IMPROVED ADAPTIVE META-NETWORK DESIGN EMPLOYING GENETIC ALGORITHM TECHNIQUES

Ben McElroy and Gareth Howells
School of Engineering and Digital Arts, University of Kent, Canterbury, Kent, CT2 7NT, U.K.

Keywords: Weightless Neural Network, Meta-Networks, Genetic Algorithm, Neural Architectures, Ultrasonic Sensors.

Abstract: This paper investigates the employment of a Genetic Algorithm to optimally configure the parameters of a class of weightless artificial neural network architectures. Specifically, the Genetic Algorithm is used to vary the parameters of the architecture and reduce the rigidity of the mutation algorithm to allow for a more varied population and avoidance of local minima traps. An exemplar of the system is presented in the form of an obstacle avoidance system for a mobile robot equipped with ultrasonic sensors.

1 INTRODUCTION

The optimisation problem for classes of artificial neural networks has been investigated over a number of years (Kordík, et al., 2010) (Abraham, 2004) (Abraham, 2002). This paper investigates the optimisation problem for a class of simple yet flexible artificial networks termed RAM based or weightless networks. The specific problem considered in this paper is that of enabling a mobile robot to determine the optimal direction of travel using weightless neural networks. This work looks at using simple measures fed into the network to determine if it can settle on an appropriate obstacle avoiding response/route. Adaptive systems offer the ability to learn and generalize from a set of known examples allowing them to recognize previously unseen inputs based on their similarity of characteristics with previously seen examples. RAM-based weightless neural networks are a type of neural network well suited to the expression of solutions to such logical problems (McElroy, et al., 2010). Such neural networks possess many advantages over conventional weighted neural networks since they allow the possibility of:-

- One shot learning – this is an object categorization problem of current research interest. Usually machine learning based object categorization algorithms require a lot of training on hundreds or thousands of items, one-shot learning attempts to minimize this by having sufficient information about the object categories from just one, or only a few, training images/items.
- Arbitrary mappings from inputs to outputs – the system is more robust as it does not require specific pathways to be assigned from input to output.
- Easier direct hardware implementation.

Manual optimisation of the configuration parameters of Artificial Neural Networks (ANNs) for a specific problem domain currently relies heavily on human experts with sufficient knowledge on the different aspects of the network as well as the problem domain itself (Yao, 1999). As the complexity of the problem domain increases and when near-optimal networks are desired, manual searching becomes more difficult and unmanageable. This paper seeks to evaluate the potential of using weightless ANNs within a meta-network structure to determine the direction of a robot when it encounters an obstacle and improve the accuracy of the response generated. The paper is structured as follows. Section 1.1 briefly outlines evolutionary neural networks and their advantages. Section 2 describes the robot being used and the setup of the sensors. Section 3 goes into detail about weightless neural networks and how they differ from their weighted counterparts. Section 4 covers the problem this paper is exploring while section 5 explains how the meta-network operates in detail. Section 6 shows the experimental setup and how the data was encoded, and section 7 displays the results. Finally, section 8 concludes the paper with finishing remarks.

McElroy B. and Howells G..
IMPROVED ADAPTIVE META-NETWORK DESIGN EMPLOYING GENETIC ALGORITHM TECHNIQUES.
DOI: 10.5220/0003539401420148
In *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2011), pages 142-148
ISBN: 978-989-8425-74-4
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

1.1 Evolutionary Artificial Neural Networks

Evolutionary Artificial neural networks (EANNs) are a specific type of ANN whereby evolution is used as another form of adaptation which supplements learning (Abraham, 2004) (Abraham, 2002) (Slowik, et al., 2008). One distinct feature of EANN's is their adaptability to a dynamic environment. EANN's can adjust to a multitude of scenarios as well as changes in the environment. There are three significant areas to which evolution has been introduced: the connection weights, the learning rules and the architectures (Yao, 1999). In this paper we deal primarily with the evolution of architectures which enables ANN's to adapt their topologies to different tasks without human intervention and thus provides an approach to automatic weightless ANN design, a class of architectures to which this technique has not previously been applied.

