Evolutionary Approach for Structural and Paramentric Adaptation of BP-like Multilayer Neural Networks

Jaroslav Tuhársky, Tomaš Reiff, Peter Sinčák

Center for Intelligent Technologies, Department of Cybernetics and Artificial Intelligence, FEI TUKE Kosice, e-mail: jaroslav.tuharsky@gmail.com

Abstract: Paper describes experience with selected methods of structural and parametric adaptation of BP like multilayer neural networks for real time learning and application in Reinforcement strategy. Non-linear function aproximation is tested and evaluated with these approaches with the aim of perspective application in Computer games and building an intelligence for Bots in the virtual reality. TWEANN and NEAT methods are tested and experimental and theoretical study was accomplished on these methods. They are based on evolutionary computation and optimization of neural networks structure and synaptic weights. Application potential for supporting the intelligence of NAO humanoid robots is mentioned in the conclusion of the paper.

Keywords: neural networks, evolutionary algorithms, Neuroevolution, TWEANN, NEAT, genetic algorithms.

1 Project Definition and Task Determination

The papers deals with the overview and confirmation project in the scope of structural and parametric training regarding neural networks with Error backpropagation learning. The main goals of of this paper are :

- Make an analysis of selected methods and accomplish a theoretical comparison of these methods.
- Implement pilot experiments and test these methods on non-linear function aproximation (XOR)

2 The State of the Art in the Domain

Evolutionary (EA) is used for solving optimization problems and one of these tasks could be a search for optimal neural networks topology of BP like neural network.

Finding the optimal neural networks (NN) by using EA may consist of NN topologies optimization - searching NN topology able to solve the problem, NN synaptic weights (SW) optimization - search for suitable values of SW. As it is described in [6], neuroevolution (NE), the artificial evolution of neural networks using genetic algorithms (GA), has shown great promise in complex learning tasks. NE searches through the space of behaviors for a network that performs well at given task. Past studies have shown NE to be faster and more efficient than reinforcement learning methods such as Adaptive Heuristic Critic and Q-Learning on single pole balancing and robot arm control. Because NE searches for a behavior instead of a value function, it is effective in problems with continuous and high-dimensional state spaces. In addition, memory is easily represented through recurrent connections in neural networks.

3 Selected Methods and Approaches

3.1 TWEANN

As is described in [6], in traditional NE methods, NN topology is determined, ie. that it isn't evolved by EA. Usually this consists of input, one hidden and output layer, with full connection between the layers. The goal of NE methods with a fixed NN topology is to optimize the values of NN's SW and to determine its functionality. Evolution searches the space of SW of this fully-connected topology by allowing high-performing NN to reproduce. The weight space is explored through the crossover of network SW and through the mutation of single NN SW. However, SW are not the only aspect of NN that contribute to their behavior. The topology of NN also affects their functionality.

Many systems have been developed over the last decade that evolve NN topologies and its SW combined under a name TWEANN - Topology and Weight Evolving Artificial Neural Networks. TWEANN can be divided to those that use a direct encoding and those that use an indirect one. Direct encoding schemes, employed by most TWEANN, specify in the genome every connection and node that will appear in the phenotype. In contrast, indirect encodings usually only specify rules for constructing a phenotype.

One of the main problems of NE is the Competing Conventions Problem, also known as Permutation Problem. It is a situation where the NS computes the same function (the same in terms of their functionality), despite their neurons in the hidden layer are in a different order, ie. that are represented by different genomes. Such NS are incompatible for their mutual crossing. In cases where the genomes represent the same solutions and do not have the same encoding, GO crossover produces damaged offspring.

Figure 1 depicts the problem for simple 3-hiden-unit network. The three hiden neurons A, B, and C, can represent the same general solution in 3! = 6 different permutations. When one of these permutations crosses over with another, critical information is likely to be lost.

An even more difficult form of competing conventions is present in TWEANN because its NN can represent similar solutions using entirely different topologies, or even genomes of different sizes.

