Evolving Artificial Neural Networks through L-System and Evolutionary Computation

Lídio Mauro Lima de Campos
Prog.de Pós-Graduação em Engenharia Elétrica
Universidade Federal do Pará-UFPA
Rua Augusto Corrêa 01-Guamá,
Belém, Pará CEP 66075-100, Brasil
Email: lidio@ufpa.br

Roberto Célio Limão de Oliveira
Prog.de Pós-Graduação em Engenharia Elétrica
Universidade Federal do Pará-UFPA
Rua Augusto Corrêa 01-Guamá,
Belém, Pará CEP 66075-100, Brasil
Email: limao@ufpa.br

Mauro Roisenberg
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina-UFSC
Florianópolis, SC, Brasil P.O. Box 476,
USFC-CTC-INE, Campus Trindade
Florianópolis, SC, Brasil
Email:mauro@inf.ufsc.br

***Abstract*—The aim of this paper is to present a biologically inspired Neuro Evolutive Algorithm (NEA) able to generate modular, hierarchical and recurrent neural structures as those often found in the nervous system of live beings, and that enable them to solve intricate survival problems. In our approach we consider that a nervous system design and organization is a constructive process carried out by genetic information encoded in DNA. Our NEA evolves Artificial Neural Networks (ANNs) using a Lindenmayer System with memory that implements the principles of organization, modularity, repetition (multiple use of the same sub-structure), hierarchy (recursive composition of sub-structures), as a metaphor for development of neurons and its connections. In our method, this basic neural codification is integrated to a Genetic Algorithm (GA) that implements the constructive approach found in the evolutionary process, making it closest to the biological ones. Our method was initially tested on a simple, but non-trivial, XOR problem. We also submit our method to two other problems of increasing complexity: Prediction of the effect of a new drug on Breast Cancer and KDDCUP'99 benchmark intrusion detection dataset. Some advantages of the proposed methodology are that it increases the level of implicit parallelism of the GA and seems to be capable of generating minimal satisfactory neural architectures, resulting in a reduction of project costs and increasing the performance of the evolved ANN, suggesting a promising potential for future applications.**

I. Introduction

Nowadays applications of Artificial Neural Networks (ANNs) vary from pattern recognition, control systems, financial forecast, medicine and agriculture and many others [20]. Designing the network architecture and training it are the most important problems with exploiting ANNs. However to obtain an optimized architecture in terms of the number of neurons and appropriate topology is a decision making process, which traditionally depends on the human experience. It involves a heuristic method of trial and error to select which kind of transfer function and which training algorithm should be used to adjust the synaptic weights. All these requirements affect the network's learning capacity and generalization, and usually needs to be performed by experts; they have to experiment on different topologies to find a suitable one, in order to solve a specific problem.

There are several optimization algorithms, and they're suited for different tasks. NEAs use Evolutionary Algorithms (EAs) in the ANN designing and/or training. GAs are population-based, stochastic search algorithms inspired by the natural evolution. Due GAs search globally and can handle infinitely large, non-differentiable and multimodal search space of architectures, the interest in research in this field has increased lately. As usual GAs are slow, but they are efficient to deal with large and complex search space. Moreover, in this paper we're investigating the possibilities of using biological inspiration in a local search problem.

The Bio-inspired Algorithms (BIOAs) have gained popularity due their adaptability in solving Real-world optimization problems [12] and have shown efficient in solve different nonlinear optimization problems [14],[15]. In the specific case of NEAs, (BIOAs) have become efficient tools to deal with large and complex search space(space of all feasible solutions) of that class of problem, in which conventional methods tend to fail.

According to the encoding method used, NEAs can be classified in two groups. At one extreme there are the methods that use direct encoding schemes (DES), for DES the chromossome contains the information about the parameters of the topology, architecture, learning parameters or weights of the ANN. At other extreme there are indirect encoding schemes (IES) in (IES) genotype specifies indirectly how that network should be generated. Indirect encodings are often used to achieve several aims: allow recurring structures or features in the network to form (modularity and other regularities), compression of phenotype to a smaller genotype (chromosome), the knowledge added to the algorithm reduces the search space and leads to a compact encoding.

One challenge of NEAs is to build an efficient indirect encoding scheme [4],[7],[8],[9],[22]. Which means that the evolutionary algorithm evolves a compressed description of the ANN rather than the ANN itself, such that information can be reused. Such compression allows the final solution to contain more components than its description [9]. These approaches are biologically motivated by the fact, that in case of the human brain, there are more neurons than nucleotides in the genome. Moreover, in biological genetic encoding the mapping between genotype and phenotype is indirect. The phenotype typically contains orders of magnitude more structural components than the genotype contains genes [8].

Nevertheless, neuroevolution has not so far addressed that

978-1-4799-1959-8/15/$31.00

the development and evolution of neurons is carried out by genetic information encoded in DNA and when followed will generate the final shape of the organs including the brain. Our new approach ADEANNs (Artificial Development and Evolution of ANNs) method provides a new perspective and shows that a direction to be followed in the neuroevolution is to evolve ANNs through developmental models encoded in the genome. Whereas the analogous biological process should be viewed as an organization process carried out by the genetic information encoded in DNA. This research is justified as it contributes to filling some gaps in some Neuro-Evolutives algorithms described in the next section state of the art.

The contributions of our system are: 1)ADEANN can utilize expert knowledge about the problem in order to more efficiently search in the infinite space of topologies, moreover invest a minimal of expert's effort for optimization. 2)Our approach increases the level of implicit parallelism of the GA, by reading the chromosome in several alternate positions, see (section III.B, Function Rule-Extraction-with-GA, step 6). 3)In our indirect encoding scheme, we have used biological inspiration as computational design techniques to solve a local search problem with optimal resource allocation (see section III.A, Dawkins' assumptions [2]). 4)Our IES leads to a compact encoding (section IV.B, Table V) of the proposed L-Rules (section III.A), which reduces the search space of all feasible solutions and the scalability. 5)Another contribution of our methodology is to propose a fitness function, equation (7), that explicitly rewards smaller ANNs that have better generalization capacity and also are easy to implement (see section III.D and section IV.B, table V). 6)Our L-System generates topologies of ANNs, without requiring additional configuration of substrates for each particular type of problem, as required by the methodologies [9], [16], our model automates this process.

The Contributions in relation to our previous methodology [8] are: 1)ADEANN can use chromosomes that are 35% shorter than the ones used in our previous research which reduces the search space (reduction from 512 to 180 bits). 2)We propose a generalized model of the Lindenmayer system that enables to generate more complex, direct, recurrent and multilayer networks (see section III.A). Our model of artificial development is more biologically inspired (see section III.A). 3)Determining appropriate coefficients for the fitness function of our previous GA is a problem dependent, time-consuming process that needs to try-and-test different values. The fitness function given by equation 7, automates this process and implements a penalty approach to simplifying ANNs generated through the evolution process (see section IV.A).

The paper is organized as follows. Section II discusses the State of the Art. In section III we introduce a biologically inspired methodology for automatic design of ANNs, in section IV experimental setup and results are shown and finally, conclusions are described in section V.

