24

Evolutionary Artificial Neural Networks in Neutron Spectrometry

José Manuel Ortiz-Rodríguez[1,3], Ma. del Rosario Martínez-Blanco[2] and Héctor René Vega-Carrillo[2]

Unidades Académicas: [1]*Ingeniería Eléctrica,*
[2]*Estudios Nucleares, Universidad Autónoma de Zacatecas.*
[3]*Depto. de Electrotecnia y Electrónica, Escuela Politécnica Superior, Córdoba España.*
México

1. Introduction

1.1 Artificial neural networks

Artificial Neural Networks (ANN), are highly simplified models of the brain processes (Graupe, 2007; Kasabov, 1998). An ANN is a biologically inspired computational model which consists of a large number of simple processing elements called neurons, units, cells, or nodes which are interconnected and operate in parallel (Galushkin, 2007; Lakhmi & Fanelli, 2000). Each neuron is connected to other neurons by means of directed communication links, which constitute the neuronal structure, each with an associated weight (Dreyfus, 2005). The weights represent information being used by the net to solve a problem. Figure 1 shows an abbreviated notation for an individual artificial neuron, which is used in schemes of multiple neurons (Beale et al., 1992). Here the input **p**, a vector of R input elements, is represented by the solid dark vertical bar at the left. The dimensions of **p** are shown below the symbol **p** in the figure as $Rx1$. These inputs post multiply the single-row, $R-column$ matrix **W**. A constant 1 enters the neuron as an input and is multiplied by a bias b. The net input to the transfer function f is n, the sum of the bias b and the product **Wp**. This sum is passed to the transfer function f to get the neuron's output a.

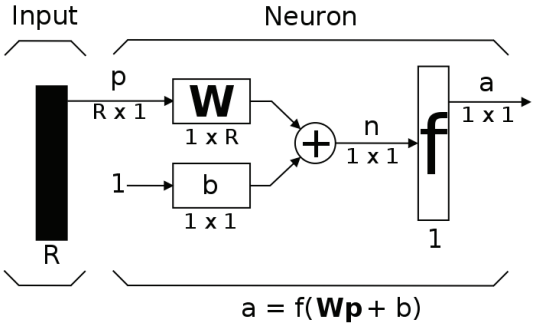

Fig. 1. Abbreviated notation for an individual artificial neuron

Although a single neuron can perform certain simple information-processing functions, a single node is insufficient for many practical problems, and networks with a large number of nodes are frequently used. A single layer of neurons having different transfer functions can be created simply by putting the neuron shown earlier in parallel (Kishan et al., 2000). All the neurons would have the same inputs, and each neuron would create the outputs, however, a layer of neurons is not constrained to have the number of its inputs equal to the number of its neurons, and it is common for the number of inputs to a layer to be different from the number of neurons. To describe networks having multiple layers, it needs to make a distinction between weight matrices that are connected to inputs and weight matrices that are connected between layers. It also needs to identify the source and destination for the weight matrices. Weight matrices connected to inputs are called *input weights* (IW), whereas weight matrices coming from layer outputs are called *layer weights* (LW). Further, superscripts are used to identify the source (second index) and the destination (first index) for the various weights and other elements of the network.
Figure 2 shows an abbreviated notation of a single layer of neurons. As can be seen from this figure, the weight matrix connected to the input vector **p** is labeled as an input weight matrix ($\mathbf{IW}^{1,1}$). The input vector elements enter the network through the weight matrix **W**. The row indices on the elements of matrix **W** indicate the destination neuron of the weight, and the column indices indicate which source is the input for that weight. Thus, the indices in $w_{1,2}$ say that the strength of the signal from the second input element to the first (and only) neuron is $w_{1,2}$. Elements of layer 1, such as its bias, net input, and output have a superscript 1 to say that they are associated with the first layer.

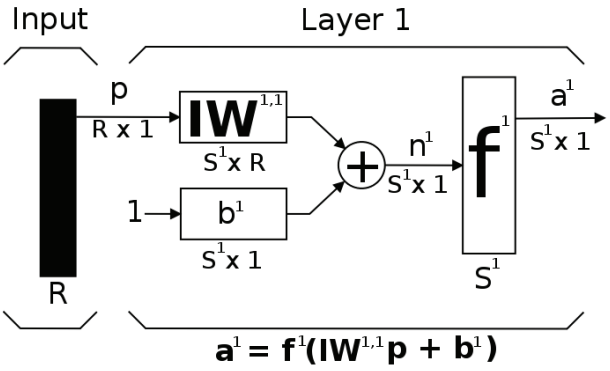

Fig. 2. Abbreviated notation of a single layer of neurons

Multiple-layer networks are quite powerful and can solve more complicated problems than can single-layer nets. A multilayer neural network consists of a combination of neurons or nodes and synaptic connections, which are capable of passing data trough multiple layers (Fausett, 1993). Each layer has a weight matrix **W**, a bias vector **b**, and an output vector **a**. The layers of a multilayer network play different roles, i.e., the x-y-z neural network structure refers to number of neurons in the input, hidden and output layers respectively. Input layers receive input signal or values from an external source, output layer transmit the result of the neural network processing and hidden layer(s) make up the internal layer(s) between input and output node layers (Haykin, 1999). To distinguish between the weight matrices, output vectors, etc., as mentioned previously, the number of the layer is appended as a superscript to the variable of interest.

Figure 3 shows a three-layer network using abbreviated notation. From this figure can be seen that the network has $\mathbf{R}^1$ inputs, $\mathbf{S}^1$ neurons in the first layer, $\mathbf{S}^2$ neurons in the second layer, etc. A constant input 1 is fed to the bias for each neuron. The outputs of each intermediate layer are the inputs to the following layer. Thus layer 2 can be analyzed as a one-layer network with $\mathbf{S}^1$ inputs, $\mathbf{S}^2$ neurons, and an $\mathbf{S}^2 x \mathbf{S}^1$ weight matrix $\mathbf{W}^2$. The input to layer 2 is $\mathbf{a}^1$; the output is $\mathbf{a}^2$. Now that all the vectors and matrices of layer 2 have been identified, it can be treated as a single-layer network on its own. This approach can be taken with any layer of the network.

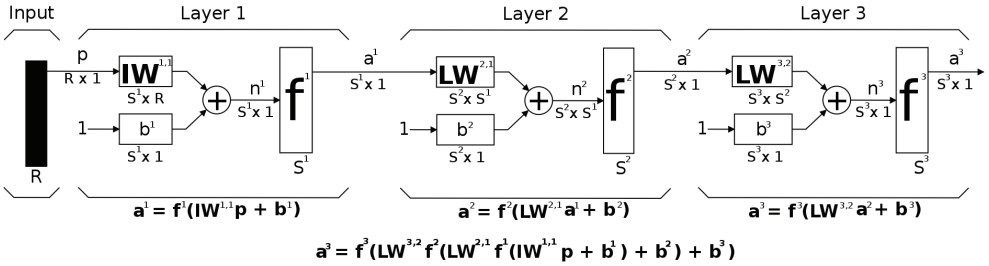

Fig. 3. Three-layer neural network using abbreviated notation

The arrangement of neurons into layers and the connection patterns within and between layers is called the *net architecture* (Jain et al., 1996; Zupan, 1994). According to the absence or presence of feedback connections in a network, two types of architectures are distinguished:

- **Feedforward architecture**. There are no connections back from the output to the input neurons; the network does not keep a memory of its previous output values and the activation states of its neurons; the perceptron-like networks are feedforward types.
- **Feedback architecture**. There are connections from output to input neurons; such a network keeps a memory of its previous states, and the next state depends not only on the input signals but on the previous states of the network; the Hopfield network is of this type.

The central idea of neural networks, where w and b are both adjustable parameters of the neuron, is that such parameters can be adjusted by means of learning or training, so that the network exhibits some desired or interesting behavior (Fausett, 1993; Graupe, 2007; Haykin, 1999; Kasabov, 1998; Kishan et al., 2000; Lakhmi & Fanelli, 2000). Learning is not an individual ability of a single neuron, it is a collective process of the whole neural network and a result of a training procedure. Training is the algorithmic procedure whereby the parameters of the neurons of the network are estimated, in order for the neural network to fulfill, as accurately as possible, the task it has been assigned.

As shows figure 4, an ANN is trained so that a set P of input vectors produces the desired, or at least a consistent, set of target output vectors T, or the network learns about internal characteristics and structures of data from a set P. The set P used for training a network is called *training set* and the elements $\mathbf{p}$ of this set P are called *training examples*. The training process is reflected in changing the connection weights of the network. The default performance function for feedforward networks is the mean square error (*mse*), which is the average squared error between the network outputs a and the target outputs T. During training, the network weights should gradually converge to values such that each input vector $\mathbf{p}$ from the data set training causes a desired output vector $\mathbf{t}$ produced by the network.