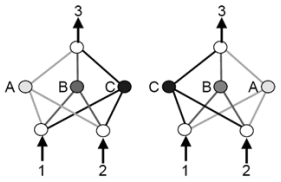

Figure 1
The competing conventions problem

Another problem of TWEANN is the protection of innovations. In TWEANN methods arises innovation by adding a new structure using mutations. Adding a new structure, in most cases, causes deterioration of the NN fitness. For example, adding a new node represents the non-linearity, where previously was not and adding a new connection can reduce fitness of NN and its disposal of an evolutionary process before the SW connection has a chance to optimize. Optimization of the new structure requires several generations and therefore innovations must survive in a population for a sufficient number of generations to have enough time for this process. Therefore it is important to somehow protect the NN with a structural innovation so that those have a chance to use their new structure.

In many TWEANN systems, the initial population is a collection of random topologies. However, random initial populations turn out to produce many problems for TWEANN. For example, there is a chance that NN will have no path from each of its inputs to its outputs. It is desirable to evolve minimal solutions, so that the number of parameters which have to be searched is reduced.

NEAT method, see chapter 3.2, solves this problem by initializing the population with a minimal structure of the NN without hidden layers, and with the phylosiphy that their structure is rising only when it is appropriate for a given solution.

3.2 NEAT

The method NeuroEvolution of Augmenting Topologies (NEAT) was created by K. O. Stanley and R. Miikkulainen, from the Texas University in Austin, described in [6]. From the same publication is the following description.

3.2.1 Genetic Encoding

NEAT's genetic encoding scheme is designed to allow corresponding genes to be easily lined up when two genomes cross over during matting. Genomes are linear representations of network connectivity. Each genome includes a loist of connections genes, each of which refers to two node genes being connected. Each connection gene specifies the in-node, the out-node, the weight of the connection, whether or not the connection gene is expressed, and an innovation number, which allows finding corresponding genes.

Mutation in NEAT can change both SW and NN topology. SW mutate as in any NE system, with each connection either perturbed or not at each generation. Structural mutations occur in two ways. In the add connection mutation, a single new connection gene with a random weight is added connecting two previously unconnected nodes. In the add node mutation, an existing connection is split and the new node placed where old connection used to be. The old connection is disabled and two new connections are added to the genome. The new connection leading into the new node receives a weight of 1, and the new connection leading out receives the same weight as the old connection.

3.2.2 Historical Markings of Genes

Whenever a new gene appears through structural mutation, a global innovation number is incremented and assigned to that gene. The innovation number thus represents a chronology of the appearance of every gene in the system. The historical markings of genes give NEAT new capability. The system now knows exactly which genes match up with which. When crossing over, the matching genes in both genomes with the same innovation numbers are lined up.

3.2.3 Protecting Innovations through Speciation

Speciating the population allows organisms to compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected in a new niche where they have time to optimize their structure

through competition within the niche. This task appears to be a topology matching problem. In NEAT is the measure of the compatibility distance of a different structures a simple linear combination of the number of excess E and disjoint D genes, as well as the average weight differences of matching genes W, including disabled genes.

$$\delta = c_1 \cdot \frac{E}{N} + c_2 \cdot \frac{D}{N} + c_3 \cdot \overline{W} \tag{1}$$

$\overline{W}$ – average SW differences of matching genes

E – number of excess genes

D – number of disjoint genes

N – number of genes in larger genome (for normalization because of its size)

c_1, c_2, c_3 – coefficients

δ – compatibility (gene's) distance

4 Implementation of NEAT Approach

For the implementation of experiments we have proposed and implemented the software in the creation of which we was inspired by the method NEAT, see chapter 3.2. The reason why we have decided for NEAT is that it provides the best solutions for the problems associated with TWEANN - see chapter 3.1. and is using evolutionary calculations, namely the GA to find the simplest topologies with optimized SW for XOR problem.

4.1 Representing Individuals

The population is made up of individuals - NN. Particular individual, which we call the genome, contains a list of "genes of links".