II. State of the Art

In this section we discuss some researches, in the field of NEAs, according to the encoding method used. The works can be followed in two major groups Direct and Indirect encoding schemes.

In Direct Encoding Scheme(DES) all of the network architecture details are encoded in a chromosome. Assuming the ANN as a direct graph and using an adjacent matrix for representing its genotype in a common direct encoding method [11]. The methodology uses EAs to evolve the architecture of the ANN, in other words all the network architecture details are encoded in a chromosome [11] and gradient-based search approaches may be used to optimize the weights of the neural network. Neuroevolution of Augmenting Topologies (NEAT) is a successful system in this area. Stanley et al. (2002) [6] uses a DES that incorporates a few aspects of biological plausibility. It alters both the weighting parameters and structures of networks, attempting to find a balance between the fitness of evolved solutions and their diversity.

Daniela Snchez et al. (2014) [10] proposed a new model of multi-objective optimization for hierarchical genetic algorithm (MOHGA) based on the micro-GA approach for modular neural networks (MNNs) optimization. This approach is used in iris recognition. The MOHGA divides the data automatically into granules or sub modules, and chooses which data are for the training and which are for the testing phase. The proposed method is responsible for determining the number of granules or sub modules and the percentage of data for training that can allow it to have better results. MOHGA was applied to human recognition and its applicability with good results is shown.

The other approach is Indirect Encoding Scheme (IES) where the genotype specifies indirectly how that network should be generated. Following we discuss several methods that use (IES) in so-called Generative and developmental systems (GDS). (GDS) is a subfield of evolutionary computation that focuses on evolving IES. In such representation the phenotype is 'grown' via a developmental process specified by genotype .

A hypercube based NEAT with an indirect encoding has also been introduced by Stanley et al. (2009) [9] compact genotype, input as well output scalability, and utilizing the geometry of the problem domaim are the most important advantages of their system. The ES-HyperNEAT [16] is an improvement over HyperNEAT [9]. While the location of the hidden nodes in the substrate, had to be decided by the designer on the original HyperNEAT. ES-HyperNEAT showed that the decision might in fact be automated. In the HyperNEAT, CPPN encodes an infinite number of weights within the Hypercube, from which a subset should be chosen to be incorporated into the ANN. Risi et al. (2012) [16] consider other important information is the density of nodes from which an additional increase of the same does not offer any advantage. ES-HyperNEAT [16] has shown promising results enabling the development of complex regular plastic ANNs.

The following methods belong to the class of GDS using grammatical evolution. [18] have used Grammar Evolution(GE) for design and train ANNs with one hidden layer. As stated by Tsoulos et al. (2008) [18] the use of grammar evolution has the benefit of allowing easy shaping of the resulting search, in addition to leading to a compact encoding offering key advantages in terms of scalability. These methods generate genotypes that are much smaller than the phenotypes they produce. Consequently, searching genotype space is more feasible.

The methodologies [7],[8] took a step towards those of biologically inspired approaches. Lee, Kong and Sim [7] sought inspiration from the DNA, in their research, information is coded using the symbols A, G, T and C. A sequence of three of these symbols is known as a codon. The sequence of codons between these delimiters is translated into a production rule for developing a neural controller. A similar idea [8] using binary strings also exists in which the process of reading and translating bits can be repeated from different bits in the string to produce different production rules. The NEA proposed by [8] has one disadvantage that is the difficulty to set the parameters of the fitness function of Genetic Algorithm to direct the search for economic architectures.

Soltanian et al. (2013) [3] have applied grammar evolution(GE) only for designing the network topology while its weights are optimized through backpropagation (BP). (BP) is used too for the networks evaluations against training data.

Ahmadizar et al.(2015) [22] have proposed a NEA that simultaneously evolve the topology and the connection weights of ANNs by means of a new combination of grammatical evolution (GE) and genetic algorithm (GA). GE is adopted to design the network topology while GA is incorporated for better weight adaptation. The proposed algorithm is capable of generating any feedforward ANN with one hidden layer. Moreover, due to the fact that the generalization ability of an ANN may decrease because of overfitting problems, the algorithm utilizes a novel adaptive penalty approach to simplify ANNs generated through the evolution process. As a result, it produces much simpler ANNs that have better generalization ability and are easy to implement. For more details about state of the art, query the authors' research [24].

III. BIOLOGICALLY INSPIRED NEA

This section goes on to describe and explore a new NEA. Below we described the decision-making process for our new approach. The Neuro-evolutive algorithm, Figure 1, starts with a population of production rules encoded in terms of 0's and 1's (a), after that the processes of transcription (b) and translation (c), of these bits on valid production rules are realized. These steps are performed by the Function Rule-Extraction-with-GA, presented in section III.A. The L-System, described in section III.B and illustrated in Table II, produces genotypic representations strings (d) that are mapped into ANNs architectures. The ANNs are trained (f) and the fitness is calculated using equation (6) or equation (7), stage (i). The ANNs are classified according the obtained performance (j) and the database is updated with training, test and validation data (g). The GA selects the best individuals for apply the genetic operators and a new population is obtained (k). Repeat the previous steps a number "n" of generations. Finally the reports of simulation are generated and statistical analyses are performed (h).

A. *Neural Structure Codification with L-System*

An important step in applying Evolutionary Algorithms (EA) to any type of problem is the designing of an appropriate genotype representation. Our approach is biologically motivated by the fact, that in case of the human brain, there are much more neurons than nucleotides in genome [8].

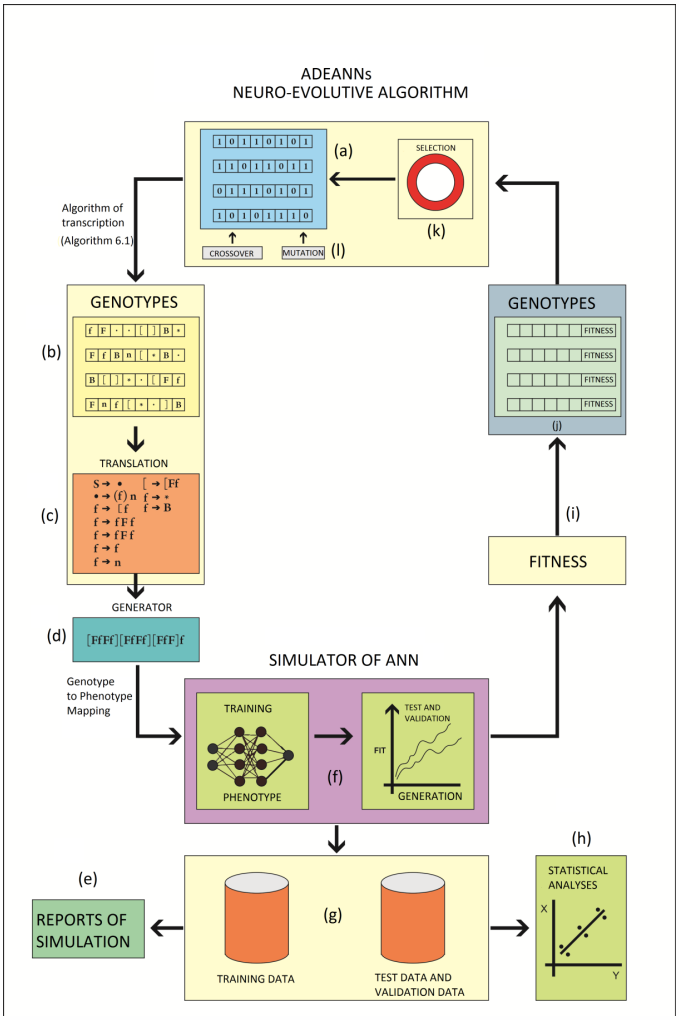

Fig. 1. Decision-Making process for the proposed approach (a,b,c,d,e,f,g,h,i,j,k)