Learning occurs if after supplying a training example, a change in at least one synaptic weight takes place. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined by the experimenter.

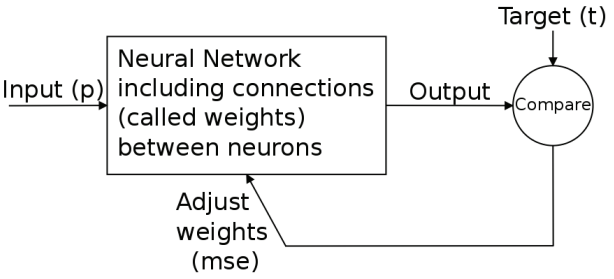

Fig. 4. Training procedure of neural networks

The learning ability of a neural network is achieved through applying a learning or training algorithm. Training algorithms are mainly classified into three groups:

- **Supervised**. This training algorithm has been the most used mainly because the training examples comprise input vectors **p** and the desired target output vectors **t**. Training is performed until the neural network "learns" to associate each input vector **p** to its corresponding and desired output vector **t**.
- **Unsupervised**. Only input vectors **p** are supplied; the neural network learns some internal features of the whole set of all the input vectors presented to it.
- **Reinforcement learning**. Sometimes called reward-penalty learning, is a combination of the above two paradigms; it is based on presenting input vector **p** to a neural network and looking at the output vector calculated by the network. If it is considered "good," then a "reward" is given to the network in the sense that the existing connection weights are increased; otherwise the network is "punished," the connection weights, being considered as "not appropriately set," decrease. Thus reinforcement learning is learning with a critic, as opposed to learning with a teacher.

Several different training algorithms for feedforward networks use the gradient of the performance function to determine how to adjust the weights to minimize performance. The gradient is determined using a technique called Back-Propagation (BP), which involves performing computations backward through the network, which refine one of the principal components of neural networks: the connection weights. The BP computation is derived using the chain rule of calculus (Taylor, 1993). BP was created by generalizing the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions (Fausett, 1993; Graupe, 2007; Haykin, 1999; Kasabov, 1998; Kishan et al., 2000; Lakhmi & Fanelli, 2000).
Despite the apparent success of the BP learning algorithm, there are some aspects, which make the algorithm not guaranteed to be universally useful.

- One of the problems of BP is that it can get stuck in a local minimum. This is not too bad if the local minimum turns out to be close to the global minimum, but there is no guarantee that is the case.

- Another problem associated with BP is that the place at which one starts on the error surface (which is determined by the initial weight settings, which are often random) determines whether or not a good or the best solution is found. When a solution is found that performs well on the training set, the network might still perform badly on the overall set of input, if the training set was not representative.
- A last problem is the occurrence of interference. This occurs when a network is supposed to learn similar tasks at the same time. Apart from the fact that smaller networks are unable to learn to many associations, they simply are full after a certain amount of learned associations, there is also the danger of input patterns being so hard to separate, that the network can't find a way to do it.

ANN has been well known for its effectiveness in representing nonlinear process system (Apolloni et al., 2009). The power of neural computation comes from connecting neurons in networks, and the way these nodes are connected, determines how computations proceed, and constitutes an important early design decision by a neural network developer. However, ANN can not solve all problems in the real world. One of the biggest drawbacks in the use of neural networks nowadays is the problem of finding an appropriate structure for a given task, mainly because it is very hard to know beforehand the size and the structure of a neural network one needs to solve a given problem (Ortiz-Rodríguez et al., 2006; Packianather & Drake, 2004; Packianather et al., 2000; Peterson et al., 1995). An ideal structure is a structure that independently of the starting weights of the net, always learns the task, i.e. makes almost no error on the training set and generalizes well.
The problem with neural networks is that a number of parameter have to be set before any training can begin. However, there are no clear rules how to set these parameters. Yet these parameters determine the success of the training. Among the limitation of ANN, the followings should be given added emphasis:

1. **Network architecture**. There is a lack of fixed rule or systematic guideline for optimal ANN architecture design. Since there is no a prior knowledge about the problem complexity, the network architecture was typically set arbitrarily. The network topology was often determined by trial and error. This subjected the network to performance uncertainties since the size of network influence the network performance: too small a network cannot learn well, but too large may lead to overfitting. Thus, algorithms that can find appropriate network architecture are needed. This includes the determination of optimum number of neuron in each layer as well as number of hidden layer needed.
2. **Training algorithm**. The best training algorithm still cannot be singled out for general neural network. Although BP algorithm has been widely used, it does not guarantee the global optimal solution. The training may result in ANN model that is only accurate in the same operating zones as in the training data set but inaccurate in others. Besides, the selection of some parameters in BP training, such as learning rate and momentum, also lacks of systematic guideline.
3. **Training data**. The quality and quantity of training data is an important issue for ANN modeling. Usually, the success of ANN relies heavily on a large amount of data, but this demand more computing time for training. In order to reduce the amount of data whilst maintaining the model quality, the data used must be carefully selected to ensure that they are sufficiently rich. This demands project understanding on the process involved. Additionally, to eliminate noise and outliers, process data may require pre-processing prior to application in neural network model development.

4. **Training set**. Since it is normally impossible to present a network with all possible inputs, we only present it with part of it, the training set. This set has to be chosen in such a way that the network also gives correct output for an input that was not in the training set. If the network also responds well to inputs that were not in the training set, it is said to generalize well. Often an ANN is trained with one set of patterns (the training set) and tested with another (the test set). If the training set was not a good representation of all possible inputs, the network probably will not perform too well on inputs that are not in the training set. Generalization is quite similar to interpolation in mathematics.
5. **Process relationship**. Being black-box method for modeling, ANN is criticized for unable to explain and analysis the relationship between inputs and outputs. This may cause difficulties in interpreting results from the network.

All of these limitations have motivated researchers to generate ideas of merging or hybridizing ANN with other approaches in the search for better performance. Some of the available schemes include expert systems, statistical methods (Ortiz-Rodríguez et al., 2006; Packianather & Drake, 2004), fuzzy logic (Chennakesava, 2008), wavelet transform and as well as Neuro Evolutionary (NE) Approaches (Floreano & Mattiussi, 2008; Leardi, 2003; Melin & Castillo, 2005; Yao, 1993). In this work, the use of NE is considered.

1.2 Neuro Evolution

Neuro Evolution (NE) leverages the strengths of two biologically inspired areas of Artificial Intelligence (AI) (Coppin, 2004; Luger, 2005; Rasskin-Gutman, 2009): Artificial Neural Networks (ANN) and Evolutionary Algorithms (EA) (Gen et al., 2009; Munakata, 2008; Rothlauf, 2006; Whiteson & Stone, 2006). EA are stochastic and adaptive population-based search methods based on the principles of natural evolution. They involve a population of individuals represented in a genotypic form (chromosomes/genotypes), each of which is a potential solution to the problem. Each individual has a fitness score associated with it, and individuals with better fitness scores are better solutions. Between one generation and the next, individuals are selected from which to create offspring by applying mutation and crossover operators. Generally selection is biased towards fitter individuals, and unpromising areas of the search space are abandoned with the loss of poorer performing individuals from the population over time. EA encompass Genetic Algorithms (GA), Evolutionary Programming (EP) and Evolution Strategies (ES) (Affenzeller et al., 2009; Goldberg, 1989; Haupt & Haupt, 2004; Mitchell, 1998; Periaux & Winter, 1995).

Although EA and ANN have in common that they are general search strategies, they vary in their range. EA perform a more global search than ANN with BP. Figure 5, illustrates the convergence of the strategies. BP takes more time to reach the neighborhood of an optimal solution, but then reaches it more precisely. On the other hand, EA investigate the entire search space. Hence, they reach faster the region of optimal solutions, but have difficulties to localize the exact point. This happens, because for the final "fine-tuning" of the solution relies almost entirely on mutation. Combining both strategies seems to be the best thing to do (Floreano & Mattiussi, 2008; Leardi, 2003; Melin & Castillo, 2005; Yao, 1993), and the NE approach outperforms EA as well as ANN in finding a satisfying solution.

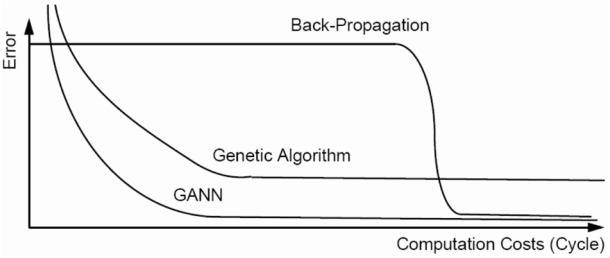

Fig. 5. Convergence of ANN and EA

Because GAs sample many points in the search space simultaneously, they are less susceptible to local minima than single solution methods (BP), and are capable of rapidly locating high payoff regions of high dimensional search spaces. Figure 6, shows a hypothetical fitness landscape to illustrate how a GA operates. The fitness of each individual in the population is represented by its position on the landscape. In a single solution method, such as BP training, if the initial search point (the yellow circle) happens to fall in the neighborhood of a local maxima, the algorithm can become trapped because it has only local information with which to make a next guess and improve the solution. Therefore, it will climb the gradient towards the local maxima. In a GA, although some individuals (the red circles) may reside near local maxima, it is less likely to get trapped because the population provides global information about the landscape. There is a better chance that some individual will be near the global maxima, and the genetic operators allow the GA to move the population in large jumps to focus the search in the most fruitful regions of the landscape.

