, which is significantly ahead of the results of analogues. It should also be noted that when testing is synthesized ANN in the case of MGA, the results are much better than analogs, so the average error value of the output of the ANN is 2.3 times less than, for example, in ANN synthesis by the method SANE. Therefore, it is possible to make a conclusion that the proposed MGA significantly exceeds the existing methods in the accuracy of the synthesized neural network.

As already noted, the testing of PMGA was carried out under a different scenario for a more complete study of the applicability and feasibility of the method on different parallel computing systems.

As can be seen from Table 3, the proposed method has an acceptable degree of parallelism and is effective on both MIMD-systems and SIMD. So on when using CPU cores it was possible to reduce execution time of a method from 86132.26 seconds (on one core) to acceptable 8053.90 seconds on 16 cores. However, it should be noted that when using a slightly different MIMD system, such as a cluster, there would be significant performance differences due to architectural features. In the cluster, the cores are connected using the InfiniBand Communicator, and in the multi-core computer they are located on the single chip, which explains the smaller impact of overhead (transfers and synchronizations). In addition, the processor model in a multi-core computer supports Turbo Boost [64] technology, so that the execution time of the method on one such core is much less than the execution time on the cluster core, which does not support such technology.

On the GPU with 960 cores involved in the execution time became 20192.78 seconds that can be adequately compared with the four cores of the computer.

From Table 4 and the graphs in Fig. 4 and 5 it can be seen that the speedup, though not linear, but approaches to linear. This is due to the fact that the share of overhead (Table 5) communication overhead execution of the proposed method in computer systems is relatively small, and the number of parallel operations significantly exceeds the number of consecutive operations and synchronizations. In communication overhead we understand the ratio of the time spent by the system on forwarding and synchronization between cores, in the time of target calculations on a given number of cores.

The graph of efficiency of computer systems NUZP is presented in Fig. 6. It shows that the using of even 16 cores of computer systems for the implementation of the proposed method retains the efficiency at a relatively acceptable level and indicates the potential, if necessary and possibly, to use even more cores.

Thus, the proposed method is well developed on modern computer architectures, which can significantly reduce the time of the task of synthesis of ANN. The parallel approach significantly increases the efficiency of sequential MGA and makes it even more acceptable for the synthesis of ANNs, through a significant reduction in time costs and maintaining high accuracy of the obtained neural networks.

CONCLUSIONS

The urgent problem of the synthesis of the ANNs using for diagnosis and future forecasting has been solved.

The scientific novelty lies in the fact that for the synthesis of ANNs is proposed to use a modification of the classical GA. So the input of the high probability of mutation allows to increase the diversity within the population and to prevent early convergence of the method. The choice of a new best individual, as opposed to a complete restart of the method, significantly saves system resources and ensures the exit from the area of local extrema. The use of new criterias for adaptive selection of mutations, firstly, does not limit the number of hidden neurons, and, secondly, prevents the immeasurable increase in the network. The use of uniform crossing significantly increases the efficiency, as well as allows you to emulate other crossover operators without problems. Moreover, the use of uniform crossover that increases GA flexibility. The parallel approach significantly reduces the number of iterations and significantly accelerates the synthesis of ANNs.

The practical significance of obtained results in the fact that the practical problems of synthesis of ANNs are solved, which can later be used for diagnosis and pattern recognition. The experimental results showed that the proposed synthesis methods allow to obtain accurate ANN based on the input data and can be used in practice to solve practical problems of diagnosis, prediction and pattern recognition.

Prospects for further research are the introduction of the possibility of using genetic information of several parents to form a new individual and modification of synthesis methods for recurrent ANNs architectures for big data processing [65–68].

ACKNOWLEDGEMENTS

The work is supported by the state budget scientific research project of National University University “Zaporizhzhia Polytechnic” “Intelligent methods and software for diagnostics and non-destructive quality control of military and civilian applications” (state registration number 0119U100360).

REFERENCES

1. Mocanu D., Mocanu E., Stone P., Nguyen P., Gibescu M., Liotta A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science, *Nature Communications*, 2018, No. 9, pp. 1–17. DOI: 10.1038/s41467-018-04316-3.