Which simple and compact rules could the computer easily obey to generate complex topologies of ANNs?'. Dawkins [2] points out that one direction to be adopted in relation to genetic formula, is that the same would be recursive and with possibilities to generate the ramifications in an artificial development process. Thus the recursive branching could be a good metaphor for the Artificial Development of Neurons (ADN). Dawkins [2] also mentions that a nervous system design and organization is a constructive process carried out by genetic information encoded in DNA. Moreover, at biological level, the genes do two things: influence the development and are transmitted to subsequent generations. ADEANN have used these ideas as computational design techniques.

To mimic the mechanism of grown structures, among them the neurons, we use a grammar called parametric L-System with memory. The same can be described as a grammar $G = \{\Sigma, \Pi, \alpha\}$, where the alphabet is $\Sigma = \{.; f; F; n; [;]; *; B, \}$ the productions rules (Π) are described in Table I. The axiom (α = .) is the start point where the process of development begins, where "f" means drawing a provisional neuron and "F" draw a connection between neurons, "[" means storing the current state of the development and "]" recovering the

same state, the third rule is divided into six others.

Figure 2 illustrates a simple example of the construction

Rule Identifier	Rule
1,2	S→. (axiom) (2) .→(f...f)n
3	(3.1)f→[f (3.2)f→fF(w,l)f (3.3)f→fF(w,l) (3.4)f→n
3,4,5	(3.5)f→f (3.6)f→fB (4) [→[F(w,l)f] (5) f→f*

TABLE I. THE PRODUCTION RULES OF THE PARAMETRIC L-SYSTEM WITH MEMORY

process of one branch of a neural network. Supposing that starting with the axiom ($\alpha = .$), applying the second production rule ($. \rightarrow f$) to the axiom the string that results is **f**. Considering the third rule (3.1) ($f \rightarrow [f$) applied to the string **f**, obtained previously, the resulting string will be **[f**. Considering the fourth rule ($[\rightarrow [Ff]$) applied to the string **[f** three times, the resulting string will be **[Ff]f**, **[Ff]Ff]f** and **[Ff]Ff]Ff]f** respectively. That represents a branch with three neurons, which is shown in the last row and the third column of Figure 2, after the string **[Ff]Ff]Ff]** is stored for later use.

The rule (5) ($f \rightarrow f*$) that means ($f \rightarrow f[Ff]Ff]Ff]$) where * denotes recovery the string previously stored), is applied to the previous string **[Ff]Ff]Ff]f**, considering only the first **f**. At this stage we have used the principles of repetition (multiple use of the same substructure) and hierarchy (recursive composition of sub-structures). After that, the resulting string will be **[Ff[Ff]Ff]Ff]]Ff]Ff]f**, which represents the phenotype shown in the fourth row and third column of Figure 2, a new branch emerges starting from N3. The rule (3.2) is applied to the second **f** of the previous string the resulting string will be **[Ff[FfFf]Ff]Ff]Ff]f** the phenotype is shown in the third row and third column of Figure 2, a new neuron **N8** is created.

In the next iterations, the rules (3.2) and (5), Table I, are applied recursively and the development process continues. The Figure 2, also shows the mapping between genotype and phenotype. The same represents an ANN's architecture that can be described as a directed graph in which each neuron I performs a transfer function fi. Our indirect encoding method, presented in Section III.B enables generating ANNs with variable number of neurons in the range [X,Y], whose values X and Y are given by equations (1) and (2).

$$X = [Ninputs + Ninputs * random(NRmin) + Noutputs] \quad (1)$$

$$Y = [Ninputs + Ninputs * random(NRmax) + Noutputs] \quad (2)$$

Where: Ninputs and Noutputs are the number of neurons in the input and output layers of the ANN, NR is the number of neurons of each ramification.

Figure 3(a) Figure 3(a) illustrates in a dismembered way how an iterated ANN is generated in each stage of the development process. The algorithm starts by interactively finding the placement of hidden neurons from inputs and their connections. After the placement of other hidden neurons and their connections. Finally the placement of the output neurons from hidden neurons and their connections. Figure 3(b) illustrates the process of ANN's construction. The generated topologies in each stage of the development are shown in an aggregated manner. Where only those nodes are kept that have a path to an input and output neuron. The search through the