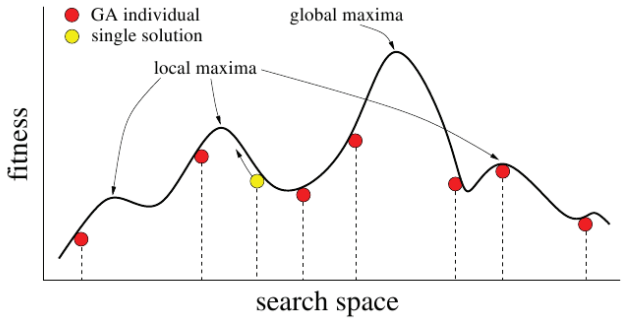

Fig. 6. Fitness landscape which illustrates how a GA operates

As is showed in figure 7, the basic idea of NE is to search the space of neural network policies directly by using a GA. From this figure can be seen that each chromosome is transformed into a neural network phenotype and evaluated on the task. The agent receives input from the environment (observation) and propagates it through its neural network to compute an output signal (action) that affects the environment. At the end of the evaluation, the network is assigned a fitness according to its performance. The networks that perform well on the task are mated to generate new networks.

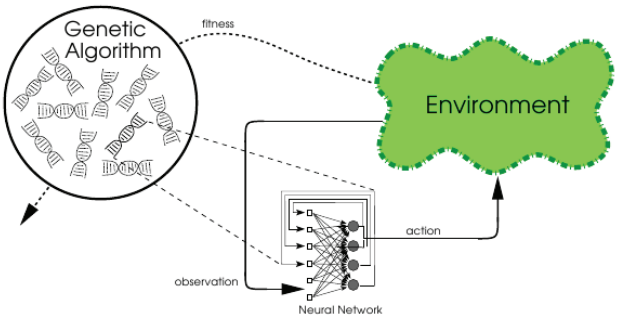

Fig. 7. Neuro Evolution approach

GA were introduced by John Holland in 1975 (Goldberg, 1989), and are a class of stochastic search procedures founded on the principles of natural selection. Unlike conventional search methods that iteratively improve a single solution, a GA maintains a set or population of candidate solutions that sample the search space at multiple points. These solutions are encoded as strings called chromosomes that represent the genotype of the solution. The chromosomes are usually composed of a fixed number of genes that can take on some set of values called alleles.

1.2.1 Genetic algorithms

GA belongs to a class of population-based stochastic search algorithm that are inspired from principles of natural evolution known as EA. Similar to other EA algorithms, GA is based on the principle of ”survival of fittest”, as in the natural phenomena of genetic inheritance and Darwinian strife for survival. In other words, GA operates on a population of individuals which represent potential solutions to a given problem. Mimicking the biological principles in nature, a single individual of a population usually is affected by other individuals as well as the environment. Normally, the better an individual performs under these competitive conditions the greater is the change for the individual to survive and reproduce. This in turn inherits the good parental genetic information. Hence, after several generations, the bad individual will be eliminated and better individuals are produced.
The most important terms used in the GA, analogous to the terms used to explain the evolutionary processes, are:

- **Gene**. A basic unit, which controls a property of an individual.
- **Chromosome**. A string of genes; it is used to represent an individual, or a possible solution of a problem in the solution space.
- **Population**. A collection of individuals.
- **Operation of crossover or mating**. Substrings of different individuals are taken and new strings (offsprings) are produced.
- **Mutation**. Random change of a gene in a chromosome.
- **Fitness or goodness function**. A criterion which evaluates each individual.
- **Selection**. A procedure for choosing a part of the population that will continue the process of searching for the best solution, while the other part of the population "dies".

Although the evolutionary principle of GA is similar with other EA varieties, its implementation is different. Initially, the evolutionary process of GA starts with the creation of the blind random population defined by the problem statement. This is followed by the series of activity including solutions encoding, fitness evaluation, selection, genetic operator alteration and replacement which are iteratively executed until the stopping criterion is satisfied. Figure 8 outlines a typical evolutionary structure of GA.

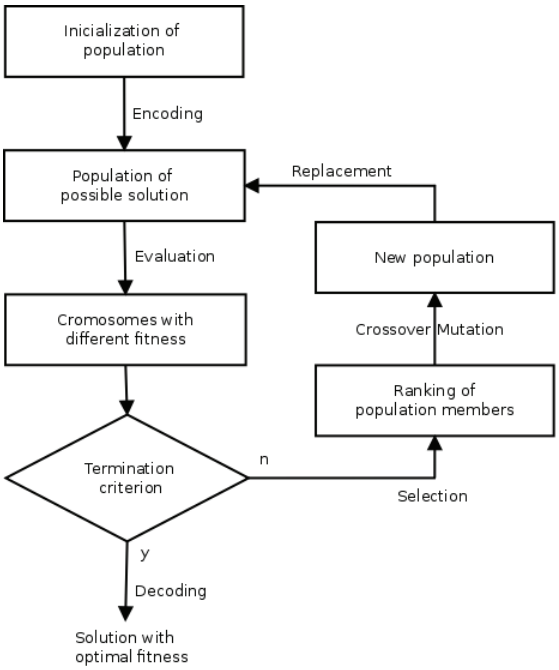

Fig. 8. Evolutionary structure of a GA

As a population-based search algorithm, information exchange among the individual is particularly important for GA. Such mechanisms are achieved by genetic operators, the more common ones are reproduction, crossover and mutation. Following a process analogous to natural evolution, each genotype is transformed into its phenotype and evaluated on a given problem to assess its fitness. Those genotypes with high fitness are then mated using crossover and mutation at low levels to produce new solutions or offspring. Figure 9 illustrates how crossover and mutation work. Crossover produces two offspring from two parents by exchanging chromosomal substrings on either side of a random crossover point, each offspring is a concatenation of contiguous gene segments from both parents. When an offspring is mutated, one of its alleles is randomly changed to a new value. By mating only the most fit individuals, the hope is that the favorable traits of both parents will be transmitted to the offspring resulting in a higher scoring individual, and eventually leading to a solution. In general, GA is applicable to a wide range of problem in learning and optimization. They can deal with complex problems which are multimodal and discontinuous. GA has two prominent features that are different from other search algorithms. First, it is population-based. Second, there is information exchange among individuals in a population.

Primarily, GA was designed to optimally solve sequential decision processes more than to perform function optimization but over the years, it has been used widely in both learning and optimization. For these reasons, GAs are well suited for searching the space of neural networks. Instead of training a network by performing gradient descent on an error surface, the GA samples the space of networks and recombines those that perform best on the task in question.

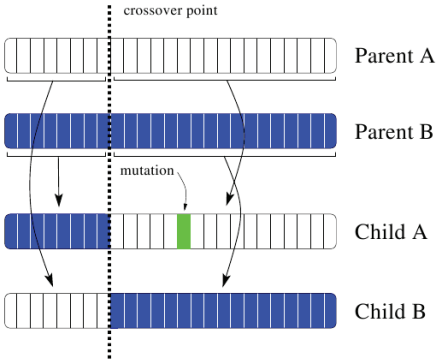

Fig. 9. Crosover and mutation genetic operators

Today neural networks can be trained to solve problems that are difficult for conventional computers or human beings, and have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, speech, vision, and control systems. Recently, the use of ANN technology has been applied with success in the research area of nuclear sciences, mainly in the neutron spectrometry and dosimetry domains.