In the initialization of the population are individuals with a minimum NN topology - see Figure 2., whose structure is made up of only input and output layer, i.e. without hidden layers. Input layer consists of two input nodes, output layer of one output node. Node "bias" was incorporated into the topology so that we can introduce the entry of the external world to all of the neurons.

For each neuron in the NN, we used the same sigmoid activation function.

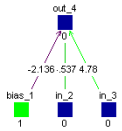

Figure 2
Initial NN topology

4.2 Genetic Operators

Using the Genetic Operators (GO) program searches the SW space for an optimal NN, able to solve the required task, which is in our experiment XOR. In the program we have used the following GO: crossover, SW mutation, add node mutation, and add connection mutation.

4.2.1 Crossover

The probability of crossing is given by parameter CrossProb. Crossover between two compatible genes is indeed in the calculation of SW value, which is inherited by the offspring from the values of his parents SW. Crossover can be done in two ways, either by calculating the average SW value of the parents - AVG parameter or the value of SW is randomly generated from one or the other parent.

Crossover GO is applied only to the individuals of the same species, i.e. can not be a crossing of different species individuals, thus addressing the problem of permutation, see chapter 3.1. The number of offsprings for each species is based on a probabilistic relationship, see (2).

$$\lambda_S = \frac{\overline{\Phi}_S}{\sum_{j=1}^{N} \overline{\Phi}_j} \cdot \mu \tag{2}$$

$\overline{\Phi}_S$ – average fitness of individuals of the species *S*, see equation (3)

μ – number of individuals in the population

N – number of species in the population

λ_S – number of offsprings of the species *S*

where

$$\overline{\Phi}_S = \frac{\sum_{i=1}^{\mu_S} \Phi(a_i)}{\mu_S} \tag{3}$$

$\Phi(a_i)$ – fitness function of individual a_i

μ_S – number of individuals in the species S

4.2.2 Fitness Function

We split the calculation of a Fitness function (FF) into 2 steps. Calculating gross FF (4) from purpose function (5), and calculating adjusted FF (6) based on fitness sharing method. In the experiment, the purpose function is the Sum of Squared Error (SSE) of NNs (5) in solving the XOR task.

$$\Phi\left(f(a_i)\right) = (1-\varepsilon)\cdot\frac{f(a_i) - \min_j f(a_j)}{\min_j f(a_j) - \max_j f(a_j)} + 1 \tag{4}$$

Φ – fitness function

$f(a_i)$ – purpose function of individual a_i

ε – lowest possible value of fitness

where

$$f(a_i) = \sum_{k=1}^{4} e_k^2 \tag{5}$$

e_k^2 – SSE Sum of Squared Error on k-solution of XOR

$$\Phi'(a_i) = \frac{\Phi(a_i)}{\sum_{j=1}^{\mu} sh\left(d(a_i, a_j)\right)} \tag{6}$$

$\Phi(a_i)$ – fitness function of individual a_i

$\Phi'(a_i)$ – adjusted fitness function of individual a_i

$sh(d(a_i,a_j))$ – sharing function

where

$$\sum_{j=1}^{\mu} sh\left(d\left(a_i, a_j\right)\right) = \mu_S \tag{7}$$

μ_S – number of individuals in the species S

S – species in which is the individual who had adjusted the fitness

5 Experiments

5.1 Experiment Example

This experiment shows that in 500 generations, the program NEAT created 4-Node, 5-Node, 6-Node and 7-Node NN. Figure 3 shows the number of individuals pertaining to the topology in the certain generation, as well as the emergence and disappearance of species in the population.

In this case, the 7-node NN was created at the end of the experiment, in the 450. gen. there were only 5 such individuals in the whole population (Figure 3.). Therefore, the program NEATS had sufficient time to search SS (State Space) of SW of 6-Node NN and founds its optimization in the 350. gen., i.e. founds the optimal topology and values of NN's SW able to solve XOR task. The 5-Node NN was optimized in the 130.gen. and the search for the 4-Node NN (which we know that is able to solve XOR task) program NEATS stopped in 260.gen., so that entire 4-Node type was thrown away from the population. See Figure 3 and charts of SSE (Sum of Squared Error) during the evolution which are shown in Figures 4-8.