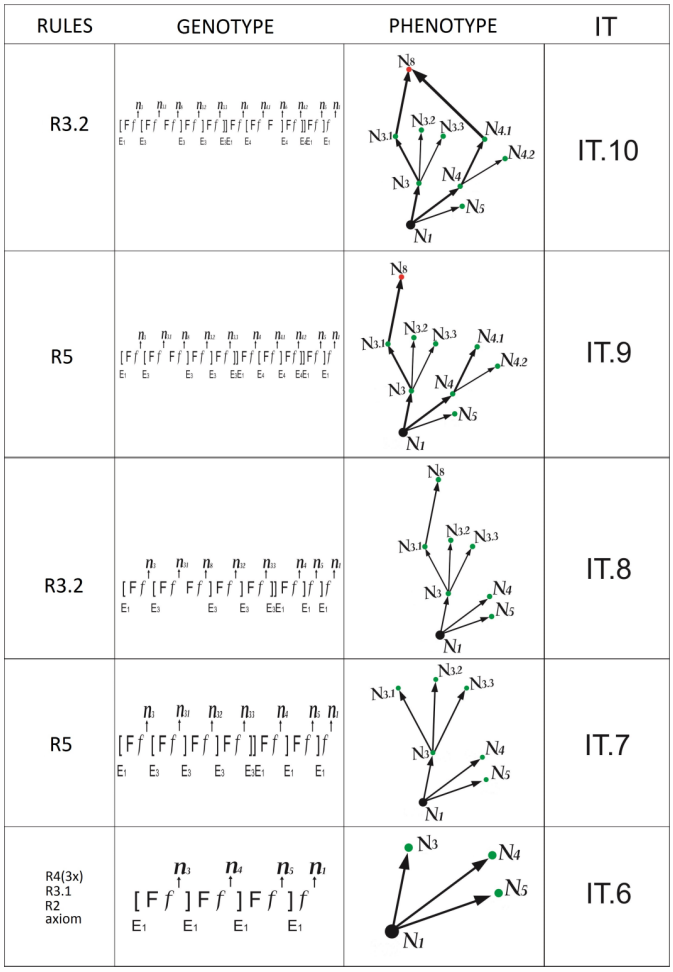

Fig. 2. Construction Process of one branch of an ANN

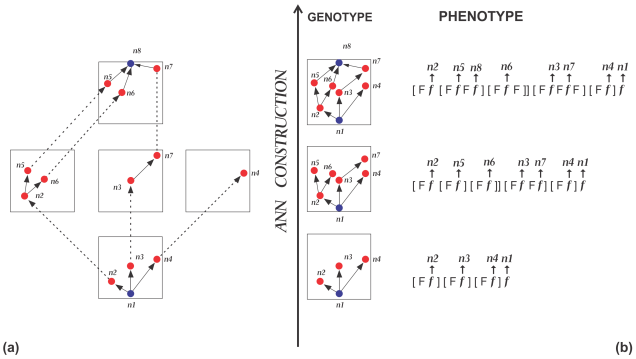

Fig. 3. Construction of an iterated network

search space is restricted to functional ANN topologies. We have used the sigmoid (equation 3) as activation function in all neurons of the ANNs. The ANNs use the parameters : learning rate $\alpha_0 = 0.9$, in equation 4 α is the learning rate, t(epoch) and ρ (number maximum of epochs=200000).

$$\vartheta(x) = \frac{1}{1 + e^{-kx}} \quad (3)$$

$$\alpha_t = \frac{\alpha_{(t-1)}}{(1 + \frac{t}{\rho})} \quad (4)$$

A suitable measure to evaluate the fitness of an ANN is the mean square error, where the error per pattern is the difference between the ANN's output and the desired output, given by equation (5):

$$MSE(k) = \frac{\sum_{p=1}^{q}(d_p - o_{kp})^2}{q} \quad (5)$$

where d_p and o_{kp} are the desired output and the output of individual k for pattern p.

B. *L-Rules extraction with Genetic Algorithm*

In this subsection we present the process of rules extraction with Genetic Algorithm(GA). GA is a computational model that over-simplifies what happens in Nature. Usually in the problems of local search where the GAs are employed, a bit string is called a chromosome (the genotype), each bit is a gene and a gene set represent the parameters of a function to be optimized. Each string can be assigned a fitness, which indicates the quality of the solution (the phenotype) it encodes.

In our research with the goal of making GAs closer to biological processes, in the indirect encoding scheme adopted, the genes encode a recipe (L-system, described in subsection III.A). In order to model the "constructive" approach found in the evolutionary process, in our approach L-rules are extracted automatically by the Genetic Algorithm(GA) in order to control the number of firing times, i.e. times we can apply each rule. The recursive rules, shown in Table I, will drive the stages of development of neurons (process described in subsection III.A and illustrated in Figure 2) and this genetic information is passed on to their offspring's by evolutionary processes acting on genes and modifying them by genetic operators (mutation and crossover) over generations.

Dawkins [2] emphasizes: *The biological genes, only have any meaning when they are translated by means of protein synthesis in "rules of growth" for the development of organs. However, geneticists do not know the complete genetic formula that drives this process of development of neurons.*

At biological level, see Figure 4(b), first the Deoxyribonucleic Acid (DNA) is transcribed into Ribonucleic Acid (RNA) and after occurs the translation of RNA into proteins, in this process a linear sequence of amino acids consisting of three of the four nucleotides (U,G,A,G) of the genetic code (Table II), join to form proteins. From the Table II, the translation process is obtained, for example UUU is translated into Phenylalanine (Phe), UUA is translated into Leucine (Leu) and so on. In Figure 4(b), the protein is formed by a sequence of amino acids starting with Methionine (Met) and ending with Proline (Pro). The protein synthesis triggers all stages of the neurons' development (Phenotypic effects), as shown in Figure 4(b). This flow of information DNA→transcription→RNA→translation→protein is referred to as the central dogma in molecular biology.

The elements of the alphabet $\Sigma = \{.; f; F; n; [;]; *; B, \}$ displayed in the Table II are a metaphor of the genetic code. Each two bits represent one nucleotide, for example the following set (00,01,10,11) symbolize (U,C,A,G) in the genetic code. Accordingly six bits represent three nucleotides, likewise (000000,011111) symbolizes (UUU,CGG). The Function Rule-Extraction-with-GA, attempts to mimic

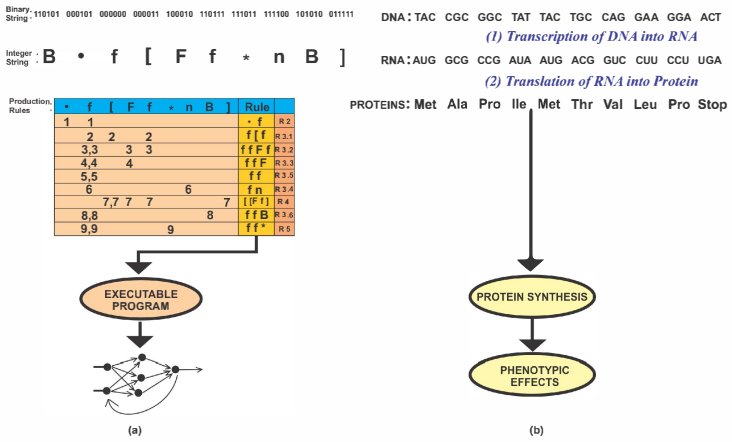

Fig. 4. Process of DNA transcription into RNA and translation into protein and analogue artificial where a binary string is transcribed into an integer string and after that translated into production rules of L-System

the two biological processes: transcription and translation presented previously, the flow of information for the metaphor is now Binary String→transcription→Integer String→translation→Production Rules, see Figure 4(a).

Function Rule-Extraction-with-GA

Step 1 - Dynamically allocate an array of integers 'B' with dimension [IxG], where I is the number of individuals in the population and G is the number of desired genes, randomly assign values 0 or 1 to each position (i,g) in the array.
Step 2 - Read the chromosome, six bits at a time, starting from the first one.
Step 3 - Each group of six bits that was read in step 2, must be converted for a symbol of the alphabet defined by the grammar (L-System), described in section III.A, in agreement with Table II, that will result in a string. The order of reading of Table II is the following: to determine the bits that codes each character following the steps. Determine which of the four rows on the left corresponds to the first two bits of the string, and then choose a column according to the middle two bits, and finally choose a row on the right using the last two bits. For example, the character corresponding to 001000 is 'n'.
Step 4 - Dynamically allocate an array of characters 'D' with dimension [Ix(G/6)], where I is the number of individuals in the population and G is the number of desired genes. Assign to each position in the array (i',g') values obtained in the conversion of binary characters in step 3. Each line in array 'D' will be represented by a string, which represents possible rules illustrated in Figure 4(a).
Step 5 -For each individual (row) of Table 'D' obtained in step 4, find all the substrings which encode a valid production rule, reading from the first character and continuing in contiguous positions or not.
Step 5.1-Reject all the substrings that do not meet with the representation: **.f[Ff *nB]**
Step 6- Repeat step 5 for all individuals in the population, starting to read the strings obtained in step 4 from other positions, for example, from the start to the end and vice versa. Individuals in the population who do not have valid rules will be assigned zero fitness.
In Figure 4(a), are shown the positions of the string **.f[Ff *nB]** in which the production rules(substrings) of the L-System are found.