DOI 10.15588/1607-3274-2019-4-7

2. Goodfellow I., Bengio Y., Courville A. Deep Learning, Deep Learning. Massachusetts, MIT Press, 2016, 800 p. (Adaptive Computation and Machine Learning series).
3. Cao W., Wang X., Ming Z., Gao J. A review on neural networks with random weights, *Neurocomputing*, 2018, Vol. 275, pp. 278–287. DOI: 10.1016/j.neucom.2017.08.040.
4. Kieffer B., Babaie M., Kalra S. et al. Convolutional neural networks for his-topathology image classification: Training vs. using pre-trained networks, *Conference on Image Processing Theory, Tools and Applications (IPTA 2017) : Seventh International conference, 28–1 December 2017 : proceedings.* Montreal, IEEE, 2017, pp. 1–6. DOI: 10.1109/IPTA.2017.8310149.
5. Oliinyk A. Production rules extraction based on negative selection, *Radio Electronics, Computer Science, Control,* 2016, No. 1, pp. 40–49. DOI: 10.15588/1607-3274-2016-1-5.
6. Oliinyk A. A., Skrupsky S. Yu., Shkarupylo V. V., Subbotin S. A. The model for estimation of computer system used resources while extracting production rules based on parallel computations, *Radio Electronics, Computer Science, Control*, 2017, No. 1, pp. 142–152. DOI: 10.15588/1607-3274-2017-1-16.
7. Kolpakova T., Oliinyk A., Lovkin V. Integrated method of extraction, formalization and aggregation of competitive agents expert evaluations in group, *Radio Electronics, Computer Science, Control,* 2017, No. 2, pp. 100–108. DOI: 10.15588/1607-3274-2017-2-11.
8. Yadav J., Rani A., Singh V., Murari B. Prospects and limitations of non-invasive blood glucose monitoring using near-infrared spectroscopy, *Biomedical Signal Processing and Control,* 2015, Vol. 18, pp. 214–227. DOI: 10.1016/j.bspc.2015.01.005
9. Lymariev I. O., Subbotin S. A., Oliinyk A. A., Drokin I. V. Methods of large-scale signals transformation for diagnosis in neural network models, *Radio Electronics, Computer Science, Control*, 2018, No. 4, pp. 113–121. DOI: 10.15588/1607-3274-2018-4-11.
10. Smith-Miles K., Gupta J. Neural Networks in Business: Techniques and Applications for the Operations Researcher, *Computers and Operations Research*, 2000, Vol. 27, No. 11. pp. 1023–1044. DOI: 27. 10.1016/S0305-0548(99)00141-0.
11. Van Tuc N. Approximation contexts in addressing graph data structures. Wollongong, University of Wollongong Thesis Collection, 2015, 230 p.
12. Handa A., Patraucean V. Backpropagation in convolutional LSTMS, *University of Cambridge*, 2015, pp. 1–5.
13. Boden M. A guide to recurrent neural networks and backpropagation, *Halmstad University*, 2001, pp. 1–10.
14. Guo J. BackPropagation Through Time, The Harbin Institute of Technology, 2013, pp. 1–6.
15. Yue B., Fu J., Liang J. Residual Recurrent Neural Networks for Learning Sequential Representations, *Information,* 2018, Vol. 9, Issue 56, pp. 1–14.
16. Eluyodel O. S., Akomolafe D. T. Comparative study of biological and artificial neural networks, *European Journal of Applied Engineering and Scientific Research*, 2013, Vol. 2, No. 1, pp. 36–46.
17. Neural Networks: Is Your Brain Like A Computer? [Electronic resource]. Access mode: https://towardsdatascience.com/neural-networks-is-your-brain-like-a-computer-d76fb65824bf.
18. Jiang J., Trundle P., Ren J. Medical Imaging Analysis with Artificial Neural Networks, *Computerized medical imaging and graphics: the official journal of the Computerized Medical Imaging Society,* 2010, Vol. 34, No. 8, pp. 617–631.
19. Tahmasebi P., Hezarkhani A. A hybrid neural networks-fuzzy logic-genetic algorithm for grade estimation, *Computers & Geosciences*, 2012, Vol. 42, pp. 18–27.
20. Oliinyk A., Skrupsky S., Subbotin S., Blagodariov O., Gofman Ye. Parallel computing system resources planning for neuro-fuzzy models synthesis and big data processing, *Radio Electronics, Computer Science, Control,* 2016, No. 4, pp. 61–69. DOI: 10.15588/1607-3274-2016-4-8.
21. Oliinyk A. A., Subbotin S. A., Skrupsky S. Yu., Lovkin V. M., Zaiko T. A. Information Technology of Diagnosis Model Synthesis Based on Parallel Computing, *Radio Electronics, Computer Science, Control*, 2017, No. 3, pp. 139–151. DOI: 10.15588/1607-3274-2017-3-16.
22. Oliinyk A. A. Subbotin S., Lovkin V., Blagodariov O., Zaiko T. The System of Criteria for Feature Informativeness Estimation in Pattern Recognition, *Radio Electronics, Computer Science, Control,* 2017, No. 4, pp. 85–96. DOI: 10.15588/1607-3274-2017-4-10.
23. Oliinyk A., Subbotin S., Leoshchenko S., Ilyashenko M., Myronova N., Mastinovsky Y. Additional training of neuro-fuzzy diagnostic models, *Radio Electronics, Computer Science, Control,* 2018, No. 3, pp. 113–119. DOI: 10.15588/1607-3274-2018-3-12.
24. Ding S. Hui L., Su C., Yu J., Jin F. Evolutionary artificial neural networks: A review, *Artificial Intelligence Review*, 2013, Vol. 39, No. 3, pp. 251–260.
25. Castillo P. A., Arenas M. G., Castillo-Valdivieso J. J., Merelo J. J., Prieto A., Romero G. Artificial Neural Networks Design using Evolutionary Algorithms, *Advances in Soft Computing*, 2003, pp. 43–52.
26. Davoian K. Advancing evolution of artificial neural networks through behavioral adaptation. Münster, Universität Münster, 2011, 131 p.
27. Nissen V. A Brief Introduction to Evolutionary Algorithms from the Perspective of Management Science, *Innovative Research Methodologies in Management*, 2018, pp. 165–210.
28. Soltoggio A., Stanley K. O., Risi S. Born to learn: The inspiration, progress, and future of evolved plastic artificial neural networks, *Neural Networks,* 2018, Vol. 108, pp. 48–67. DOI: 10.1016/j.neunet.2018.07.013.
29. Molfetas A. Multiple control levels in structured genetic alogorithms. A thesis submitted for the degree of Doctor of Philosophy. Sydney, University of Western Sydney, 2006, 302 p.
30. AI 101: Intro to Evolutionary Algorithms [Electronic resource]. Access mode: https://www.ev olv.ai/blog/ai-101-intro-to-evolutionary-algorithms/
31. Angeline P. J., Saunders G. M., Pollack J. B. An evolutionary algorithm that constructs recurrent neural networks, *IEEE Transactions on Neural Networks,* 1994, Vol. 5, Issue 1, pp. 54–65.
32. Baxter J. The evolution of learning algorithms for artificial neural networks, Complex Systems, 1992, pp. 313–326.
33. Belew R., McInerney J., Schraudolph N. Evolving networks: using the genetic algorithm with connectionist learning, *CSE technical report CS90-174*, 1991, pp. 511–547.
34. Kearney W. T. Using Genetic Algorithms to Evolve Artificial Neural Networks, *Honors Theses,* 2016, pp. 1–21.
35. Shabash B., Wiese K. EvoNN: a customizable evolutionary neural network with heterogenous activation functions, *Tne Genetic and Evolutionary Computation Conference Companion : A Recombination of the 27th International Confer-*

