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);
$Nh$ is a hidden neuron;
$Ni$ is an input neuron;
$No$ is an output neuron;
$NN$ is a neural net or individual from the population (generation);
$P$ is a population of the individuals (neural nets);
$p_{convergenc}$ 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-
The number of layers determines the complexity and, at formed, which calls a neural network [18]. Vertically output. When neurons are joined together, a structure is presented as a microprocessor with several inputs and one lar to biological analogues. Each neuron can be repre-

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

The subject of study is the sequential and parallel method of 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 simi-
lar to biological analogues. Each neuron can be repre-

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 $N_h \leq N_I + N_O$;

2) increasing the number of inputs and outputs of the network leads to the need to increase the number of neu-

3) for the ANNs modeling multistage processes re-quired 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 $w = \{w_{ij}\}$. 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 problem is the task of choosing the topology of ANN, and settings of the weights of all neurons and rig-

The advantage include:

1) independence from the structure of ANN and character-

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 de-

2) increasing the number of inputs and outputs of the network leads to the need to increase the number of neu-

3) for the ANNs modeling multistage processes re-

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 $w = \{w_{ij}\}$. 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 set-

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 $w = \{w_{ij}\}$. 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 set-

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 $w = \{w_{ij}\}$. 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 set-

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 $w = \{w_{ij}\}$. 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 set-

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 $w = \{w_{ij}\}$. 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 set-

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 $w = \{w_{ij}\}$. 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 set-

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 $w = \{w_{ij}\}$. Having determined the values of the elements of the sets, we can consider the synthesis of ANN – complete.
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 same 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 synopsis 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. \( Ind_i \rightarrow NN_i \) [51]. During initialization population divided into two halves, the genes of the first half of the individuals is randomly assigned

\[ g_{Ind_i} = \{g_1, g_2, ..., g_n\} \]

Genes of the second half of the population are defined as the inversion of genes of the first half

\[ g_{Ind_{i'}} = \{g_1 = \text{Rand}, g_2 = \text{Rand}, ..., g_n = \text{Rand}\} \]

This allows for a uniform distribution of single and zero bits in the population to minimize the probability of early convergence of the method: \( p_{\text{convergence}} \rightarrow \min \).

After initialization, all individuals have coded networks in their genes without \( N_{g_i} \), 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 \( f_{\text{fitness}} \) of the individual [51–53]. After evaluation, all individuals are sorted in order of reduced fitness, and a more successful half of the
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 crossover can be represented as follows:

\[
\begin{align*}
&\text{Crossover}(Ind_1, Ind_2, Data\text{of\_Cros}) = \text{Ind}_3 \\
&\text{g}_3 = |\text{g}_1| = \text{Rand}(|\text{g}_1|, |\text{g}_2|) \\
&\text{g}_2 = \text{Rand}(\text{g}_1, |\text{g}_2|) \\
&\text{g}_1 = \text{Rand}(\text{Ind}, g_\text{Ind}, |\text{g}_2|) \\
&\text{An example of a uniform crossover is shown in Fig. 1. (1)}
\end{align*}
\]

Figure 1 – Example of a uniform crossover

It has long been known that setting the probability of transmission of the parent gene to the offsping 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 operators 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:

\[
J_{con} = \frac{N_v}{2^{FB} - N_v(N_v - 1) - N_v(N_v - 1) - (1 - FB)N_v(N_v - 1)}
\]

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
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 \( f_{\text{top,diff}} \) and \( f_{\text{comp,diff}} \).

The choice of mutation type is determined based on the value of the multiplication \( f_{\text{con}} \cdot f_{\text{top,diff}} \cdot f_{\text{comp,diff}} \). 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 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).
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.

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.

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.

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

Table 1 – General information about data set

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Characteristics</td>
<td>Multivariate</td>
</tr>
<tr>
<td>Attribute Characteristics</td>
<td>Integer</td>
</tr>
<tr>
<td>Number of Instances:</td>
<td>6000000</td>
</tr>
<tr>
<td>Number of Attributes</td>
<td>129</td>
</tr>
</tbody>
</table>