2 ULTRASONIC SENSORS

A robot equipped with seven ultrasonic sensors was used in the investigation. Although further sensors may prove necessary for a practical system, this configuration was used for the purposes of evaluating the proposed network configuration as it formed a balance between complexity and utility of the data. The ultrasonic sensors that adorn the robot are shown in figure 1. Each sensor has an associated ID which was used to identify them in the data. The grey area in Figure 1 shows the approximate cone that the sensors operate within. The sensors have a maximum range of four metres, and return the distance to the closest object within their cone of detection. The sensors are set to return data every half a second with a distance given in millimetres.

Ultrasonic sensors work in a similar way to radar or sonar which assess characteristics of a target by interpreting the return signal, or 'echoes', from radio or sound waves respectively. These sensors typically produce high frequency sound waves and look for the echo from them. The Sensors then have a time delay between sending and receiving the echo and use this to calculate the distance to the object. Sonar sensors are used successfully in a wide variety of applications, such as medical imaging, non-destructive testing and vehicle ranging systems.

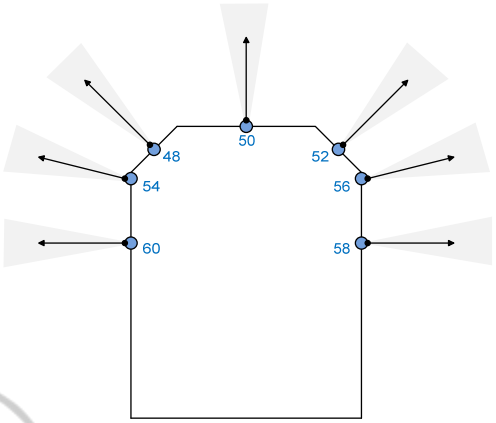

Figure 1: Robot Sensor Setup.

3 WEIGHTLESS ARTIFICIAL NEURAL ARCHITECTURES

As a class of architectures, Weightless Neural Networks were first introduced by Bledsoe and Browning in 1959 (Bledsoe, et al., 1959). They consist of 'weightless' neurons - neurons that have no weight between the input and the node – with the inputs and outputs expressed as simple binary values. While their weighted brethren require a lot of training, Weightless networks can be trained very quickly and installed on much simpler hardware. Further, whereas in other neural network models the weights are adjusted, WsNNs are trained by modifying the composition of the look-up tables.

The architecture to be employed in this investigation is the Generalised Convergent Network (GCN) (Howells, et al., 1995). It employ layers of neurons each independently attached to a given sample pattern whose distinct outputs are merged via a further layer to produce an output matrix equal in dimension to the original sample pattern. A varying number of groups of such layers arranged sequentially may be employed by the various architectures.

An example GCN architecture is illustrated in Figure 2 and it possesses the following general properties:-

- Network neurons are typically arranged in a two dimensional layer where each row represents a component of the input data.
- Each element within the pattern is associated with a corresponding neuron within each layer.
- The layers comprising the network are arranged in two groups, termed the Pre group and the Main group.

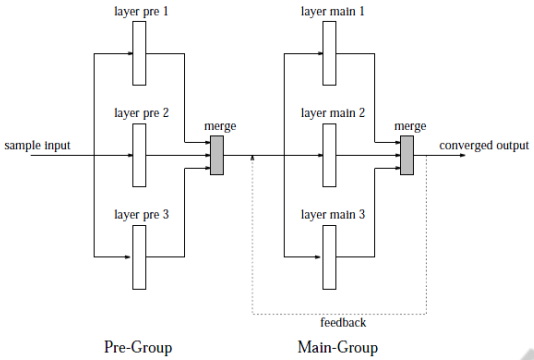

Figure 2: GCN example layer setup.

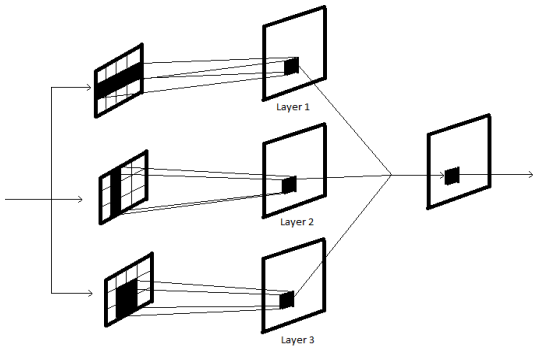

Figure 3: Sample potential layer connectivities.