DOI 10.15588/1607-3274-2019-4-7

ence on Genetic Algorithms (ICGA) and the 23rd Annual Genetic Programming Conference (GP), 15–19 July 2018 : proceedings. New York, ACM, 2018, pp. 1449–1456. DOI: 10.1145/3205651.3208282.
36. Rempis C., Pasemann F. An Interactively Constrained Neuro-Evolution Approach for Behavior Control of Complex Robots, *Variants of Evolutionary Algorithms for Real-World Applications*, 2012, pp. 305–341.
37. Baldominos A., Sáez Y., Isasi P. On the Automated, Evolutionary Design of Neural Networks-Past, Present, and Future, *Neural Computing and Applications*, 2019, pp. 1–45. DOI: 10.1007/s00521-019-04160-6.
38. Baldominos A., Sáez Y., Isasi P. Evolutionary Design of Convolutional Neural Networks for Human Activity Recognition in Sensor-Rich Environments, *Sensors,* 2018, Vol. 18, No. 4, pp. 1–24. DOI:10.3390/s18041288.
39. Schrum J., Miikkulainen R. Solving Multiple Isolated, Interleaved, and Blended Tasks through Modular Neuroevolution, *Evolutionary computation*, 2016, Vol. 24, № 3, pp. 1–29. DOI: 10.1162/EVCO_a_00181.
40. Gruau F. Genetic synthesis of Boolean neural networks with a cell rewriting developmental process, COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks, 1992, pp. 55–74.
41. Gruau F., Whitley D. Adding Learning to the Cellular Development of Neural Networks: Evolution and the Baldwin Effect, *Evolutionary Computation*, 1993, Vol. 1, No. 3, pp. 213–233. DOI: 10.1162/evco.1993.1.3.213.
42. Bayer J., Wierstra D., Togelius J., Schmidhuber J. Evolving Memory Cell Structures for Sequence Learning, *Artificial Neural Networks, ICANN 2009. ICANN 2009. Lecture Notes in Computer Science*, 2009, Vol. 5769, pp. 755–764.
43. Moriarty D., David R., Miikkulainen R. Hierarchical evolution of neural networks, *Evolutionary Computation Proceedings, IEEE International Conference, 4–9 May 1998. Anchorage, IEEE,* 1998, pp. 428–433. DOI: 10.1109/ICEC.1998.699793.
44. He H. Self-Adaptive Systems for Machine Intelligence. New York, Wiley-Interscience, 2011, 248 p.
45. Greer B., Hakonen H., Lahdelma R., Miikkulainen R. Numerical optimization with neuroevolution, *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600),* 2002, pp. 396–401. DOI: 10.1109/CEC.2002.1006267.
46. Braylan A., Hollenbeck M., Meyerson E. et al. Reuse of Neural Modules for General Video Game Playing, *Conference on Artificial Intelligence : Thirtieth AAAI conference on artificial intelligence, Phoenix, Arizona, 12–17 February 2016 : proceedings.* Arizona, AAAI 2016, pp. 353–359.
47. Enforced Subpopulations (ESP) neuroevolution algorithm for balancing inverted double pendulum [Electronic resource]. Access mode: http://blog.otoro.net/2015/03/10/esp-algorithm-for-double-pendulum.
48. Stanley K. O., Miikkulainen R. Evolving Neural Networks through Augmenting Topologies, *The MIT Press Journals*, 2002, Vol. 10, No. 2, pp. 99–127.
49. Manning T., Walsh P. P. Automatic Task Decomposition for the NeuroEvolution of Augmenting Topologies (NEAT) Algorithm, *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. EvoBIO 2012. Lecture Notes in Computer Science*, 2012, Vol. 7246. pp. 1–12.
50. Whiteson S., Whiteson D. Stochastic optimization for collision selection in high energy physics, *Conference on Artificial Intelligence : Twenty-second AAAI conference on artificial intelligence, Vancouver, British Columbia, 22–26 July 2007, proceedings.* California, AAAI Press, 2007, Vol. 2, pp. 1819–1825.
51. Tsoy Y. R. Evolutionary Algorithms Design: State of the Art and Future Perspectives, *IEEE East-West Design and Test Workshop (EWDTW'06)*, 2006, pp. 375–379.
52. Behjat A., Chidambaran S., Chowdhury S. Adaptive Genomic Evolution of Neural Network Topologies (AGENT) for State-to-Action Mapping in Autonomous Agents, *Conference on Robotics and Automation (ICRA' 2019), IEEE International Conference, 20–24 May 2019, proceedings*. Montreal, IEEE, 2019, pp. 1–7. arXiv:1903.07107.
53. Pagliuca P., Milano N., Nolfi S. Maximizing adaptive power in neuroevolution, *PLoS ONE,* 2018, Vol. 13, No. 7, pp. 1–27. DOI: 10.1371/journal.pone.0198788.
54. Petroski Such F., Madhavan V., Conti E., Lehman J., Stanley K., Clune J., Jeff Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning, *Technical report 1712.06567,* 2017, pp. 1–16. arXiv:1712.06567.
55. Capcarrere M., Freitas A. A., Bentley P. J. et al. Advances in Artificial Life, *Conference on Artificial Life, Eighth European Conference, 5–9 September 2005, proceedings.* Canterbury, Springer, 2005, 949 p.
56. McQuesten P. H. Cultural Enhancement of Neuroevolution. Austin, The University of Texas, 2002, 123 p.
57. Tsoy Y., Spitsyn V. Using genetic algorithm with adaptive mutation mechanism for neural networks design and training, *The 9th Russian-Korean International Symposium on Science and Technology, 2005*. KORUS, 2005, pp. 709–714.
58. Marzullo A., Stamile C., Terracina G., Calimeri F., Huffel S. A tensor-based mutation operator for Neuroevolution of Augmenting Topologies (NEAT), *IEEE Congress on Evolutionary Computation (CEC),* 2017, pp. 681–687. DOI: 10.1109/CEC.2017.7969376.
59. Papadimitriou F. Mathematical modelling of land use and landscape complexity with ultrametric topology, *Journal of Land Use Science*, 2011, Vol. 8, No. 2, pp. 1–21.
60. Papadimitriou F. Artificial Intelligence in modelling the complexity of Mediterranean landscape transformations, *Computers and Electronics in Agriculture,* 2012, Vol. 81, pp. 87–96.
61. Skillicorn D. Taxonomy for computer architectures, *Computer*, 1988, Vol. 21, No. 11, pp. 46–57.
62. Physical Unclonable Functions Data Set [Electronic resource]. Access mode: https://archive.ics.uci.edu/ml/datasets/Physical+Unclonable+Functions.
63. Aseeri A. O., Zhuang Y., Alkatheiri M. S. A Machine Learning-Based Security Vulnerability Study on XOR PUFs for Resource-Constraint Internet of Things / A. O. Aseeri, // *2018 IEEE International Congress on Internet of Things (ICIOT)*, 2018, pp. 49–56. DOI: 10.1109/ICIOT.2018.00014.
64. Intel Turbo Boost Technology 2.0 [Electronic resource]. – Access mode: https://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html.
65. Oliinyk A. A., Skrupsky S. Yu., Shkarupylo V. V., Blagodariov O. Parallel multiagent method of big data reduction for pattern recognition, *Radio Electronics, Computer Science, Control,* 2017, No. 2, pp. 82–92. DOI: 10.15588/1607-3274-2017-2-9.
66. Oliinyk A., Subbotin S., Lovkin V., Ilyashenko M., Blagodariov O. Parallel method of big data reduction based on stochastic programming approach, *Radio Electronics, Com-*