The steps 1-4 of the Function Rule-Extraction-with-GA mimic the process of transcription of DNA into RNA, now we have the transcription of binary string into integer string, see Figure 4(a). The step 5 represents the translation of an integer string, obtained previously, on valid production rules of the L-system proposed in section III.A (Table II). The shortest string that contains all rules is **.f[Ff *nB]**. In Figure 4(a), are shown the positions in the string **.f[Ff *nB]** where the production rules(substrings) of the L-System are found. For example the Rule 2 (.→f) symbolically represented by (.f), see Figure 4(a), is found at positions 1 and 2. Identically the Rule 3.1 (f→[f) is found at positions 1 and 2,3 and 5 and so on.

	00 (U)	01 (C)	10 (G)	11 (A)	
00 (U)	f (UUU)	F (UCU)	n (UAU)	. (UGU)	00 (U)
00 (U)	n (UUC)	. (UCC)	f (UAC)	F (UGC)	01 (C)
00 (U)	F (UUA)	f (UCA)	B (UAA)	f (UGA)	10 (A)
00 (U)	[(UUG)	n (UCG)	[(UAG)	* (UGG)	11 (G)
01 (C)	f (CUU)	] (CCU)	n (CAU)	* (CGU)	00 (U)
01 (C)	* (CUC)	F (CCC)	f (CAC)	F (CGC)	01 (C)
01 (C)	] (CUA)	f (CCA)	* (CAA)	[(CGA)	10 (A)
01 (C)	f (CUG)	* (CCG)	B (CAG)	] (CGG)	11 (G)
10 (A)	* (AUU)	] (ACU)	n (AAU)	f (AGU)	00 (U)
10 (A)	f (AUC))	B (ACC)	f (AAC)	B (AGC)	01 (C)
10 (A)	F (AUA)	[(ACA)	B (AAA)	n (AGA)	10 (A)
10 (A)	* (AUG)	f (ACG)	* (AAG)	] (AGG)	11 (G)
11 (G)	] (GUU)	[(GCU)	F (GAU)	n (GGU)	00 (U)
11 (G)	n (GUC)	B (GCC)	[(GAC)	. (GGC)	01 (C)
11 (G)	f (GUA)	] (CGA)	B (GAA)	F (GGA)	10 (A)
11 (G)	B (GUG)	f (GCG)	* (GAG)	[(GGG)	11 (G)

TABLE II. THE GENETIC CODE, FROM THE PERSPECTIVE OF MRNA, THIS CODE IS TRANSLATED AS IN FIGURE 4(B). IN THE SAME TABLE THE DNA'S METAPHOR

C. *Crossover and Mutation Operators*

ADEANN have used multi-point crossover in the chromosome, randomly chosen, proportional to the chromosome length divided by six and multiplied by the Crossover rate(Cr). After applying the crossover operator, the mutation operator performed by varying the genes values of a chromosome is applied to each of the resulting offspring. It aims at preventing the GA from trapping in local optima by exploiting new solutions spaces. ADEANN have used multiple mutation points randomly chosen with a probability Pm, proportional to the chromosome length divided by six and multiplied by the Mutation rate(Mr).

D. *Fitness Evaluation and Selection Mechanism*

The tournament selection is adopted in this paper to select individuals for reproduction. In ADEANN the tournament size (Ts) is set to three, which means that there are three chromosomes competing with each other. Only the best chromosome among them is selected to reproduce. In our system we propose a penalty approach that prevents the algorithm from unnecessarily producing ANN's with a large number of hidden neurons (NNHL). In order to do this our system compares two fitness functions, given by: equations (6) and (7).

$$FitnessI = (\frac{A1}{MSE} + \frac{A2}{NNHL}) \quad (6)$$

$$FitnessII = A3 + A4 \quad (7)$$

where:

$$A3 = \exp(-MSE) \times \exp(-NNHL) \quad (8)$$

$$A4 = \frac{1}{(MSE \times NNHL)} \quad (9)$$

IV. EXPERIMENTAL SETUP AND RESULTS WITH ADEANN

To validade ADEANN and make comparisons with other NEAs, we have used for tests three well-known classification problems of increasing complexity: XOR, Prediction of the effect of a new drug on breast cancer and KDDCUP'99. Table III presents a brief description of the datasets used in this paper.

Dataset	No.of inputs	No.of Classes	Patterns
Breast Cancer [13]	15	3	699
KDDCUP99 [26]	42	5	125973

TABLE III. DESCRIPTION OF THE DATASETS USED IN THIS PAPER

A. *Experimental Setup*

Due to the stochastic nature of the proposed NEA, it must run multiple times for each dataset, and the, comparisons with other NEAs are made by performing proper statistical tests. In this paper we have used the technique k-fold cross-validation (k=2) to measure the accuracy of our algorithm. In 2-fold cross-validation, the data is divided in two halves and the algorithm run two times, in each fold, the algorithm is trained with one half and tested on the other half, and vice versa. In each run of the algorithm, the test set is used to calculate the error stated in equation (5) and after the equation (7) is used to the measure the classification accuracy of each ANN generated in the search process as a estimation of its generalization performance.

However, before conducting formal experiments it is necessary to set the parameters involved in ADEANN. We have empirically determined these parameters, based on several experiments for the XOR problem, different combinations of the parameters values have been tested, see Table V. The first column specifies the number of the experiment, generations and individuals per population, the second the neural network solution returned by the search process. The third describes the chromosome length, the fourth the number of crossover and mutation points, the fifth the crossover rate the last three columns present the simulations results : average (μ) and standard deviation (σ) of the number of neurons in the hidden layer and MSE(Mean Square Error) in the generalization phase. The settings observed to be superior are given in Table IV.

Pop.size(υ)	Generations	C_r	P_m	P_{elit}
[20-100]	100	[0.3,0.9]	0.1	2%

TABLE IV. THE PARAMETERS USED IN THE EXPERIMENTS

B. *Evolving XORs*

The XOR problem is not linearly separable, an ANN requires hidden neurons to solve it. The inputs must be combined at some hidden unit, as opposed to only at the output node. These structural requirements make XOR suitable for testing ADEANN's ability to evolve ANNs architectures.

In the experiments, it has been observed that in the last fifteen simulations, see Table V, the fitness given by equation 7 automatically prevents the algorithm from unnecessarily producing complex ANNs with a large number of hidden neurons, which is not always possible using the fitness function given by equation 6, according to the first five experiments. For chromosomes smaller than 300 bits, the crossover rate of 0.6 increases the number of production rules which are found and for chromosomes larger than 300 bits, the crossover rate of 0.9 is more adequate. A greater number of crossover and mutation points are needed for larger chromosomes, which has direct influence on the number of production rules that are found. The results of the best(ninetieth) and worst(first) experiments, considering a small MSE and a minimum number of hidden neurons, are highlighted in bold in the table V.