1.3 Artificial neural networks and neutron spectrometry

Nowadays, neutrons are widely used in many fields of both research and technology (Wielunski et al., 2008b). Reliable determination of neutron doses is still an issue in the field of radiation protection (Lacoste et al., 2007). In recent years, the characterization of ionizing radiation fields in workplaces is one of the challenging activities over the world (Mazrou et al., 2008). The workers subject to these radiations especially those who are submitted to neutron risk have to be well monitored and protected according to relevant national regulations which are more and more restrictive. As a result, there is an increasing demand in the field of radiation protection to quantify these various neutron fields and to determine the radiation doses involved (Mazrou et al., 2008). The dosimetry of neutron radiation is one of the most complicated tasks in radiation protection (Wielunski et al., 2008a), mainly because is a complex technique (Bedogni et al., 2007), and highly neutron energy dependent, and a precise knowledge on neutron spectrometry is highly essential for all dosimetry-related studies as well as many nuclear physics experiments. In consequence, it becomes necessary to develop additional measuring techniques to enhance the actual workers monitoring systems.
The term radiation spectrometry can be used to describe measurements of the intensity of a radiation field with respect to energy, wavelength, momentum, mass, angle of incidence or any other related quantity (Thomas, 2004; Vega-Carrillo et al., 2009a; 2010). The distribution of the intensity with one of these parameters is commonly referred to as the spectrum, i.e.,

the measurement of neutron energy spectra is the variation of the intensity of these radiations with energy (Vega-Carrillo et al., 2009b).
One of the suitable approaches to improve the knowledge on neutron radiation fields to which individuals are exposed during their work, is based on spectrometric measurements (Brooks & Klein, 2002; McDonald et al., 2002; Thomas, 2004). The measured yield of a neutron source is the convolution of the neutron energy distribution with the response function of the spectrometer summed over the interaction energy range. As is shown in equation 1, which represents the discrete form of the Fredholm's integral equation of the first kind (Vega-Carrillo et al., 2006).

$$C_j = \sum_{i=1}^{N} R_{i,j}\Phi_i - - - > j = 1, 2, ..., m \tag{1}$$

where C_j is j^{th} detector's count rate; $R_{i,j}$ is the j^{th} detector's response to neutrons at the i^{th} energy interval; Φ_i is the neutron fluence within the i^{th} energy interval and m is the number of spheres utilized. Equation 1 is an ill-conditioned problem wich has an infinite amount of solutions
Although there is a wide range of different devices used for neutron spectrometry, the majority of the instruments can be grouped together into a small number of broad categories, each one based on a common underlying technique (Matzke, 2003; McDonald et al., 2002; Thomas, 2004). Among the many available neutron spectrometry techniques, the multisphere or Bonner sphere spectrometer (BSS) system is the most used for radiation protection purposes (El Messaoudi et al., 2004; Lacoste et al., 2004; Vylet, 2002), due to advantageous characteristics as wide energy range (from thermal to GeV neutrons), large variety of active or passive thermal sensors allowing adapting the sensitivity to the specific workplace, good photon discrimination and simple signal management. Disadvantages are the poor energy resolution, which does not allow appreciating fine structures as narrow peaks, the weight, and the need to sequentially irradiate the spheres, requiring, in general, long exposure periods (Bedogni et al., 2007).
The BSS consists of a thermal neutron sensor such as $^6LiI(Eu)$, which is placed at the centre of a number of moderating spheres of different diameter. With the BSS, neutron spectrum can be obtained, however, the derivation of the spectral information is not simple. The unknown neutron spectrum is not given directly as a result of the measurement. The BSS response matrix, the count rates and the neutron spectrum are related trough the equation 1. The most delicate part in the neutron spectrometry based on the BSS, is the unfolding process. The unfolding spectra of the neutrons measured consist on establishing the rate of energy distribution of fluency $\phi(E)$, known as the response matrix, $R_{i,j}$, and the group of carried out measures, C_j. Because the number of unknowns overcome to the number of equations, this is an ill-conditioned system and has an infinite number of solutions. The procedure of selecting the solution that has meaning for the problem type, is part of the unfolding process (Vega-Carrillo et al., 2005; 2006). The spectral information needs to be unfolded from the BSS system detector responses by using a suitable computational code, most of them are based in some of these methods: least square, iterative (Bedogni et al., 2007; Miller, 1993), bayesian and maximum entropy (Reginatto & Zimbal, 2008), and Monte-Carlo (Vega-Carrillo et al., 2007a).
The current interest in the neutron spectrometry problem, has stimulated the development of diverse unfolding procedures which try to obtain a better energy resolution through the reconstruction of the spectrum. During the past decades have been carried out intents to develop new neutron spectra unfolding codes like BUNKIUT (Miller, 1993), FRUIT (Bedogni

et al., 2007), MAXED, UMG (Roberts, 2007), etc., to attain improved energy resolution through spectrum unfolding. However, these methods still present the serious drawback of requiring a very expert user for their operation and the necessity to provide an initial guess spectrum for the deconvolution of the spectrum. To overcome these drawbacks, alternative approaches have been studied and proposed, to make an efficient neutron dosimetry, and several unfolding procedures combined with various types of experimental methods have been reported such as Genetic Algorithms (GA) (Freeman et al., 1999; Mukherjee, 2004), and Artificial Neural Networks (ANN) (Vega-Carrillo et al., 2007b; 2005; 2006; 2009a; 2010).
Many of the previous studies in neutron spectrometry and dosimetry by using the ANN approach have found serious drawbacks in the ANN design process itself, mainly in the proper determination of the structural and learning parameters of the networks being designed (Ortiz-Rodríguez et al., 2006). These parameters are significant contributing factors to the ANN performance, however, the optimal selection of these parameters follows in practical use no rules, and their value is at most arguable, mainly because they are generally heuristically chosen by using the trial and error technique, which produces poor artificial neural networks with low generalization capacity and poor performance. For the anterior, the nuclear research community needs approaches that implement ANN models faster than what is currently available. In consequence, more research has been suggested in order to overcome these drawbacks (Bedogni et al., 2007; Vega-Carrillo et al., 2007b; 2006; 2010).
At present, one promising technique to design the structural and learning parameters of ANN is by introducing adaptation of network training using EA. EA seems to be a proper alternative to solve the ANN optimization problem and can be used to assist in the ANN design and training (Ortiz-Rodríguez et al., 2008; 2009b; 2010a). However, as a novel approach in the nuclear sciences area, the lack of information and tools for the analysis of the results obtained whit these new technologies, makes difficult the work in this research area.
The aim of the present work is focused in analyzing the intersection of ANN and GA, analyzing like it is that GA can be used to help in the design processes and training of ANN, i.e., in the optimum selection of the structural and learning parameters of ANN, improving its generalization capacity, in such a way that the neural network designed is able to unfold in an efficient way neutron spectra, starting only from the count rates obtained with a BSS system.
Because the novelty of Evolutionary Artificial Neural Networks (EANN) technology in neutron spectrometry, lack of tools for the analysis is observed. For this reason, an unfolding code based on EANN technology, devoted to the operational workplace neutron monitoring, would be of great help to the radiation protection community. With this purpose, in this work a new computer tool based on EANN technology called "Neutron Spectrometry and Dosimetry based on Evolutionary Artificial Neural Networks" (NSDEann), was developed in a customized front end user capable to unfold neutron spectra and to simultaneously calculate 13 equivalent doses, by using only as input data the count rates coming of a BBS system, in just a few seconds if compared with the time spent with the classical techniques, not being needed a priori information about the spectra being calculated.

2. EANN in neutron spectrometry

EANN technology was used for the modeling and optimization of ANNs capable to solve the neutron spectra unfolding problem, starting from the count rates coming from a BSS with a $^{6}LiI(Eu)$ thermal neutron detector, 7 polyethylene spheres of $0, 2, 3, 5, 8, 10, and, 12$ inches of diameter respectively, and the $UTA4$ response matrix expressed for 31 energy bins (Vega-Carrillo et al., 2009a). In this study, two neutron spectra were produced using a $^{239}PuBe$

neutron source (Vega-Carrillo et al., 2009b). A neutron spectrum was produced by the $^{239}PuBe$ neutron source located at 100 cm of distance, in an open space at 200 cm above floor level, and was measured using the BSS previously described. Then, the same neutron source was inserted in a cylindrical container with water, and the neutron spectrum was measured again. The count rates measured in both cases with the BSS, were utilized to unfold the neutron spectra and to calculate the dosimetric features using four EANN designed for each case, varying some GA and ANN parameters during training stage. To analyze the performance and generalization capability of the evolved networks, a new neutron spectra unfolding code denominated "Neutron Spectrometry and Dosimetry based on Evolutionary Artificial Neural Networks" (NSDEann), was used. Also, the spectra were unfolded with the BUNKIUT code and UTA4 response matrix (Vega-Carrillo et al., 2009a), and compared with that obtained with NSDEann code using a computer tool known as "Neutron Spectrometry and Dosimetry Tool Box" (NSDTB) (Ortiz-Rodríguez et al., 2009a;b; 2010b).
The general idea of combining EA and ANN is illustrated in figure 10. The EANN approach is based on a fundamental cyclic process which consists of:

1. Creating an initial population of genotypes (genetic representations of the ANN).
2. Building neural networks (phenotypes) based on the genotypes.
3. Training and testing the neural networks to determine how fit they are.
4. Comparing the fitness of the networks and keeping the best.
5. Selecting those networks in the population which are better, discarding those which aren't.
6. Refilling the population back to the defined size.
7. Pairing up the genotypes of the neural networks.
8. Mating the genotypes by exchanging genes (features) of the networks.
9. Mutating the genotypes in some random fashion; Then returning back to step (2) and continuing this process until some stopping criteria is reached or manually stops the process.