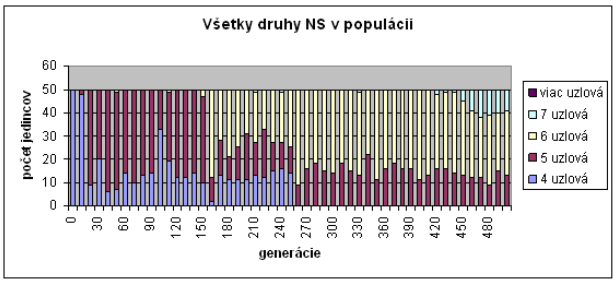

Figure 3
Number of individuals in the population

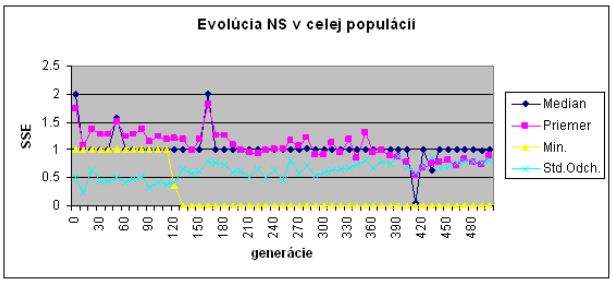

Figure 4
SSE of NN through evolution in all population

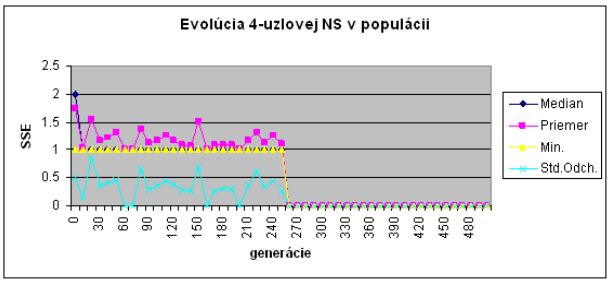

Figure 5
SSE of 4-Node NN through evolution

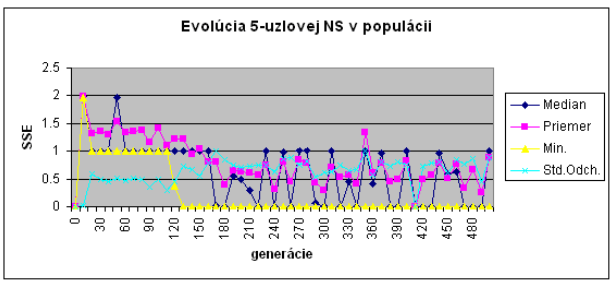

Figure 6
SSE of 5-Node NN through evolution

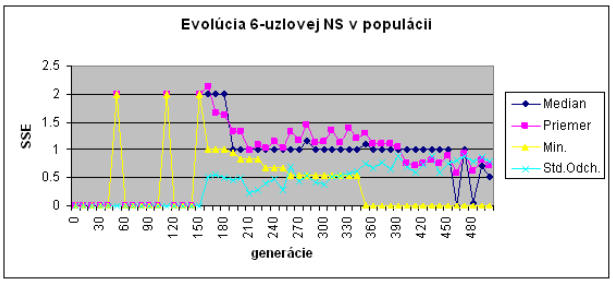

Figure 7
SSE of 6-Node NN through evolution

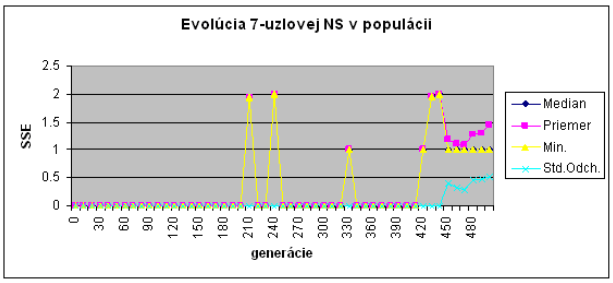

Figure 8
SSE of 7-Node NN through evolution

The reason why the 7-Node NN was not optimized, is due to lack of time (generations) needed for sufficient scan of NN's SW state space.