DOI 10.15588/1607-3274-2019-4-7

puter Science, Control, 2018, No. 2, pp. 60–72. DOI: 10.15588/1607-3274-2018-2-7
67. Omelianenko I. Performance evaluation of NEAT algorithm's implementation in GO programming language, 2018, pp. 1–6. DOI: 10.13140/RG.2.2.28676.83844.
68. Kadish D. Clustering sensory inputs using NeuroEvolution of Augmenting Topologies, *Tne Genetic and Evolutionary Computation Conference Companion : A Recombination of the 27th International Conference on Genetic Algorithms (ICGA) and the 23rd Annual Genetic Programming Conference (GP), 15–19 July 2018, proceedings.* New York, ACM, 2018, pp. 1–2. DOI: 10.1145/3205651.3205771.

Received 04.06.2019.
Accepted 12.09.2019.

УДК 004.89

МОДИФІКАЦІЯ ТА ПАРАЛЕЛІЗАЦІЯ ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ СИНТЕЗУ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

Леощенко С. Д. – аспірант кафедри програмних засобів Національного університету «Запорізька Політехніка», Запоріжжя, Україна

Олійник А. О. – канд. техн. наук, доцент, доцент кафедри програмних засобів Національного університету «Запорізька Політехніка», Запоріжжя, Україна.

Субботін С. О. – д-р техн. наук, професор, завідувач кафедри програмних засобів Національного університету «Запорізька Політехніка», Запоріжжя, Україна.

Литвин В. А. – аспірант кафедри програмних засобів Національного університету «Запорізька Політехніка», Запоріжжя, Україна.

Шкарупило В. В. – канд. техн. наук, доцент, доцент кафедри комп'ютерних систем і мереж Національного університету біоресурсів і природокористування України.

АННОТАЦІЯ

Актуальність. Вирішено задачу автоматизації синтезу штучних нейронних мереж для подальшого використання при діагностуванні, прогнозуванні та розпізнаванні образів. Об'єкт дослідження – процес синтезу штучних нейронних мереж за допомогою генетичного алгоритму. Предмет дослідження – послідовий та паралельний методи синтезу штучних нейронних мереж. Мета роботи – зменшити час синтезу та підвищити точність отриманої нейронної мережі.