Through the process described previously, the better networks survive and their features carry forward into future generations and are combined with others to find better and better networks for the problem considered. This genetic search capability is much more effective than random searching, as the genetic process of recombining features vastly improves the speed of identifying highly fit networks.
To train the evolved networks, a data set of 187 neutron spectra, compiled by the International Atomic Energy Agency (IAEA) (IAEA, 2001) was used (Iñiguez & Vega-Carrillo, 2002; Vega-Carrillo et al., 2005; 2006). The ANN genetically evolved were designed by meas of the NeuroGenetic Optimizer (NGO) software (BiocompSystems, 2010). In the use of NGO, there are 10 working steps as follows:

1. **Initial population**. The first step is started from building new population consisted of chromosomes derived from random sampling by having total chromosomes as population size.
2. **Decode**. This step is involved with decoding heredity from chromosomes derived from population in the 1^{st} step by transforming them in hidden neuron in the hidden layer of ANNs.

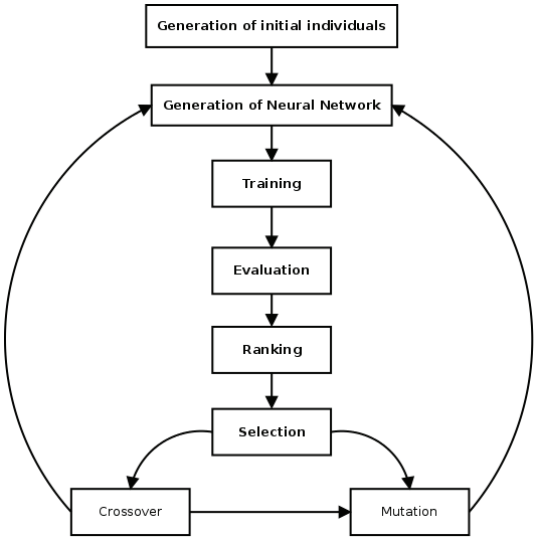

Fig. 10. General approach of Evolutionary ANN

3. **Train Back-propagation Neural Network**. In this step, network for learning would be derived from the past data by assigning numbers of hidden neuron as equal as the value from decoding. Before data could be applied in learning, data must be made into normal design. Afterwards, data must be divided into 2 set, one in training, another in testing. Learning process is the reverse learning process with variance that could be reversed back to adjust weight so that variance may be reduced.
4. **Fitness evaluation**. In this step, calculation is done to find variance of network which considered proper fitness value of heredity by using the Root-Mean-Square Error of Prediction (RMSEP) as objective function. RMSEP is the square root of the sum of the squared differences between the observed and predicted values for all observations in the test set divided by the number of such observations, to estimate the prediction error, as ishowed in the following equation:

$$RMSE = \sqrt{\frac{1}{n-1}\Sigma_{i=1}^{n}(y_i - \widehat{y_i})^2} \tag{2}$$

where n is the number of trials, y_i is the measured values of each response and $\widehat{y_i}$ is the neural model output.

5. **Stop criteria**. This is the step to check when the design stops working by setting up conditions for numbers of result compilation. If answers in each round are still stable, it should stop working. For examples, stop at 10 rounds of compilation or receiving duplicate answers 5 times in the row. If the stop conditions are real, operation may stop at step 10. On the contrary, for unreal stop conditions, proceed further with step 6.
6. **Selection**. In this step, two chromosomes with the least fitness values are selected from population to be the breeders.
7. **Crossover**. This is the step for crossing species by exchanging genes among selected breeders in Step 6 for offspring.

8. **Mutation**. After deriving at offspring with 2 chromosomes after crossover, mutation would be done by randomly selected position of gene with the possibility of mutation. Then, after randomly selected position, value of gene would have opposite value.
9. **Replacement**. The offspring with fitness value would take place of the suitable chromosome in order to derive at new group of population before going back to 2^{nd} step.
10. **Stop**. After the stop conditions had been verified, the operation of model would stop with only the best network design had been collected for being used for measuring effectiveness of network.

The following steps were carried out for developing four EANN models with NGO:

1. **Identify data by dividing into 2 sets**. Input layer data set, comprised for the count rates coming from polyethylene spheres of BSS and, output layer data set, comprised for neutron spectra unfolded and equivalent doses.
2. **Specifying neuron in the structure of ANNs**. Input layer has 7 neurons, corresponding to spheres of BSS, hidden layer(s) use GAs to specify neurons and, output layer has 44 neurons, first 31 are for the neutron spectra unfolded and the remaining 13 for different equivalent doses (Vega-Carrillo et al., 2010).
3. **Input data classification**. A data set with 187 neutron spectra was used. Training and testing data sets were created by dividing the whole data set, by using a random procedure, into 80% for training and 20% for testing.
4. **Neural Parameters**. In this work, ANNs with supervised learning through BP were chosen, the net architecture was optimized with the application of GAs to find suitable networks. Sigmoid function was employed as transfer function for hidden layer(s) and linear for output layer. The optimizer of the network structure, used all inputs, and searched the neural architecture.
5. **Genetic Parameters**. In this study, as is showed in table 1, in four experimental cases, two configuration parameters of GA were varied by using the trial and error technique: the number of generations run (GR) and population size (PS). However, a major difficulty encountered when using GAs is the parameter setting (Pongcharoen et al., 2007). There exist many forms and variations of GAs and the best choice is problem dependant on the proper selection of the genetic operators. A GA can show good or weak results even when applied on the same problem. More research is needed in order to overcome the drawbacks associated with the optimum selection of GA parameters.

NET	GR	PS	SEL	ROP	MAT	MUT
1	10	60	50%S	CS	TST	RET25%S
2	10	60	50%S	CS	TST	RET25%S
3	5	30	50%S	CS	TST	RET25%S
4	10	30	50%S	CS	TST	RET25%S

Table 1. GA configuration parameters

where GR are the Generations Run, PS is the Population Size, SEL is the Selection technique, 50%S is the top 50% Surviving, ROP is the Refilling of the Population Technique, CS is Cloning the Survivors technique, MAT is the Mating technique, TST is the TailSwap Technique, MUT is the Mutation technique and RET25%S is Random Exchange Technique at a rate of 25%.

6. **Training Parameters**. As can be seen from table 2, by using the trial and eror technique, in first three experiments, the minimum network training passes (MNTP) and the cutoff for network training passes (CNTP) were selected between 200-250 respectively for each network trained. In a similar way, in the last experiment, the same parameters were selected between 50-55. The Limit on Hidden Neurons (LHN) was varied as well, as is showed in table 2. In experiments one and three, each evolved network was trained five and three times respectively and then the results were averaged. In all experiments, the weight initialization was between 0-0.3, and the learning rate and momentum between 0.1-0.4 and 0.1-0.3 respectively. The performance of the EANN is highly dependant on the proper selection of these parameters, however, from the literature reviewed, there is not a methodological criteria for this selection, and more research is needed to overcome this drawback.

NET	MNTP	CNTP	LHN
1	200	250	8
2	200	250	32
3	200	250	256
4	50	55	256

Table 2. EANN training configuration parameters

where MNTP are the minimum network training passes for each network, CNTP are the cutoff for network training passes for each network, and LHN is the the Limit on Hidden Neurons in the design process of the EANN methodology.

After optimum net topologies were determined, was observed that NGO presents several inconveniences when applied in the neutron spectroscopy domain. First problem is due the way the compendium of neutron spectra of IAEA compilation was realized. These spectra are normalized to one and are used to train the different EANNs. This does not represent a problem in the training and testing stages of EANN design, however, after training was done and the neural net is used to solve real experimental problems, this becomes a drawback because the spectra unfolded and doses calculated resulting are normalized to one and this has not physical meaning. Because the anterior, in the solution of real experimental problems a procedure to un normalize the spectra unfolded was needed. Another drawback when NGO is used in the neutron spectrometry domain is that has not the capability to graph in a proper way the spectra and doses calculated. The anterior suggested the necessity to design a customized computer tool to overcome the drawbacks mentioned.

Because the novelty of EANN methodology applied in the neutron spectrometry research and the lack of tools for the analysis of the spectrometric and dosimetric results obtained with NGO, a customized computer code denominated "Neutron Spectrometry and Dosimetry based on Evolutionary Artificial Neural Networks" (NSDEann), showed in figure 11, was designed in a graphical user interface, under the LabVIEW programming environment, which is easy, intuitive, friendly and quick in their use. This code is oriented to be used by the end user in laboratory, experimental and/or research environments, and its aim is to overcome the drawbacks associated with NGO when applied to solve the neutron spectra unfolding problem.