GxP	N	CL	(c,m)	Cr	μ	σ	MSE
E1-50x30	**(2,8,1)**	**180**	**(9,3)**	**0.6**	**6.50**	**1.50**	**0.000046**
E2-50x30	(2,7,1)	216	(10,3)	0.6	4.80	2.32	0.000049
E3-50x30	(2,5,1)	252	(12,4)	0.6	4.80	1.72	0.000048
E4-60x30	(2,6,1)	252	(12,4)	0.9	3.86	1.81	0.000032
E5-50x20	(2,6,1)	276	(13,5)	0.6	6.00	1.94	0.000029
E6-50x30	(2,2,1)	180	(18,3)	0.3	5.67	2.62	0.000076
E7-50x30	(2,2,1)	180	(18,3)	0.6	3.20	0.75	0.000075
E8-50x30	(2,2,1)	180	(18,3)	0.9	4.50	2.18	0.000051
E9-50x20	(2,2,1)	216	(32,28)	0.6	6.00	2.07	0.000056
E10-50x20	(2,2,1)	216	(32,28)	0.9	3.71	2.43	0.000056
E11-50x30	(2,2,1)	252	(37,33)	0.6	3.89	3.65	0.000056
E12-60x30	(2,2,1)	252	(37,33)	0.9	4.36	1.72	0.000055
E13-50x30	(2,2,1)	300	(45,40)	0.6	3.31	1.65	0.000053
E14-60x30	(2,2,1)	300	(45,40)	0.9	3.77	2.01	0.000053
E15-50x30	(2,2,1)	360	(54,48)	0.6	3.69	2.14	0.000056
E16-50x30	(2,2,1)	360	(54,48)	0.9	3.71	1.87	0.000056
E17-50x30	(2,2,1)	390	(58,52)	0.6	3.79	1.61	0.000053
E18-50x30	(2,2,1)	390	(58,52)	0.9	5.10	2.05	0.000053
E19-100x30	**(2,2,1)**	516	**(77,68)**	**0.9**	**5.28**	**2.03**	**0.000024**
E20-100x30	(2,2,1)	516	(77,68)	0.6	4.15	2.17	0.000025

TABLE V. PARAMETERS USED BY THE GENETIC ALGORITHM IN 20 EXPERIMENTS AND SIMULATION'S RESULTS FOR THE XOR PROBLEM.

It can be seen from Table VI that ADEANN results in high classification accuracy for most of the methods and provides the best performance on average. On 100 runs, NEAT [6] finds a structure for XOR in 32 generations (4.755 networks evaluated, $\sigma = 2.553$), on average a solution network had $\mu = 2.35$ hidden nodes. The results of the thirteenth experiment (Table VI) finds our closest solution, which has $\mu = 3.31$ and $\sigma = 1.65$ in 50 generations. ADEANN has an average classification accuracy close to DENN [17], but ADEANN produces a more accurate estimate for the best network returned by the search . ADEANN solves the XOR problem without trouble and in doing so keeps the architecture small. The number of nodes and connections are close to optimal considering that the smallest ANN has a single hidden unit. ADEANN is very consistent in finding a solution. The total number of ANNs trained in the experiments were 30100.

Dataset	NEAT [6]	ADEANN	DENN [17]
XOR	$-$ **2.35**	$98.84\%^{a}$ 3.31^{b}	**98.97%** $-$
	2.553 $-$	**1.65**^{c} **0.000024**^{d}	$-$ 0.004865

TABLE VI. COMPARISON BETWEEN ADEANN AND OTHER METHODS FOR THE XOR PROBLEM. ^{a}AVERAGE CLASSIFICATION ACCURACY, ^{b}AVERAGE NUMBER OF HIDDEN NEURONS, ^{c} STANDARD DEVIATON OF THE NUMBER OF HIDDEN NEURONS, ^{d}MSE.

C. *Prediction of the effect of a new drug on Breast Cancer (BC)*

The problem of predicting the effect of a new drug on breast cancer[13] is to find an analogous mapping between the set of 15 experimentally induced tumors and a clinical tumor (malignant breast cancer), based on their known reactions to the same drugs, so that we are able to predict the effect of a new drug on the clinical tumor after having studied the experimental tumors. On five runs, the best experiment shows that the ADEANN system finds an ANN for the Breast Cancer problem in 100 generations (15000 networks evaluated). The best results are sumarized in table VII. ADEANN results in high classification accuracy for most of the methods and provides the best performance on average. GEGA [22] generates smaller neural networks ($\mu = 3.8$ $\sigma = 1.0$) but has the lowest classification accuracy.

Dataset	GEGA[22]	ADEANN	G3PCX [23]	GA [21]
BC	96.61% **3.8**	**99.11%**^{a} 14.72^{b}	98.94% 5	98.88% 5
	1.0 -	2.54^{c} **0.000115%**^{d}	0 -	0 -

TABLE VII. COMPARISON BETWEEN ADEANN AND OTHER METHODS FOR THE BREAST CANCER DATASET. ^{a}AVERAGE CLASSIFICATION ACCURACY, ^{b}AVERAGE NUMBER OF HIDDEN NEURONS, ^{c} STANDARD DEVIATON OF THE NUMBER OF HIDDEN NEURONS, ^{d}MSE.

D. *KDDCUP'99 Benchmark Intrusion Detection Dataset*

The dataset provided for the 1999 KDD Cup was originally prepared by MIT Lincoln Labs for the 1988 DARPA Intrusion Detection Evaluation Program, with the objective of evaluating research in Intrusion Detection. Soon after, it has become a standard benchmark data set for the evaluation of Intrusion Detection Systems (IDS). The KDDCUP'99 detection contest uses a version of this data set. Attacks fall into four main categories : **DOS** - Denial of service e.g. SYN flood, **R2L** - unauthorized access from a remote machine e.g. guessing password, **U2R** - unauthorized access to local super user (root) privileges e.g. "buffer overflow" attacks and **probing** : surveillance and other probing e.g. port scanning.

In the research [27] an overview about Intrusion Detection System (IDS) is presented. Ideally, an IDS should have an attack detection rate (DR) of 100% along with false positive (FP) of 0%. Nevertheless, in practice this is really hard to achieve. Detection rate (DR) and false positive (FP) are used to estimate the performance of IDS [27] which are given by equations (10) and (11).

$$DR = \frac{Total\_Detected\_Attacks}{Total\_Attacks} * 100 \qquad (10)$$

$$FP = \frac{Total\_Misclassified\_Process}{Total\_Normal\_Process} * 100 \qquad (11)$$

*1) **Dataset Preprocessing and transformation**:* Tavallaee et al. (2009) [26] discussed the problems of KDDCUP'99 Dataset, one of the most important deficiencies is the huge number of redundant records. We addressed this matter by removing all the repeated records on both KDD train and test set, for more details about statistics of redundat records query the authors' research [26].