Метод. Запропоновано метод синтезу штучних нейронних мереж на основі модифікованого генетичного алгоритму, який може бути реалізовано послідовно та паралельно використвуючі MIMD- та SIMD-системи. Введення великої ймовірності мутації дозволяє збільшити різноманітність всередині популяції та перешкодити завчасній збіжності методу. Вибір нової кращої особини, на відміну від повного перезапуску методу, значно економить ресурси системи та гарантує вихід із області локальних екстремумів. Використання нових критеріїв для адаптивного вибору мутації, по-перше, не обмежує кількість прихованих нейронів, а, по-друге, перешкоджає безмірному збільшенню мережі. Використання рівномірного схрещування істотно підвищує ефективність, а також без проблем дозволяє емулювати інші оператори схрещування. Більш того, саме використання рівномірного схрещування підвищує гнучкість генетичного алгоритму. Паралельний підхід значно скорочує кількість ітерацій та істотно прискорює виконання синтезу штучних нейронних мереж.

Результати. Розроблено програмне забезпечення, яке реалізує запропонований метод синтезу штучних нейронних мереж і дозволяє виконувати синтез мереж послідовно та паралельно на ядрах центрального процесору або графічного процесору.

Висновки. Проведені експерименти підтвердили працездатність запропонованого методу синтезу штучних нейронних мереж і дозволяють рекомендувати його для використання на практиці при обробці масивів даних для подальшого діагностування, прогнозування або розпізнавання образів. Перспективи подальших досліджень можуть полягати у введені можливості використання генетичної інформації декількох батьків для формування нової особини та модифікуванні методів синтезу для мереж рекурентних архітектур для обробки великих даних.

КЛЮЧОВІ СЛОВА: вибірка, синтез, штучна нейронна мережа, генетичний алгоритм, нейроеволюція, мутація.

УДК 004.89

МОДИФИКАЦИЯ И ПАРАЛЛЕЛИЗАЦИЯ ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ СИНТЕЗА ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Леощенко С. Д. – аспирант кафедры программных средств Национального университета «Запорожская Политехника», Запорожье Украина.

Олейник А. А. – канд. техн. наук, доцент, доцент кафедры программных средств Национального университета «Запорожская Политехника», Запорожье Украина.

Субботин С. А. – д-р техн. наук, профессор, заведующий кафедрой программных средств Национального университета «Запорожская Политехника», Запорожье Украина.

Литвин В. М. – аспирант кафедры программных средств Национального университета «Запорожская Политехника», Запорожье Украина.

Шкарупило В. В. – канд. техн. наук, доцент, доцент, кафедра компьютерных систем и сетей, Национальный университет биоресурсов и природопользования Украины

АННОТАЦИЯ

Актуальность. Решена задача автоматизации синтеза искусственных нейронных сетей для дальнейшего использования при диагностировании, прогнозировании и распознавании образов. Объект исследования – процесс синтеза искусственных

DOI 10.15588/1607-3274-2019-4-7

нейронных сетей с помощью генетического алгоритма. Предмет исследования – последовый и параллельный методы синтеза искусственных нейронных сетей. Цель работы – уменьшить время синтеза и повысить точность полученной нейронной сети.

Метод. Предложен метод синтеза искусственных нейронных сетей на основе модифицированного генетического алгоритма, который может быть реализован последовательно и параллельно используя MIMD- и SIMD-системы. Введение большой вероятности мутации позволяет увеличить разнообразие внутри популяции и предупредить преждевременную сходимость метода. Выбор новой лучшей особи, в отличие от полного перезапуска алгоритма, значительно экономит ресурсы системы и гарантирует выход из области локальных экстремумов. Использование новых критериев для адаптивного выбора мутации, во-первых, не ограничивает количество скрытых нейронов, а, во-вторых, препятствует безмерном увеличению сети. Использование равномерного скрещивания существенно повышает эффективность, а также без проблем позволяет эмулировать другие операторы скрещивания. Более того, именно использование равномерного скрещивания повышает гибкость генетического алгоритма. Параллельный подход значительно сокращает количество итераций и существенно ускоряет синтез искусственных нейронных сетей.

Результаты. Разработано программное обеспечение, реализующее предложенный метод синтеза искусственных нейронных сетей и позволяет выполнять синтез сетей последовательно и параллельно на ядрах центрального процессора или графического процессора.

Выводы. Проведенные эксперименты подтвердили работоспособность предложенного метода синтеза искусственных нейронных сетей и позволяют рекомендовать его для использования на практике при обработке массивов данных для дальнейшего диагностирования, прогнозирования или распознавания образов. Перспективы дальнейших исследований могут состоять в введении возможности использования генетической информации нескольких родителей для формирования новой особи и модификации методов синтеза для рекуррентных архитектур сетей для обработки больших данных.

КЛЮЧЕВЫЕ СЛОВА: выборка, синтез, искусственная нейронная сеть, генетический алгоритм, нейроэволюция, мутация.