The principle of operation of NSDEann is the following: after executing the main program, a window, as showed in figure 11, will open. To unfold the spectra by using the NSDEann code, the user should execute the following steps:

1. Select a file text with the rate counts measured with the BSS system through the ”BSS counts rate” tool. Previously, the end user should be created a file with extension *.txt, with the rate counts arranged in column form. In the case of several measurements, the text file could contain several rate counts in columns tabulation separated. The tool “Select BSS from file” can be used to select one spectra from the file created just adjusting the number of column selection.
2. Store the normalized rate counts in a file text with extension *.txt by means of the button ”Save normalized counts“. In this step, the program calculates a normalized value of rate counts which has two purposes, normalize the BBS measurements to be used with NGO and to un normalize the spectra and doses calculated.
3. Open the optimum EANN trained with NGO and make a prediction with this configuration, using the normalized rate counts calculated in step two.
4. Store the neutron spectra unfolded and doses calculated by NGO, which are normalized to one, saving this information in a similar way than step one.
5. Open the file with the spectra and doses calculated with NGO by means of the tool ”Evolved Spectra&Doses“. This tool works similar to described in step one.
6. Click the button ”Plot Spectra&Doses“ to see the spectrometric and dosimetric information un normalized in the graph and numeric form in the middle and right areas of the main window.
7. The numerical values of the spectra and doses can be stored in a file text using the button ”Save Spectra&Doses“

For each spectra, the procedure before described must be repeated.
As described previously, four EANN architectures were designed with NGO for two experimental cases: the neutron spectra of a $^{239}PuBe$ neutron source in water and on air. To compare the several spectra obtained in both experiments and to analyze the performance and generalization capability of the nets designed, the tool known as ”Neutron Spectrometry and Dosimetry ToolBox” (NSDTB), was used (Ortiz-Rodríguez et al., 2009a; 2010b). NSDTB code has the capability to analyze spectrometric and dosimetric information obtained with several neutron unfolding techniques as for example: iterative procedures and the approaches based on ANN or EANN, as is the case of the present work.

3. Results

In this work, the EANN technology was analyzed and applied in the neutron spectrometry research area. The spectrometric and dosimetric features of a $^{239}PuBe$ neutron source measured under two different experimental conditions were unfolded by using the NSDEann unfolding code. The results obtained in this work reveals that the hybrid technology of EA-ANN applied in the neutron spectrometry and dosimetry problems, present some drawbacks in the optimum selection of the GA and ANN training parameters. When these parameters were varied before GA execution, in the four experimental cases considered, different network architectures were obtained, as is showed in table 3. As can be seen from this table, a total of 1650 evolutionary network architectures were designed, trained and tested by means of EA, under four different experimental conditions. The accuracy on training and testing, is similar in the four cases considered, being around 97%.
where NET-TOP is the optimum evolved net topology, ATrS is the Accuracy on Training Set, MATtS is the Maximum Accuracy on Test Set, NT are the total number of Networks Trained,

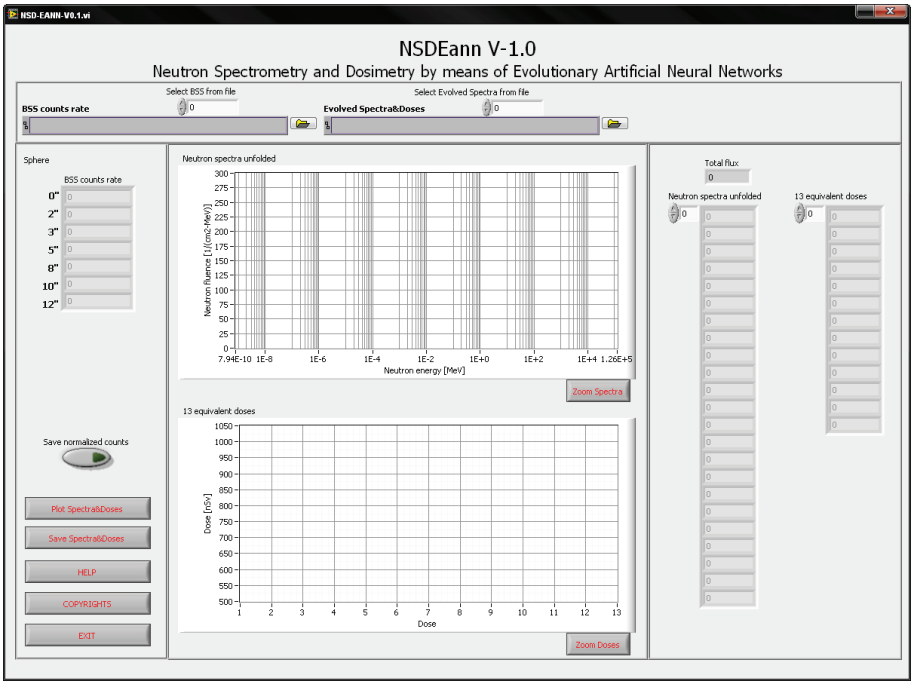

Fig. 11. Main window of NSDEann unfolding code

NET	NET-TOP	ATrS	MATtS	NT	TIME	FOG
1	7-8-44	98.67%	91.89 %	600	06:04:49	9
2	7-19(1)-44	98.00%	94.59%	600	01:54:30	10
3	7-183(14)-123(39)44	100.00%	97.30%	150	04:42:35	4
4	7-66-44	97.33%	94.59%	300	01:16:34	10

Table 3. Evolved network topologies and performance

TIME is the time used by GA to train all net topologies and FOG is the best network Found on Generation.

The accuracy on training and testing reveals that the four network architectures learned well the training and testing data sets, which lets infer that the performance and generalization capability of all networks architectures is good, which was confirmed by using the NSDTB code, as is showed in figure 12. In this figure can be seen the four neutron spectra unfolded with the four network architectures showed in table 3, for a $^{239}PuBe$ neutron source measured at 1m in water. Here, spectra-1 and doses-1 corresponds to the spectra unfolded with the architecture of the NET 1, spectra-2 and doses-2 corresponds to the spectra unfolded with the architecture of the NET 2, etc. From this figure can be seen that although the four EANN with different architectures were used to unfold the neutron spectra of the source before mentioned, by using in all cases the same rate counts measured with the BSS system, the results are similar. By comparing the spectra calculated with the codes NSDEann and BUNKIUT, as is showed in figure 13, can be seen that are alike. The spectra have a peak in the thermal region and between 0.1 and 10 MeV (Vega-Carrillo et al., 2010).

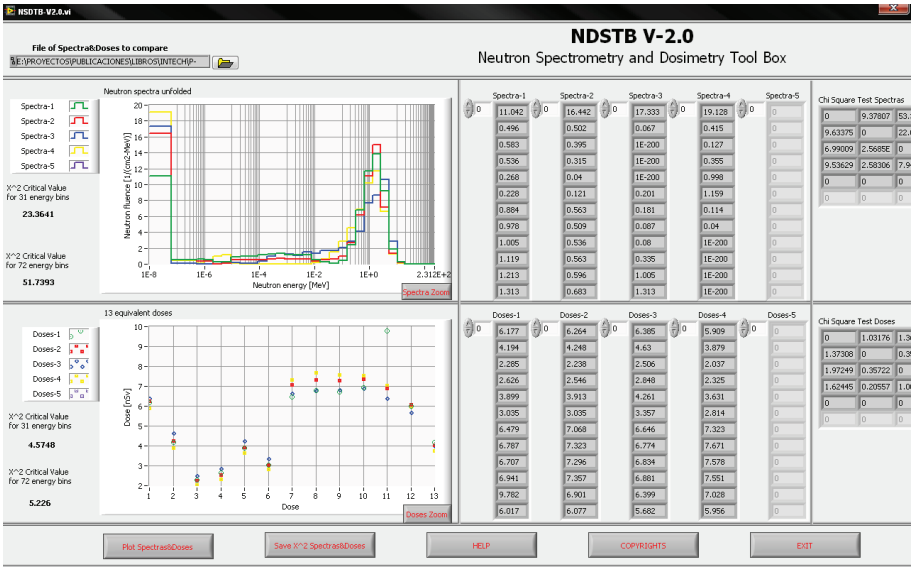

Fig. 12. Neutron spectra unfolded with the NSDEann code for a $^{239}PuBe$ at $1m$ in water

The NSDTB code realizes a chi square statistical test among the spectra analyzed, as is showed at right side of the main window. As can be seen from table 4, Spectra-1 and Doses-1 are statistically compared whit the rest of spectra and doses of row 1, Spectra-2 and Doses-2 are compared with spectra and doses of row 2, etc. This comparison reveals that there are statistical differences between most of the spectra unfolded with the different network topologies, mainly spectra 3 and 4. In the case of the equivalent doses, the test reveals that statistically there are not differences between doses calculated with the four network architectures.

Chi square test compares two groups of data, as in the case of the energy bins that conform the neutron spectra calculated with the several EANN architectures, however, when the energy bins presents high variations in some of the values of the data being compared, as can be appreciated in figures 12 to 15, this test tends to fail. From the analysis realized in the present work, has been observed that the chi square test presents some drawbacks, and a more accurate statistical test is needed.