- A further Merge layer exists after each group whose function is to combine the outputs of the constituent layers of the group. The Merge operation is performed on the corresponding neurons from each layer within the group and for each position within the layer shown in figure 3. The number of inputs of a neuron comprising a Merge layer is thus equal to the number of layers within the group to which it pertains.
- The Merged output of the Main Group is fed back, unmodified, to the inputs of each layer comprising the group.
- The number of layers within each group and the connectivity of the neurons within differing layers are arbitrary and the optimal number of each varies with the application area. It is the determination of these parameters which forms the research focus of this paper.
- The constituent layers of a group differ in the selection of elements attached to the inputs of their constituent neurons (termed the connectivity pattern). Again, this is a major parameter to be optimised within the network configuration.
- However, neurons within a given layer possess the same connectivity pattern relative to their position within the matrix and this is not a network parameter which may be modified.

The connectivity patterns for neurons of differing layers take various forms and hence represents an area for optimisation. Further, for each neuron, the values comprising the input set are calculated modulo the dimensions of the pattern. Therefore neurons which, for example, are situated at the right edge of a pattern and are virtually clamped to the 'x' bits on their right will actually be clamped to 'x' bits at the left side of the pattern. Some layers take inputs from non-local areas of the pattern. For the code we are using which has been parsed into a matrix this means that each layer will be looking at a slightly different 'picture' of the same sample, allowing for a high chance of recognition should there be a pattern.

The GCN network architecture employs RAM-based neurons with varying sized symbol sets which are the values stored within rather than more conventional Boolean symbol sets employed by alternative RAM-based networks (Howells, et al., 1994). The symbol set is extended to allow a symbol to represent each pattern class under consideration. For example, if ten pattern classes representing numerals were being considered, the symbols `0' through `9' could be employed representing the numerals 0 through 9 respectively. In the case of this paper, these 'symbols' are represented by the sensor data collected from the robot. As the network only takes binary as an input, this data has to be specially formatted – this is discussed in detail in a later section. These symbols are referred to as the *base* symbols of the network.

A comprehensive description of this architecture and its associated training and recognition algorithms may be found in (Howells, et al., 1995).

4 THE PROBLEM DOMAIN

The robot begins in an open area with a few static obstacles placed at strategic points in its path. Only one obstacle will be negotiated at any given time for these experiments. In order to correctly negotiate the robot around obstacles, the system must be able to identify not only where the object is in relation to the robot, but also which way to turn in order to avoid it. With data that could be in constant flux when in tight areas such as small corridors or cluttered environments, the ability to modify the neural network to deal with this is incredibly useful. These experiments should show that the network can identify an object in proximity and correctly

determine a new direction in order to avoid the obstacle. This is a simple task but the key aim is to demonstrate the meta-network's ability to improve the decision making ability of the robot by modifying the architecture of the network.

Weightless Neural Networks typically take binary as inputs and as such the data must to be parsed – this is shown in a later section. For this particular experiment, the network will be analysing the distances and determining a direction from this. As a comparison, a GCN network using randomly generated architectures will be employed to see how effective the meta-network is at improving the accuracy of the results.

5 META-NETWORK

As stated above, to combat the problem of finding a practically employable architecture for a weightless neural network, a genetic algorithm was chosen. The framework used in the present work is a layered process of an evolutionary search of ANNs, as illustrated in Figure 4.

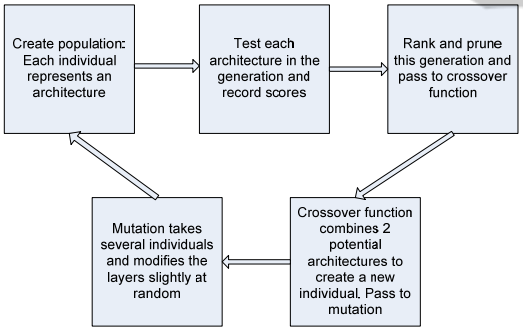

Figure 4: Basic Diagram depicting the various processes involved.

As mentioned in the introduction, architecture design is crucial to the successful application of a weightless ANN because it has significant impact on the network's information processing capabilities. With a variable number of layers, neurons per layer, and neuron placement, the number of potential architectures is potentially very large. For a particular learning scenario, a network with relatively few connections may imply it will be unable to perform the task due to its limited capability. Conversely, a network with a large number of connections may add noise to the training data and fail to generalise appropriately –a balance must be struck. To combat the problem of finding a practical architecture for a weightless neural network, the following Genetic Algorithm was employed.