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Mocanu D. & Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science / [D. Mocanu, E. Mocanu, P. Stone et al.] // Nature Communications. – 2018. – № 9. – P. 1–17. DOI: 10.1038/s41467-018-04316-3.
2. Goodfellow I. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville // Deep Learning. – Massachusetts : MIT Press, 2016. – 800 p. – (Adaptive Computation and Machine Learning series).
3. A review on neural networks with random weights / [W. Cao, X. Wang, Z. Ming, J. Gao] // Neurocomputing. – 2018. – Vol. 275. – P. 278–287. DOI: 10.1016/j.neucom.2017.08.040.
4. Convolutional neural networks for his-topathology image classification: Training vs. using pre-trained networks / [B. Kieffer, M. Babaie, S. Kalra et al.] // Conference on Image Processing Theory, Tools and Applications (IPTA 2017) : Seventh International conference, 28–1 December 2017 : proceedings. – Montreal: IEEE, 2017. – P. 1–6. DOI: 10.1109/IPTA.2017.8310149.
5. Oliinyk A. Production rules extraction based on negative selection / A. Oliinyk // Radio Electronics, Computer Science, Control. – 2016. – № 1. – P. 40–49. DOI: 10.15588/1607-3274-2016-1-5.
6. The model for estimation of computer system used resources while extracting production rules based on parallel computations / [A. A. Oliinyk, S. Yu. Skrupsky, V. V. Shkarupylo, S. A. Subbotin] // Radio Electronics, Computer Science, Control. – 2017. – № 1. – С. 142–152. DOI: 10.15588/1607-3274-2017-1-16.
7. Kolpakova T. Integrated method of extraction, formalization and aggregation of competitive agents expert evaluations in group / T. Kolpakova A. Oliinyk, V. Lovkin // Radio Electronics, Computer Science, Control. – 2017. – № 2. – P. 100–108. DOI: 10.15588/1607-3274-2017-2-11.
8. Prospects and limitations of non-invasive blood glucose monitoring using near-infrared spectroscopy / [J. Yadav, A. Rani, V. Singh, B. Murari] // Biomedical Signal Processing and Control. – 2015. – Vol. 18. – P. 214–227. DOI: 10.1016/j.bspc.2015.01.005
9. Methods of large-scale signals transformation for diagnosis in neural network models / [I. O. Lymariev, S. A. Subbotin, A. A. Oliinyk, I. V. Drokin] // Radio Electronics, Computer Science, Control. – 2018. – № 4. – P. 113–121. DOI: 10.15588/1607-3274-2018-4-11.
10. Smith-Miles K. Neural Networks in Business: Techniques and Applications for the Operations Researcher / K. Smith-Miles, J. Gupta // Computers and Operations Research. – 2000. – Vol. 27, № 11. – P. 1023–1044. DOI: 27. 10.1016/S0305-0548(99)00141-0.
11. Van Tuc, N. Approximation contexts in addressing graph data structures / N. Van Tuc. – Wollongong: University of Wollongong Thesis Collection, 2015. – 230 p.
12. Handa A. Backpropagation in convolutional LSTMS / A. Handa, V. Patraucean // University of Cambridge. – 2015. – P. 1–5.
13. Boden M. A guide to recurrent neural networks and backpropagation / M. Boden // Halmstad University. – 2001. – P. 1–10.
14. Guo J. BackPropagation Through Time / J. Guo // The Harbin Institute of Technology. – 2013. – P. 1–6.
15. Yue B. Residual Recurrent Neural Networks for Learning Sequential Representations / B. Yue, J. Fu, J. Liang // Information. – 2018. – Vol. 9, Issue 56. – P. 1–14.
16. Eluyodel O.S. Comparative study of biological and artificial neural networks / O. S. Eluyodel, D. T. Akomolafe // European Journal of Applied Engineering and Scientific Research. – 2013. – Vol. 2, № 1. – P. 36–46.
17. Neural Networks: Is Your Brain Like A Computer? [Electronic resource]. – Access mode: https://towardsdatascience.com/neural-networks-is-your-brain-like-a-computer-d76fb65824bf.
18. Jiang J. Medical Imaging Analysis with Artificial Neural Networks / J. Jiang, P. Trundle, J. Ren // Computerized medical imaging and graphics: the official journal of the Computerized Medical Imaging Society. – 2010. – Vol. 34, № 8. – P. 617–631.
19. Tahmasebi P. A hybrid neural networks-fuzzy logic-genetic algorithm for grade estimation / P. Tahmasebi, A. Hezarkhani // Computers & Geosciences. – 2012. – Vol. 42. – P. 18–27.