In our initial experiments we have used the dataset KDDCUP'99 proposed by [26] optimizing it : we reduced the number of attributes from forty-two to nineteen for the following reasons: using statistics of software "Weka", some attributes that had unique values or mostly zero were eliminated, among them, num-outbound-cmds, Land, wrong-fragment, Urgent, Hot, num-compromissed,num-root,num-file-creation, num-access-files, diff-srv-rate, dst-host-same-srv-rate, dst-host-srv-diff-host-rate, dst-host-rerror-rate, dst-host-srv-serror-rate.

Additionally, we eliminated attributes with high correlation coefficient (greater than or equal to 0.8). Highly correlated attributes influence each other and bring little information, as a result it is not interesting to maintain them in the dataset. The following attributes were removed: serror-rate, same-srv-rate, srv-serror-rate, dst-host-srv-serror-rate, rerror-rate srv-rerror-rate, srv-count. Some attributes were selected and normalized: Count, num-failed-logins, dst-host-srv-rerror-rate,num-shells and dst-host-count, these values were normalized with the values assumed in the interval [0,1]. Normalization is necessary in order to provide the data the same order of magnitude.

*2) **Classification**:* Once the changes were made in the dataset proposed by [26]. A study was conducted on the problem of Intrusion Detection using first a single ANN randomly chosen and after the ADEANN system. Figure 5 illustrates the architecture of the ANN used in the simulation process and the features of the Intrusion Detection problem illustrated as inputs to the network. The best experiment shows that the ADEANN system finds an ANN for the IDS problem in 100 generations (3000 networks evaluated), on average a solution network has $\mu = 25.66$ std $\sigma = 15.36$ hidden nodes. For the worst experiment, on average a solution network has $\mu = 29.07$ std $\sigma = 18.56$ hidden nodes. The lowest ANN for that problem has the specification (19,19,1), i.e. with 19 neurons in the hidden layer for a MSE ≤ 0.000181. The larger ANN obtained for the problem has the specification (19,76,1), i.e. with 76 neurons in the hidden layer, the same presents a MSE = ≤ 0.000230. However as the Fitness, given by equation 3, favors the smaller ANNs with good generalization capacity, the ANN (19,19,1) is ranked the best. ADEANN solves the IDS problem without trouble.

*3) **Data mining**:* Table VIII summarizes the results obtained by ADEANN and by an ANN designed by traditional methods. Comparing the results summarized in Table VIII. Based on the results it can be seen that the performance of ADEANN system is better than a single ANN for all the five classes. The evolved ANN provides a more stable and accurate prediction compared to a single ANN, without any human interactive operation. In the research [19] they have used only two classification classes (Normal and Intrusion) the detection rate (DR) obtained for the test data was DR=95% for Normal and FP=7.7% for False Positives, DR=92.3% Intrusion and FP=5% for False Positives. ADEANN used five classes and we have obtained DR=97.67% and FP=3.4% for Normal and an average of DRavg=93.79% for Intrusion and FPavg=1.82% for False Positives. In the research [5] the detection rate (DR) obtained for the test data was DR=96.3% for Normal class and DRAvg= 92.0% for Intrusion. The statistical results for other methods [19],[29] are also summarized in Table X. It can be seen that ADEANN provides an accurate detection of attacks compared to other methods. The corresponding hypothesis tests for them are illustrated in Table IX, the same also gives the hypothesis for each statistics.

Classifier	Normal	Probe	Dos	R2L	U2R	AVG
ANN (DR%)	95.45%	91.66%	85.71%	92.85%	93.33%	90.89%
ANN (FP%)	3.9%	2.9%	1.98%	0.99%	0.99%	2.15%
ADEANN (DR%)	97.67%	93.75%	92.85%	94.11%	94.44%	93.79%
ADEANN (FP%)	3.4%	2.26%	1.74%	0.87%	0.87%	1.82%

TABLE VIII. RESULTS FOR THE TEST DATASET [25],CONSIDERING FIVE CLASSES, DR(%) DETECTION RATE AND FP(%) FALSE POSITIVES .

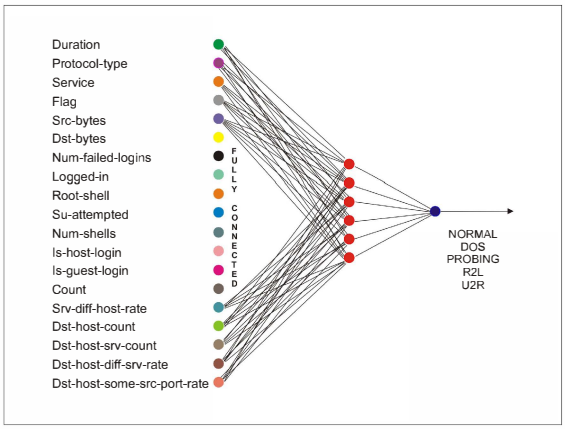

Fig. 5. The architecture of the ANN used in the simulation process and the features of the Intrusion Detection problem illustrated as inputs to the network.

Condition	Ho	H1	Ho	P
(DR%)N and (DR%)Avg I	ADEANN $\geq$ANN	ADEANN<ANN	1	0
(FP%)N and (FP%)Avg I	ADEANN$\leq$ANN	ADEANN>ANN	1	0

TABLE IX. NULL AND ALTERNATIVE HYPOTHESIS, STATISTICS AND HYPOTHESIS TEST OF EVALUATION FOR THE IDS PROBLEM

Dataset	[19]	ADEANN	[5]	[?]
KDDCUP99	95.0% 7.7%	**97.67%**^{a} **3.4%**^{b}	96.3% -	94.44%^{e} -
	92.3% 5.0%	**93.79%**^{c} **1.82%**^{d}	92.0% -	94.25%^{f} -

TABLE X. COMPARISON BETWEEN ADEANN AND OTHER METHODS FOR THE KDDCUP'99 DATASET. ^{a}DR(%) FOR NORMAL CLASS , ^{b}FALSE POSITIVES FOR NORMAL CLASS, ^{c} DR(%) FOR INTRUSION CLASS, ^{d}FALSE POSITIVES FOR INTRUSION CLASS, ^{e,f} USING A MLP NETWORK AND TRAINING FUNCTION : TRAINOSS.

V. DISCUSSION AND CONCLUSION

One of the characteristic of the ADEANN is that it needs a minimal expert's effort for customization. It has five parameters listed in Table IV to be set by the user. The experiment results based on KDD99 dataset show that removing redundant and irrelevant features is very important for the classification task, and even may improve the classification accuracy. One goal of this paper is to propose a system that can produce

ANNs with not only low error on the training set but also high generalization on previously unseen data, measured by the classification accuracy on the test set. Accordingly, as stated previously, the fitness function given by equation (7) incorporated into ADEANN simplify ANNs generated through the evolutionary process, thus leading to more generalization ability.