NET	S1	S2	S3	S4	D1	D2	D3	D4
1	0	9.378	53.355	58.761	0	1.032	1.366	1.346
2	9.634	0	22.043	45.063	1.373	0	0.359	0.145
3	7E+199	3E+199	0	1E+200	1.972	0.357	0	0.945
4	1E+201	3E+200	8E+200	0	1.624	0.206	1.007	0

Table 4. Chi square test of NSDTB code for a $^{239}PuBe$ at $1m$ in water

Figure 13 shows a comparison of the four neutron spectra obtained with the NSDEann code with respect to the neutron spectra calculated using the BUNKIUT code. Here, spectra-1 corresponds to the spectra calculated with the BUNKIUT code and the rest as was explained for figure 12.

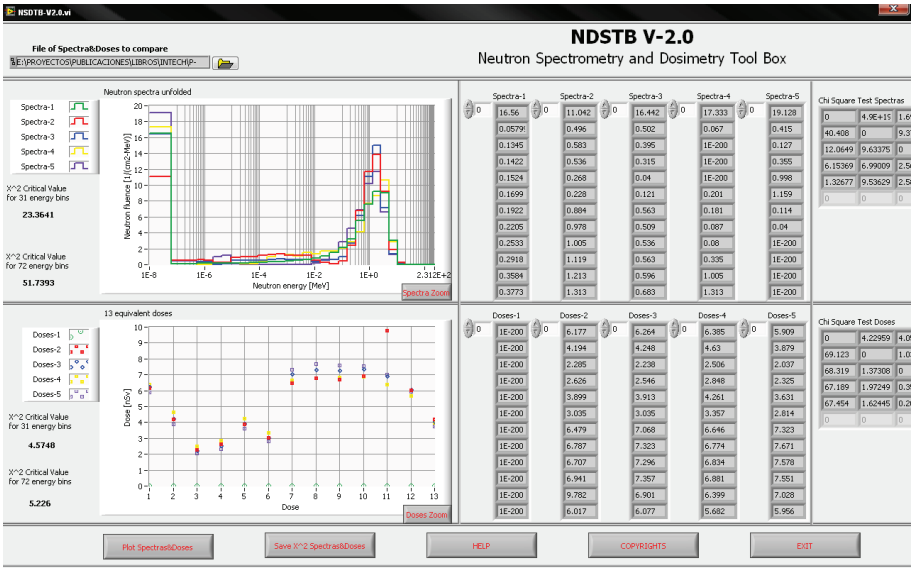

Fig. 13. BUNKIUT-NSDEann comparison of the spectra of a $^{239}PuBe$ at $1m$ in water

In the case of figure 13, the chi square test fails because the high variations of some of the values of the energy bins that compose the several spectra compared, despite the similarity between the shape and features of the spectra, as was explained for figure 12.
Figure 14 shows the neutron spectra unfolded for a $^{239}PuBe$ measured at $1m$ on air. As can be seen from this figure, the spectra are very similar in shape and features, however, the chi square test fails when compare the different spectra, as is showed in table 5.
Table 5 shows the statistical test realized by the NSDTB code. As can be seen, despite the similarities of the spectra, the test fails. A more accurate statistical test for comparing the neutron spectra obtained with several unfolding codes is needed.

NET	S1	S2	S3	S4	D1	D2	D3	D4
1	0	2E+200	5E+201	6E+201	0	0.261	0.153	1.74
2	14.892	0	108.52	98. 579	0.27	0	0.413	0.711
3	7E+198	1E+200	0	5E+200	0.162	0.419	0	1.788
4	5E+201	2E+201	4E+201	0	1.888	0.741	1.893	0

Table 5. Chi square test of NSDTB code for a $^{239}PuBe$ at $1m$ in air

Figure 15 shows a comparison of the four neutron spectra obtained with the NSDEann code with respect to the neutron spectra calculated using the BUNKIUT code for a $^{239}PuBe$ measured at $1m$ on air. Here, spectra-1 corresponds to the spectra calculated with the BUNKIUT code. As can be seen from this figure, the spectra are very similar in shape and features, however, the chi square test fails when compare the different spectra.

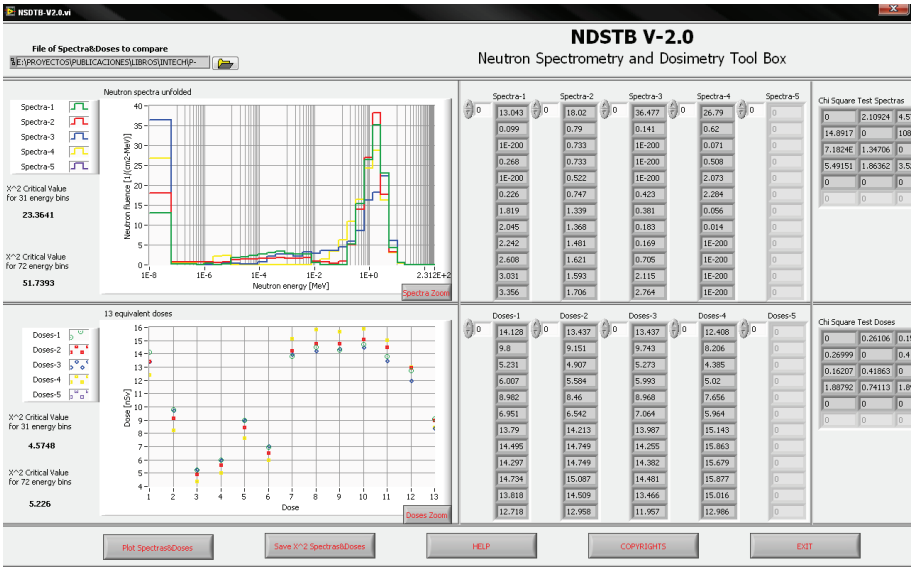

Fig. 14. Nueutron spectra unfolded with the NSDEann code for a $^{239}PuBe$ at $1m$ in air

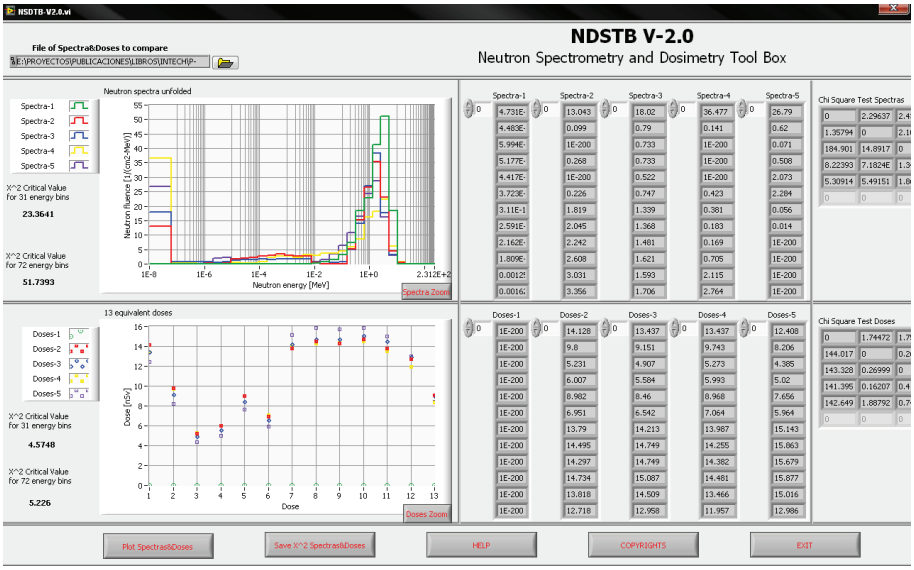

Fig. 15. BUNKIUT-NSDEann comparison of the spectra of a $^{239}PuBe$ at $1m$ in air

4. Conclusions

In this work, the intersection of GA and ANN in the neutron spectrometry research by means of ANN technology was analyzed. The most common and widely used methods of optimization in ANN are classical gradient methods, such as BP. These methods are characterized by a very fast reaching the optimum, yet, they have a considerable disadvantage "converging the algorithm to the closest optimum". So never is known whether the result of the optimization is a local or a global optimum. On the other hand, the genetic search capability in ANN design is much more effective than random searching approaches, as the genetic process of recombining features vastly improves the speed of identifying highly fit networks. It also has a potential advantage over just using personal experience in building neural networks, as new and potentially better solutions may be found through this process than might be found using the nearly unavoidable assumptions made by the user.

Contrary to the classical methods, the Evolutionary Algorithms cope with the global optimum searching quite well, but they are not as precise as the classical methods. Another disadvantage of the EA is a big (sometimes very big) amount of compilations. A major difficulty encountered when using GAs is the parameter setting. There exist many forms and variations of GAs and the best choice is problem dependant. Accordingly, a GA can show a good or weak result even when applied on the same problem. Like other learning paradigms, the performance of GA is dependent on the parameter choice, on the problem representation and on the fitness landscape.