DOI 10.15588/1607-3274-2019-4-7

20. Oliinyk A. Parallel computing system resources planning for neuro-fuzzy models synthesis and big data processing / [A. Oliinyk, S. Skrupsky, S. Subbotin et al.] // Radio Electronics, Computer Science, Control. – 2016. – № 4. – P. 61–69. DOI: 10.15588/1607-3274-2016-4-8.
21. Information Technology of Diagnosis Model Synthesis Based on Parallel Computing / A. A. Oliinyk, S. A. Subbotin, S. Yu. Skrupsky et al] // Radio Electronics, Computer Science, Control. – 2017. – № 3. – P. 139–151. DOI: 10.15588/1607-3274-2017-3-16.
22. The System of Criteria for Feature Informativeness Estimation in Pattern Recognition / [A. Oliinyk, S. Subbotin, V. Lovkin et al] // Radio Electronics, Computer Science, Control. – 2017. – № 4. – P. 85–96. DOI: 10.15588/1607-3274-2017-4-10.
23. Additional training of neuro-fuzzy diagnostic models / [A. Oliinyk. S. Subbotin, S. Leoshchenko el al] // Radio Electronics, Computer Science, Control. – 2018. – № 3. – P. 113–119. DOI: 10.15588/1607-3274-2018-3-12.
24. Evolutionary artificial neural networks: A review / [S. Ding, L. Hui, C. Su et al.] // Artificial Intelligence Review. – 2013. – Vol. 39, № 3. – P. 251–260.
25. Artificial Neural Networks Design using Evolutionary Algorithms / [P. A. Castillo, M. G. Arenas, J. J. Castillo-Valdivieso et al.] // Advances in Soft Computing. – 2003. – P. 43–52.
26. Davoian K. Advancing evolution of artificial neural networks through behavioral adaptation / K. Davoian. – Münster: Universität Münster, 2011. – 131 p.
27. Nissen V. A Brief Introduction to Evolutionary Algorithms from the Perspective of Management Science / V. Nissen // Innovative Research Methodologies in Management. – 2018. – P. 165–210.
28. Soltoggio A. Born to learn: The inspiration, progress, and future of evolved plastic artificial neural networks / A. Soltoggio, K. O. Stanley, S. Risi // Neural Networks. – 2018. – Vol. 108. – P. 48–67. DOI: 10.1016/j.neunet.2018.07.013.
29. Molfetas A. Multiple control levels in structured genetic alogorithms. A thesis submitted for the degree of Doctor of Philosophy / A. Molfetas. – Sydney : University of Western Sydney, 2006. – 302 p.
30. AI 101: Intro to Evolutionary Algorithms [Electronic resource]. – Access mode: https://www.ev olv.ai/blog/ai-101-intro-to-evolutionary-algorithms/
31. Angeline P.J. An evolutionary algorithm that constructs recurrent neural networks / P.J. Angeline, G.M. Saunders, J.B. Pollack // IEEE Transactions on Neural Networks. – 1994. – Vol. 5, Issue 1. – P. 54–65.
32. Baxter J. The evolution of learning algorithms for artificial neural networks / J. Baxter // Complex Systems. – 1992. – P. 313–326.
33. Belew R. Evolving networks: using the genetic algorithm with connectionist learning / R. Belew, J. McInerney, N. Schraudolph // CSE technical report CS90-174. – 1991. – P. 511–547.
34. Kearney W. T. Using Genetic Algorithms to Evolve Artificial Neural Networks / W. T. Kearney // Honors Theses. – 2016. – P. 1–21.
35. Shabash B. EvoNN: a customizable evolutionary neural network with heterogenous activation functions / B. Shabash, K. Wiese // Tne Genetic and Evolutionary Computation Conference Companion : A Recombination of the 27th International Conference on Genetic Algorithms (ICGA) and the 23rd Annual Genetic Programming Conference (GP), 15–19 July 2018 : proceedings. – New York : ACM, 2018. – P. 1449–1456. DOI: 10.1145/3205651.3208282.
36. Rempis C. An Interactively Constrained Neuro-Evolution Approach for Behavior Control of Complex Robots / C. Rempis, F. Pasemann // Variants of Evolutionary Algorithms for Real-World Applications. – 2012. – P. 305–341.
37. Baldominos A. On the Automated, Evolutionary Design of Neural Networks-Past, Present, and Future / A. Baldominos, Y. Sáez, P. Isasi // Neural Computing and Applications. – 2019. – P. 1–45. DOI: 10.1007/s00521-019-04160-6.
38. Baldominos A. Evolutionary Design of Convolutional Neural Networks for Human Activity Recognition in Sensor-Rich Environments / A. Baldominos, Y. Sáez, P. Isasi // Sensors. – 2018. – Vol. 18, № 4. – P. 1–24. DOI: 10.3390/s18041288.
39. Schrum J. Solving Multiple Isolated, Interleaved, and Blended Tasks through Modular Neuroevolution / J. Schrum, R. Miikkulainen // Evolutionary computation. – 2016. – Vol. 24, № 3. – P. 1–29. DOI: 10.1162/EVCO_a_00181.
40. Gruau F. Genetic synthesis of Boolean neural networks with a cell rewriting developmental process / F. Gruau // COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks. – 1992. – P. 55–74.
41. Gruau F. Adding Learning to the Cellular Development of Neural Networks: Evolution and the Baldwin Effect / F. Gruau, D. Whitley // Evolutionary Computation. – 1993. – Vol. 1, № 3. – P. 213–233. DOI: 10.1162/evco.1993.1.3.213.
42. Evolving Memory Cell Structures for Sequence Learning / [J. Bayer, D. Wierstra, J. Togelius, J. Schmidhuber] // Artificial Neural Networks – ICANN 2009. ICANN 2009. Lecture Notes in Computer Science. – 2009. – Vol. 5769. – P. 755–764.
43. Moriarty D. Hierarchical evolution of neural networks / D. Moriarty, R. David, R. Miikkulainen // Evolutionary Computation Proceedings : IEEE International Conference, 4–9 May 1998. – Anchorage: IEEE, 1998. – P. 428–433. DOI: 10.1109/ICEC.1998.699793.
44. He H. Self-Adaptive Systems for Machine Intelligence / H. He. – New York : Wiley-Interscience, 2011. – 248 p.
45. Numerical optimization with neuroevolution / [B. Greer, H. Hakonen, R. Lahdelma, R. Miikkulainen] // Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600). – 2002. – P. 396–401. DOI: 10.1109/CEC.2002.1006267.
46. Reuse of Neural Modules for General Video Game Playing / [A. Braylan, M. Hollenbeck, E. Meyerson et al.] // Conference on Artificial Intelligence : Thirtieth AAAI conference on artificial intelligence, Phoenix, Arizona, 12–17 February 2016 : proceedings. – Arizona: AAAI 2016. – P. 353–359.
47. Enforced Subpopulations (ESP) neuroevolution algorithm for balancing inverted double pendulum [Electronic resource]. – Access mode: http://blog.otoro.net/2015/03/10/esp-algorithm-for-double-pendulum.
48. Stanley K. O. Evolving Neural Networks through Augmenting Topologies / K. O. Stanley, R. Miikkulainen // The MIT Press Journals. – 2002. – Vol. 10, № 2. – P. 99–127.
49. Manning T. Automatic Task Decomposition for the Neuro-Evolution of Augmenting Topologies (NEAT) Algorithm / T. Manning, P. Walsh P. // Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. EvoBIO