There are several variables that can be modified when using the GCN architecture, including input size, number of layers, number of neurons (per layer), and the size of the training set size. For these experiments, the size of the training set and the input size were set to seven and 6x7 respectively. The reason for the input size is described in a later section. This paper investigates the use of a Genetic Algorithm to optimise the parameter configuration for employment in an obstacle avoidance task.

There were some inherent problems with using at standard genetic algorithm as a base however:

5.1 Inputs

Typically the inputs to a genetic algorithm are strings or numbers which are the parameters the genetic algorithm can modify. However, for this experiment, numbers would not suffice due to the complexity of the problem – It needed to modify 3 component parameters of information. The first is the number of layers, the second is the number of neurons within each of these layers, and the third is the placement of these neurons. As such, a custom input was defined as shown in Figure 5. On the left, Figure 5 shows 3 'layers'– each pair of zeros represents the relative coordinates for a neuron from which its inputs will be derived, remembering that the dimensions of the layer and input pattern are identical. As described in the previous section discussing the weightless neural architecture, the pattern 'wraps' around, meaning that neurons on the right side of the pattern are virtually clamped to those on the left. If a coordinate given exceeds the boundaries of the matrix, it simply wraps around. So the layer on the right in Figure 3 translates as a straight line of neurons for that layer for the element in the centre of the layer.

5.2 Initial Population

The initial population is created using a random generator for both the number or layers and how many neurons will be in each individual layer. Subsequently, tests are carried out on the data and the error rate is returned for each individual. The error rates are then multiplied by a factor of their complexity – each additional layer adds a 0.05 to a value that the error rate will be multiplied by, so that smaller networks with similar results will edge out those with larger networks. For example, if there are two architectures – A and B - each with an error

rate of 10%, but A has 3 layers while B has 5, A has a better fitness rating (10*1.15=11.5) than B (10*1.25=12.5) and as such, the genetic algorithm will favour it in selection. These fitness ratings are then ranked, and the bottom 50% are pruned.

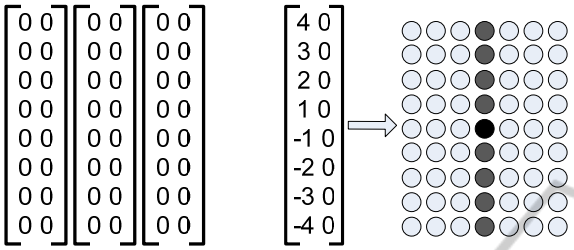

Figure 5: (Left) Shows input design for the genetic algorithm. (Right) Shows graphic translation of coordinates to layer.

5.3 Crossover

Crossover takes selected individuals and 'mates' them by taking the first half of each layer of the first individual and the second half of each layer of the second individual, as shown in figure 6. This creates a new individual (representing an architecture) which is then added to the next generation for testing.

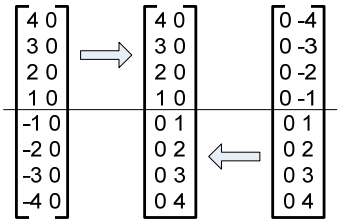

Figure 6: During Crossover, top of parent A combined with bottom half of parent B for each layer.

5.4 Mutation

Mutation is employed to a few individuals via slight modifications to create new architectures. This is achieved by adding a matrix of integers that range between -1 and 1, as shown in figure 7.

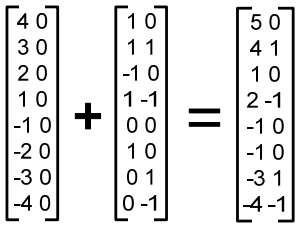

Figure 7 : Each layer has a matrix of the same size added to it. This layer has values between -1 and 1.

6 EXPERIMENTATION

The robot needed to be able to identify an obstacle and turn in the correct direction in order to avoid it. As such data needed to be collected from the sensors so that it could be analysed and parsed. If the object is a medium distance away for example, it should take less drastic action than if it was a short distance away.

Data was collected from the sensors during a test run of the robot which lasted for 37 seconds. 518 readings were taken during this time, or 74 per sensor. The data format is shown below in Table 1. From this simulated data was created representing different scenarios the robot would encounter.