The use of grammar evolution has the benefit of leading to a compact encoding offering key advantages in terms of scalability. ADEANN can use chromosomes which are 35% shorter than the ones used in our previous research [8] (reduction from 512 to 180 bits). In the method [1] if an ANN has N neurons will result in a binary string of $(N(N+1)/2)$ bits and the method of [4] has a quadratic behavior N^2. If the two methods [4],[1] had used the chromosomes length that we used in the experiments described in the section IV, which are in the interval [180,516], they would have generated ANNs with a maximal number of neurons in the intervals (13,22) and (18,31) respectively. Depending on the value of NR (randomly generated), in the equations (1) and (2) ADEANN can generate larger ANNs than the methodologies [4],[1]. The compact encoding scheme used by ADEANN allows to generate complex topologies of ANNs.

Additionally, our approach increases the level of implict parallelism of the GA, by reading the chromosome in several alternate positions, see (section III.B, Function Rule-Extraction-with-GA), so as to realize the parallel search of the rules of the L-system, which represents a gain over traditional methods. The experiments results shows that ADEANN is very promising and can generate economical neural networks architectures with optimal number of neurons and good generalization capacity.

REFERENCES

[1] R.B. Boozarjomehry and W. Svrcek, *Automatic design of neural networks structures. Computers and Chemical Engineering vol.(25), pp.1075-1088, 2001*

[2] R. Dawkins, The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design *W.W. Norton & Co.*,1996.

[3] Soltanian, K.; Tab, F.A.; Zar, F.A.; Tsoulos, I., "Artificial neural networks generation using grammatical evolution," Electrical Engineering (ICEE), 2013 21st Iranian Conference on , vol., no., pp.1,5, 14-16 May 2013.

[4] KITANO.H.(1990) "Designing neural networks using genetic algorithms with graph generation system, " Complex Systems, vol.4, no. 4, pp.461-476.

[5] Mohammed Sammany and Marwa Sharawi and Mohammed El Beltagy and Imane Saroit, Artificial Neural Networks Architecture For Intrusion Detection Systems and Classification of Attacks *,in: 5th international conference INFO2007, Cairo University.*

[6] Stanley,Kenneth O. and Miikkulainen, Risto.Evolving neural networks through augmenting topologies. Evolutionary Computation, 10:99-127,2002.

[7] Dong-Wook,Lee and Sang-Wook,Seo and Kwee-Bo, Sim. Evolvable Neural Networks Based on Developmental Models for Mobile Rôbot Navigation. International Journal of Fuzzy Logic and Intelligent Systems Vol.7 No.3, 2007, 176-181.

[8] L.M.L. De Campos, ; M. Roisenberg, ; R.C.L. de Oliveira."Automatic design of Neural Networks with L-Sytem and genetic algorithms - A biologically inspired methodology," Neural Networks (IJCNN), The 2011 International Joint Conference on , vol., no., pp.1199-1206, July 31 2011-Aug.5.2011 doi: 10.1109/IJCNN.2011.6033360.

[9] J. Gauci , K.O. Stanley (2010) Autonomous evolution of topographic regularities in artificial neural networks. Neural Computation 22(7):1860-1898.

[10] Daniela Snchez, Patricia Melin, Optimization of modular granular neural networks using hierarchical genetic algorithms for human recognition using the ear biometric measure, Engineering Applications of Artificial Intelligence, Volume 27, January 2014, Pages 41-56,

[11] G.F. Miller and P.M. Tood and S.U. Hedge (1989).Designing neural networks using genetic algorithms. In Proceedings of International on Genetic Algorithms and their Applications, pages 379-384.

[12] Iztok Fister Jr. and Xin-She Yang and Iztok Fister and Janez Brest and Dusan Fister.A Brief Review of Nature-Inspired Algorithms for Optimization. Elektrotehniški vestnik, 80(3), 2013.

[13] Nikola K. Kasabov. Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering. Second Print 1998. Massachusetts Institute of Technology.

[14] Krömer, Pavel and Platoš, Jan and Snášel, Václav. Nature-Inspired Meta-Heuristics on Modern GPUs: State of the Art and Brief Survey of Selected Algorithms. Int. J. Parallel Program. 42(5) (2014) 681-709.

[15] Park, H.S. and Tran, N.H. Development of a biology inspired manufacturing system for machining transmission cases. International Journal of Automotive Technology 14(2) (2013) 233-240.

[16] Sebastian Risi and Kenneth O. Stanley. An Enhanced Hypercube-Based Encoding for Evolving the Placement, density and connectivity of neurons, Artif.Life 18 (4) 2012 331-363.

[17] Ilonen, Jarmo and Kamarainen, Joni-Kristian and Lampinen, Joun. Differential Evolution Training Algorithm for Feed-Forward Neural Networks. Journal of Neural Processing Letter,pp.93-105, 2003.

[18] Ioannis Tsoulos, Dimitris Gavrilis, Euripidis Glavas, Neural network construction and training using grammatical evolution, Neurocomputing, Volume 72, Issues 13, December 2008, Pages 269-277, ISSN 0925-2312.

[19] Campos, Lidio Mauro Lima ; Oliveira, Roberto Clio Limo ; ROISENBERG, MAURO . Network Intrusion Detection System Using Data Mining. In: Chrisina Jayne, Shigang Yue, Lazaros Iliadis. (Org.). Communications in Computer and Information Science. 1ed.Londres: Springer Berlin Heidelberg, 2012, v. 1, p. 104-113.

[20] J.P. Donate, G.G. Sanchez, R.S. Miguel. Time Series Forecasting. A Comparative Study Between An Evolving Artificial Neural Networks System and Statistical Methods. nt. J. Artif. Intell. Tools 21, 1250010 (2012) [26 pages] DOI: 10.1142/S0218213011000462.

[21] Cantu-Paz E., Kamath C. An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. IEEE Trans Syst Man Cybern B Cybern. 2005 Oct;35(5):915-27.

[22] Fardin Ahmadizar, Khabat Soltanian, Fardin AkhlaghianTab, Ioannis Tsoulos, Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm, Engineering Applications of Artificial Intelligence, Volume 39, March 2015, Pages 1-13.

[23] Kalyanmoy Deb, Ashish Anand, and Dhiraj Joshi.A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization.Evolutionary Computation 2002 10:4, 371-395

[24] L.M.L.d.Campos, R.C.L.d.Oliveira. A comparative analysis of methodologies for automatic design of artificial neural networks from the beginings until today, in: Proceedings of the 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence, BRICS-CCI-CBIS'13, IEEE Computer Society, Washington, DC, USA, 2013, pp.453-458.

[25] Stolfo SJ, et al., KDD cup 1999 data set, Irvine: University of California. KDD repository, http://kdd.ics.uci.edu.

[26] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set, Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[27] Haque, Md Enamul and Alkharobi, Talal M.Adaptive Hybrid Model for Network Intrusion Detection and Comparison among Machine Learning Algorithms.International Journal of Machine Learning and Computing, Vol. 5, No. 1, February 2015.

[28] Iztok Fister Jr. and Xin-She Yang and Iztok Fister and Janez Brest and Dusan Fister.A Brief Review of Nature-Inspired Algorithms for Optimization. Elektrotehniški vestnik, 80(3), 2013.

[29] Nazir, Afrah.A Comparative Study of different Artficial Neural Networks based Intrusion Detection Systems. International Journal of Scientific and Research Publications. Vol. 3, number:7, 2013.