DOI 10.15588/1607-3274-2019-4-7

2012. Lecture Notes in Computer Science. – 2012. – Vol. 7246. – P. 1–12.
50. Whiteson S. Stochastic optimization for collision selection in high energy physics / S. Whiteson, D. Whiteson // Conference on Artificial Intelligence : Twenty-second AAAI conference on artificial intelligence, Vancouver, British Columbia, 22–26 July 2007 : proceedings. – California : AAAI Press, 2007. – Vol. 2. – P. 1819–1825.
51. Tsoy Y. R. Evolutionary Algorithms Design: State of the Art and Future Perspectives / Y. R. Tsoy // IEEE East-West Design and Test Workshop (EWDTW'06). – 2006. – P. 375–379.
52. Behjat A. Adaptive Genomic Evolution of Neural Network Topologies (AGENT) for State-to-Action Mapping in Autonomous Agents / A. Behjat, S. Chidambaran, S. Chowdhury // Conference on Robotics and Automation (ICRA' 2019) : IEEE International Conference, 20–24 May 2019 : proceedings. – Montreal : IEEE, 2019. – P. 1–7. arXiv:1903.07107.
53. Pagliuca P. Maximizing adaptive power in neuroevolution / P. Pagliuca, N. Milano, S. Nolfi // PLoS ONE. – 2018. – Vol. 13, № 7. – P. 1–27. DOI: 10.1371/journal.pone.0198788.
54. Petroski Such F. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning / [F. Such Petroski, V. Madhavan, E. Conti] // Technical report 1712.06567. – 2017. – P. 1–16. arXiv:1712.06567.
55. Advances in Artificial Life [M. Capcarrere, A. A. Freitas, P. J. Bentley et al.] // Conference on Artificial Life : Eighth European Conference, 5–9 September 2005 : proceedings. – Canterbury : Springer, 2005. – 949 p.
56. McQuesten P. H. Cultural Enhancement of Neuroevolution / P. H. McQuesten. – Austin: The University of Texas, 2002. – 123 p.
57. Tsoy Y. Using genetic algorithm with adaptive mutation mechanism for neural networks design and training / Y. Tsoy, V. Spitsyn // The 9th Russian-Korean International Symposium on Science and Technology, 2005. – KORUS, 2005. – 2005. – P. 709–714.
58. A tensor-based mutation operator for Neuroevolution of Augmenting Topologies (NEAT) / [A. Marzullo, C. Stamile, G. Terracina et al.] // IEEE Congress on Evolutionary Computation (CEC). –2017. – P. 681–687. DOI: 10.1109/CEC.2017.7969376.
59. Papadimitriou F. Mathematical modelling of land use and landscape complexity with ultrametric topology / F. Papadimitriou // Journal of Land Use Science. – 2011. – Vol. 8, № 2. – P. 1–21.
60. Papadimitriou F. Artificial Intelligence in modelling the complexity of Mediterranean landscape transformations / F. Papadimitriou // Computers and Electronics in Agriculture. – 2012. – Vol. 81. – P. 87–96.
61. Skillicorn D. Taxonomy for computer architectures / D. Skillicorn // Computer. – 1988. – Vol. 21, № 11. – P. 46–57.
62. Physical Unclonable Functions Data Set [Electronic resource]. – Access mode: https://archive.ics.uci.edu/ml/datasets/Physical+Unclonable+Functions.
63. Aseeri A. O. A Machine Learning-Based Security Vulnerability Study on XOR PUFs for Resource-Constraint Internet of Things / A. O. Aseeri, Y. Zhuang, M. S. Alkatheiri // 2018 IEEE International Congress on Internet of Things (ICIOT). – 2018. – P. 49–56. DOI: 10.1109/ICIOT.2018.00014.
64. Intel Turbo Boost Technology 2.0 [Electronic resource]. – Access mode: https://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html.
65. Parallel multiagent method of big data reduction for pattern recognition / [A. A. Oliinyk, S. Yu. Skrupsky, V. V. Shkarupylo, O. Blagodariov] // Radio Electronics, Computer Science, Control. – 2017. – № 2. – С. 82–92. DOI: 10.15588/1607-3274-2017-2-9.
66. Parallel method of big data reduction based on stochastic programming approach / [A. Oliinyk, S. Subbotin, V. Lovkin et al.] // Radio Electronics, Computer Science, Control. – 2018. – № 2. – P. 60–72. DOI: 10.15588/1607-3274-2018-2-7
67. Omelianenko I. Performance evaluation of NEAT algorithm's implementation in GO programming language / I. Omelianenko. – 2018. – P. 1–6. DOI: 10.13140/RG.2.2.28676.83844.
68. Kadish D. Clustering sensory inputs using NeuroEvolution of Augmenting Topologies / D. Kadish // Tne Genetic and Evolutionary Computation Conference Companion : A Recombination of the 27th International Conference on Genetic Algorithms (ICGA) and the 23rd Annual Genetic Programming Conference (GP), 15–19 July 2018 : proceedings. – New York : ACM, 2018. – P. 1–2. DOI: 10.1145/3205651.3205771.

DOI 10.15588/1607-3274-2019-4-7