In most cases to find the simplest (5-Node) topology of NN able to solve XOR task (where SSE = 0) only 100.gen. were needed, but for NN with more complex structure we need more generations for its optimization.

Figure 9. and Figure 10. show individuals of particular species which were evolved on the end of the evolution process.

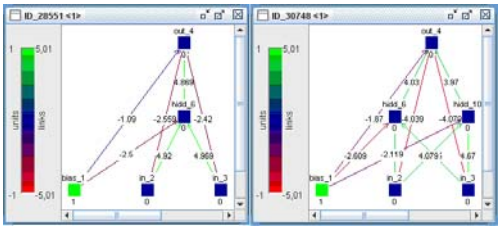

Figure 9
Individuals from species No1 and species No2

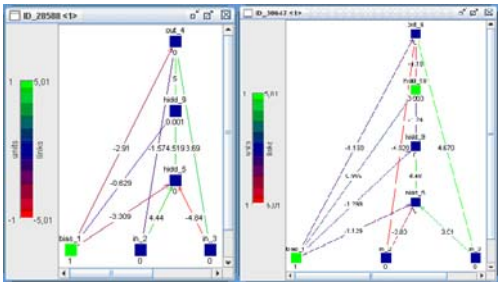

Figure 10
Individuals from species No3 and species No4

5.2 Experiment Results

For selected experiments we have demonstrated the functionality of the implemented NEAT method with ability to cope with problems that arise from the TWEANN approaches.

Conclusion

The functionality of the NEAT method was tested for its ability to evolve various topologies of NN able to deal with XOR problem.

This work has shown strong and weak points of the system TWEANN, as well as the NEAT system, and outlined possible pitfalls, which can be given when using these systems. Tested approach is very effective for the problems of TWEANN which were easily solved.

Our plan was to create a program that will help us to understand this extremely interesting method in depth which is crucial for its future use or improvement.

This work is seen as an essential step on our way to create a system capable of optimizing the NN topology, together with its SW, so that they were able to solve the challenges associated with control of BOTs in space, which would create a system able to adapt on the situation without human intervention. This intelligent control will be used in the control of real robots in space, or BOTs in video games.

References

[1] Peter Sinčák, Gabriela Andrejková : Neurónové siete (inžiniersky prístup) 1. a 2. diel, Košice : Vydavateľstvo ELFA, 1996

[2] Poznámky z prednášok predmetu „Evolučné algoritmy“ pre 4.ročník doc.Ing.Mariána Macha, CSc. na Katedre kybernetiky a umelej inteligencie na FEI TUKE

[3] Vladimír Kvasnička, Jiří Pospíchal, Peter Tiňo : Evolučné algoritmy, Bratislava: Vydavateľstvo STU, 2000, ISBN 80-227-1377-5

[4] Vladimír Mařík, Olga Štěpánková, Jiří Lažanský a kol.: Umělá inteligence 3, Praha : Vydavateľstvo ACADEMIA, 2001, ISBN 80-200-0472-6

[5] Vladimír Mařík, Olga Štěpánková, Jiří Lažanský a kol.: Umělá inteligence 4, Praha : Vydavateľstvo ACADEMIA, 2003, ISBN 80-200-1044-0

[6] Kenneth O. Stanley, Risto Miikkulainen : Evolving Neural Networks through Augmenting Topologies, The MIT Press Journals, 2002

[7] Kenneth O. Stanley, Risto Miikkulainen : Efficient Evolution of Neural Network Topogies, Proceedings of 2002 Congerss on Evolutionary Coputation. Piscataway, NJ : IEEE

[8] Kenneth O. Stanley, Boddy D. Bryant, Risto Miikkulainen : Real-Time Neuroevolution in the NERO Video Game, IEEE Transactions on Evolutionary Computation, 2005