Table 1: Example data collected.

Sensor ID	Distance (mm)
48	1982
50	2967
52	3001
54	4371
56	2489
58	1001
60	443

The distance needed to be converted into a Gray code (Gilbert, 1958) so that it could be fed into the, network. This was necessitated as it is required for codes representing similar values to possess similar binary encodings. This is not the case with natural binary due to the severe changes evident when going from certain numbers, for example 31 to 32 shown in figure 8.

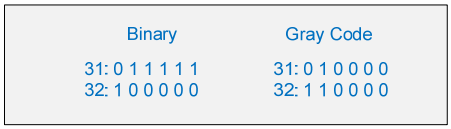

Figure 8: Difference between normal binary code and gray code.

Since the network works by finding similarities in the patterns, similar distances must be represented by similar encoded data, which is not the case in binary. Figure X shows how the data was quantized into 64 equidistant parts ranging between 0(0) – 4500(63), representing the range of distances found in the data. This number was then encoded using the above method for each of the sensors, forming a 6x7 matrix displayed in figure 9.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

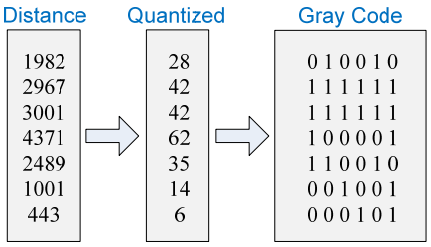

Figure 9: Showing how the data is encoded.

Each row in the above binary directly relates to a sensor and the distance recorded at that particular interval.

Five classes were created each representing a direction the robot should take. The reason for using five is to show that the network can discern the difference between taking a small change in direction as opposed to a large one. These were; left (-90˚ -45˚) slightly left (-44˚-1˚), straight (0˚), slightly right (1˚-44˚) and right (45˚-90˚). Each category was allocated a training set containing 7 encoded matrices. These included examples of an obstacle at varying distances from the sensor in different locations. A test set for each direction was also derived from the data collected were selected however, only 'simple' examples were taken, where there could only logically be 'one' obstacle (adjacent sensors 'hit').

7 RESULTS AND DISCUSSION

The meta-network was applied to the training data derived from the mobile robot. The initial population was set to 30, and the architectures were limited to a maximum of 12 layers. The reason for this is to ensure that the genetic algorithm doesn't create overly large, time consuming networks. The results show that the system finds a result that has an error rate of 4.75% in just 21 generations. The fitness value on the Y axis represents error rate multiplied by a factor of the architectures complexity, as described at the start of section 4.

To further determine the usefulness of this approach, the test set for each class was increased to 3, meaning that each architecture would see a total of 15 different scenarios. The results from this can be seen in figure 11, which show little change in the best individual.

However, as the generation average is much lower than that in figure 10, it is possible the initial system configuration was in a near optimal state, meaning that there was little room for improvement. A comparison can be seen in Table 2, which shows the differences between the experiments. The initial means differ quite substantially, meaning that the initial population from the second experiment out-performed those from the first.

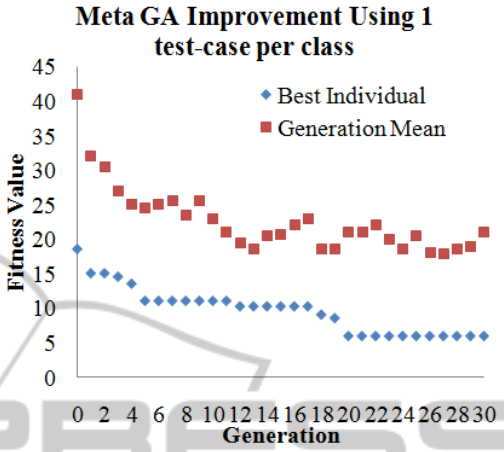

Figure 10: Results from initial experiments.

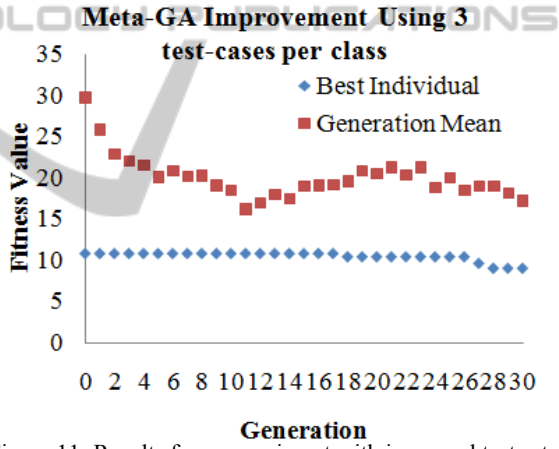

Figure 11: Results from experiment with increased test set.