Table 2 – General results of the testing

<table>
<thead>
<tr>
<th>Method</th>
<th>Training time (time for synthesis ANN), s</th>
<th>Average error in training stage</th>
<th>Average error when in test mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP</td>
<td>30923.87</td>
<td>2.72</td>
<td>2.88</td>
</tr>
<tr>
<td>SANE</td>
<td>53498.30</td>
<td>3.02</td>
<td>3.39</td>
</tr>
<tr>
<td>NEAT</td>
<td>99506.83</td>
<td>1.71</td>
<td>2.09</td>
</tr>
<tr>
<td>MGA</td>
<td>86132.26</td>
<td>1.01</td>
<td>1.47</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Using CPU</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of involved cores</td>
<td>60</td>
</tr>
<tr>
<td>Execution time</td>
<td>86132.26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Using GPU</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of involved cores</td>
<td>60</td>
</tr>
<tr>
<td>Execution time</td>
<td>86764.77</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Using CPU</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of involved cores</td>
<td>1</td>
</tr>
<tr>
<td>Speedup</td>
<td>1.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Using GPU</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of involved cores</td>
<td>60</td>
</tr>
<tr>
<td>Speedup</td>
<td>0.99</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Using CPU</th>
<th>Communication overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of involved cores</td>
<td>1</td>
</tr>
<tr>
<td>Communication overhead</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Using GPU</th>
<th>Communication overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of involved cores</td>
<td>60</td>
</tr>
<tr>
<td>Communication overhead</td>
<td>0.16</td>
</tr>
</tbody>
</table>
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

© Leoshchenko S.D., Oliinyk A. O., Subbotin S. A., Lytvyn V. A., Shkarupylo V. V., 2019
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 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 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


66. Olinyuk A., Subbotin S., Lovkin V., Ilyashenko M., Blagodariv O. Parallel method of big data reduction based on stochastic programming approach, Radio Electronics, Com-
МОДИФІКАЦІЯ ТА ПАРАЛЛЕЛІЗАЦІЯ ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ СИНТЕЗУ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

Леощенко С. Д. – аспіrant 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
Субботин С. О. – д-р техн. наук, професор, зав. кафедрою програмних засобів Національного університету «Запорізька Політехніка», Запоріжжя, Україна
Литвин В. А. – аспіrant кафедри програмних засобів Національного університету «Запорізька Політехніка», Запоріжжя, Україна
Шкапулило 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 – послідовний та паралельний методи синтезу штучних нейронних мереж. Мета роботи – змінити 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 використовуючи MIMD- та SIMD-системи. Введення великої ймовірності мутації дозволяє збільшити різноманітність всередині популяції та переховувати завдяки відповідного методу. Вибір нової кращої особини, на відміну від повного перезапуску методу, значно економить ресурси системи та гарантує відхід 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 змушувати 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 процесора або графічного процесора.

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 модифікувані методів синтезу для мереж рекурентних архітектур для обробки великих даних.

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

УДК 004.89

МОДИФІКАЦІЯ І ПАРАЛЛЕЛІЗАЦІЯ 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
Шкапулило 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 образов

© Leoshchenko S.D., Oliinyk A. O., Subbotin S. A., Lytvyn V. A., Shkarupylo V. V., 2019
DOI 10.15588/1607-3274-2019-4-7
79
нейронных сетей с помощью генетического алгоритма. Предмет исследования — последовательный и параллельный методы синте-
за 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 алго-
ритма, который может быть реализован последовательно и параллельно используя MIMD- и SIMD-системы. Введение большей вероятности мутации позволяет увеличить разнообразие внутри популяции и предупредить преждевременную сходимость метода. Выбор новой лучшей особи, в отличие от полного перезапуска алгоритма, значительно экономит ресурс-
сы системы и гарантирует выход из области локальных экстремумов. Использование новых критериев для адаптивного вы-
бора мутации, во-первых, не ограничивает количество скрытых нейронов, а, во-вторых, препятствует безмерному увеличе-
нию сети. Использование равномерного скрешивания существенно повышает эффективность, а также без проблем позволя-
ет имитировать другие операторы скрешивания. Более того, именно использование равномерного скрешивания повышает гибкость генетического алгоритма. Параллельный подход значительно сокращает количество итераций и существенно ус-
коряет синтез искусственных нейронных сетей.

Результаты. Разработано программное обеспечение, реализующее предложенный метод синтеза искусственных ней-
ронных 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 дальни-
нейшего диагностирования, прогнозирования или распознавания образов. Перспективы дальнейших исследований могут
составить в содержании возможности использования генетической информации нескольких родителей для формирования новой особы и модификации методов синтеза для рекуррентных архитектур сетей для обработки больших данных.

КЛЮЧЕВЫЕ СЛОВА: выборка, синтез, искусственная нейронная сеть, генетический алгоритм, нейрозволюция, мута-
ция.

ЛИТЕРАТУРА
15. Yue B. Residual Recurrent Neural Networks for Learning Sequential Representations / B. Yue, J. Fu, J. Liang // In-
17. Neural Networks: Is Your Brain Like A Computer? [Elec-


48. Stanley K. O. Evolving Neural Networks through Augmenting Topologies (NEAT) Algorithm / T. Manning, P. Walsh P. // Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. EvoBIO