Table 2: Comparison of results.

Comparison of experiments				
# of Test-cases	Initial Mean	Initial Best Individual	Final Individual	Standard Deviation of Best Individuals
1	41	18.5	5.9	3.4
3	29.8	10.9	9.1	0.6

It was found that increasing the number of generations past 30 did not yield better results, and it would eventually stop due to there being no improvement in fitness over a set number of generations. For both experiments the best architecture found had 5 layers, indicating a balance between network complexity and fitness (more

complex architectures with similar accuracies were discarded).

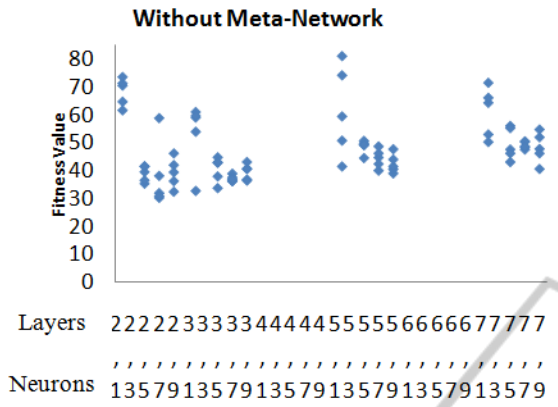

Figure 12: Using GCN without the meta-network.

When drawing a comparison with using the GCN network without the meta-network the differences are clear. Using the same data and test cases as the second experiment, the GCN was used on its own. The number of layers and the number of neurons per layer must be set manually, and a random architecture is derived from this. Figure 12 shows that over a multitude of different layer and neuron configurations, it fails to find an architecture that comes close to that displayed in figures 10 and 11.

8 CONCLUSIONS

The paper has investigated the use of a Genetic Algorithm to optimise the configuration of a simple weightless neural architecture. The results indicate that this approach possesses potential merit. A major advantage is the simple nature of both the parsed data input and the network being used.

ACKNOWLEDGEMENTS

This research is supported by the European Union ERDF Interreg V scheme under the SYSIASS Grant.

REFERENCES

Abraham, A., 2004, Meta learning evolutionary artificial neural networks. *Neurocomputing, Issue - 38 : Vol. - 56,* 0925-2312.

Abraham A., 2002, Optimization of Evolutionary Neural Networks Using Hybrid Learning Algorithms. *International Joint Conference on Neural Networks,* 0-7803-7278-6.

Bledsoe W. and Browning I., 1959, Pattern recognition and reading by machine. *AFIPS Joint Computer Conferences.* Boston, Massachusetts.

Gilbert E., 1958, Gray codes and paths on the n-cube. *BellSystem Technical Journal,* Vol. 37. 815-826.

Howells G., Fairhurst M. C., and Bisset D. L., 1994. BCN: an architecture for weightless RAM-based neural networks. *IEEE International Conference on Neural Networks.* Orlando, FL. 0-7803-1901-X.

Howells G., Fairhurst M.C., and Bisset D. L., 1995. GCN: the generalised convergent network. *International Conference on Image Processing and its Applications.* Edinburgh, 0-85296-642-3.

Kordík, P., Koutník, J, Drchala, J., Kováříka, O., Čepeka, M. and Šnoreka, M., 2010. Meta-learning approach to neural network optimization. *Neural Networks*, Vol. 23. - 0893-6080.

McElroy B. and Howells G., 2010. Evaluating the Application of Simple Weightless Networks to Complex Patterns *International Conference on Emerging Security Technologies (EST),* 978-0-7695-4175-4.

Slowik A. and Bialko M., 2008. Training of artificial neural networks using differential evolution algorithm. *Conference on Human System Interactions*, 978-1-4244-1542-7.

Yao, X., 1999 Evolving Artificial Neural Networks. *Proceedings of the IEEE,* Vol. 87. - 0018-9219 .