The new ideas and concepts of EA and ANN bring new life into artificial intelligence research applied in the nuclear ciences. However, new problems arise of combining EA and ANN, such as the proper determination of the EA parameters or the need of computer tools to apply this new technology.

Because the novelty of EANN technology in neutron spectrometry and the lack of tools for the analysis, an unfolding code based on EANN technology, called "Neutron Spectrometry and Dosimetry based on Evolutionary Artificial Neural Networks" (NSDEann), was developed in a customized front end user capable to unfold neutron spectra and to simultaneously calculate 13 equivalent doses, by using only as input data the count rates coming of a BBS system, in just a few seconds if compared with the time spent with the classical techniques, not being needed a priori information about the spectra being calculated.

One disadvantage of the NSDEann code is the dependency with NGO software. Because the anterior, a customized and independent code should be of help in the neutron spectrometry field where EANN technology is applied. At present, work is being done in this sense.

When the NSDTB code was used to compare the spectra unfolded with four evolutionary network architectures obtained with the NGO software and analyzed with the NSDEann unfolding code, was observed that the chi square statistical test failed. This is due of high variations in some of the energy bins that compose the neutron spectra. From the analysis realized in the present work, has been observed that the chi square test presents some drawbacks, and a more accurate statistical test for comparing the neutron spectra obtained with several unfolding codes is needed.

5. References

Affenzeller, M., Winkler, S., Wagner, S. & Beham, A. (2009). *Genetic algorithms and genetic programming, modern concepts and practical applications*, CRC Press.

Apolloni, B., Bassis, S. & Marinaro, M. (2009). *New directions in neural networks*, IOS Press.

Beale, M. H., Hagan, M. T. & Demuth, H. B. (1992). *Neural networks toolbox, user's guide*, Mathworks. www.mathworks.com/help/pdf_doc/nnet/nnet.pdf.

Bedogni, R., Domingo, C., Esposito, A. & Fernández, F. (2007). Fruit: an operational tool for multisphere neutron spectrometry in workplaces, *Nuclear Instruments and Methods in Physics Research A* 580: 1301–1309.

BiocompSystems (2010). Biocomp systems.

Brooks, F. D. & Klein, H. (2002). Neutron spectrometry, historical review and present status, *Nuclear Instruments and Methods in Physics Research A* 476: 1–11.

Chennakesava, R. (2008). *Fuzzy logic and neural networks, basic concepts and applications*, New Age International Publishers.

Coppin, B. (2004). *Artificial intelligence illuminated*, Jones and Barttlet Publishers.

Dreyfus, G. (2005). *Neural networks, methodology and applications*, Springer.

El Messaoudi, M., Chouak, A., Lferde, M. & Cherkaoui, R. (2004). Performance of three different unfolding procedures connected to Bonner sphere data, *Radiation Protection Dosimetry* 108(3): 247–253.

Fausett, L. (1993). *Fundamentals of neural networks, architectures, algorithms and applications*, Prentice Hall.

Floreano, D. & Mattiussi, C. (2008). *Bio-inspired artificial intelligence, theories, methods and technologies*, MIT Press.

Freeman, D. W., Edwards, D. R. & Bolon, A. E. (1999). Genetic algorithms, a new technique for solving a neutron spectrum unfolding problem, *Nuclear Instruments and Methods in Physics Research A* 425(3): 549–576.

Galushkin, A. (2007). *Neural networks theory*, Springer.

Gen, M., Green, ., Katai, O., McKay, B., Namatame, A., Sarker, R. A. & Zhang, B. T. (2009). *Intelligent and evolutionary systems*, Springer-Verlag.

Goldberg, D. (1989). *Genetic Algorithms in search, optimization, and machine learning*, Addison Wesley.

Graupe, D. (2007). *Principles of artificial neural networks*, World Scientific.

Haupt, R. L. & Haupt, S. E. (2004). *Practical genetic algorithms*, Wiley.

Haykin, S. (1999). *Neural networks: a comprenssive foundation*, Prentice Hall.

IAEA (2001). Compendium of neutron spectra and detector responses for radiation protection purposes, *Technical Report 403*.

Iñiguez, M. P. & Vega-Carrillo, H. R. (2002). Catalogue to select the initial guess spectrum during unfolding, *Nuclear Instruments and Methods in Physics Research A* 476(1): 270–273.

Jain, A. K., Mao, J. & Mohiuddin, K. M. (1996). Artificial neural networks: a tutorial, *IEEE: Computer* 29(3): 31–44.

Kasabov, N. K. (1998). *Foundations of neural networks, fuzzy systems, and knowledge engineering*, MIT Press.

Kishan, M., Chilukuri, K. & Sanjay, R. (2000). *Elements of artificial neural networks*, The MIT Press.

Lacoste, V., Gressier, V., Pochat, J. L., Fernández, F., Bakali, M. & Bouassoule, T. (2004). Characterization of bonner sphere systems at monoenergetic and thermal neutron fields, *Radiation Protection Dosimetry* 110(1-4): 529–532.

Lacoste, V., Reginatto, M., Asselineau, B. & Muller, H. (2007). Bonner sphere neutron spectrometry at nuclear workplaces in the framework of the evidos project, *Radiation Protection Dosimetry* 125(1-4): 304–308.

Lakhmi, J. & Fanelli, A. M. (2000). *Recent advances in artificial neural networks design and applications*, CRC Press.

Leardi, R. (2003). *Nature-inspired methods in chemometrics: genetic algorithms and artificial neural networks*, Elsevier.

Luger, F. G. (2005). *Artificial Intelligence, structures and strategies for complex problem solving*, Addison-Wesley.

Matzke, M. (2003). Unfolding procedures, *Radiation Protection Dosimetry* 107(1-3): 155–174.

Mazrou, H., Sidahmed, T., Idiri, Z., Lounis-Mokrani, Z., , Bedek, Z. & Allab, M. (2008). Characterization of the CRNA Bonner sphere spectrometer based on 6lii scintillator exposed to an 241ambe neutron source, *Radiation Measurements* 43: 1095–1099.

McDonald, J. C., Siebert, B. R. L. & Alberts, W. G. (2002). Neutron spectrometry for radiation protection purposes, *Nuclear Instruments and Methods in Physics Research A* 476(1-2): 347–352.

Melin, P. & Castillo, O. (2005). *Hybrid intelligent system for pattern recognition using soft computing, an evolutionary approach for neural networks and fuzzy systems*, Springer.

Miller, S. (1993). *AFITBUNKI: a modified iterative code to unfold neutron spectra from Bonner sphere detector data*, Master's thesis.

Mitchell, M. (1998). *An introduction to genetic algorithms*, MIT Press.

Mukherjee, B. (2004). Andi-03: a genetic algorithm tool for the analysis of activation detector data to unfold high-energy neutron spectra, *Radiation Protection Dosimetry* 110(1-4): 249–254.

Munakata, T. (2008). *Fundamentals of the new artificial intelligence, neural, evolutionary, fuzzy and more*, Springer.

Ortiz-Rodríguez, J. M., Martínez-Blanco, M. R., Gallego, E. & Vega-Carrillo, H. R. (2008). Artificial neural networks modeling evolved genetically, a new approach applied in neutron spectrometry and dosimetry research areas, *Proceedings of the Electronics, Robotics and Automotive Mechanics Conference (CERMA'08)*, IEEE Computer Society, Cuernavaca, Mor., México, pp. 387–392.

Ortiz-Rodríguez, J. M., Martínez-Blanco, M. R., Gallego, E. & Vega-Carrillo, H. R. (2009a). A computational tool design for evolutionary artificial neural networks in neutron spectrometry and dosimetry, *Proceedings of the Electronics, Robotics and Automotive Mechanics Conference (CERMA'09)*, IEEE Computer Society, Cuernavaca, Mor., México, pp. 113–118.

Ortiz-Rodríguez, J. M., Martínez-Blanco, M. R., Gallego, E. & Vega-Carrillo, H. R. (2009b). Evolutive artificial neural networks for neutron spectra unfolding, *American Nuclear Society Transactions* 101: 647–648.

Ortiz-Rodríguez, J. M., Martínez-Blanco, M. R., Gallego, E. & Vega-Carrillo, H. R. (2010a). Neutron spectrometry and dosimetry based on a new approach called genetic artificial neural networks, *12th Congress of the International Radiation Protection Association (IRPA12)*, IAEA Procceding Series STI/PUB/1460 1-9, International Atomic Energy Agency, Vienna.

Ortiz-Rodríguez, J. M., Martínez-Blanco, M. R., Gallego, E. & Vega-Carrillo, H. R. (2010b). A neutron spectrometry and dosimetry computer tool based on ann, *12th Congress of the International Radiation Protection Association (IRPA12)*, IAEA Procceding Series STI/PUB/1460 1-9, International Atomic Energy Agency, Vienna.

Ortiz-Rodríguez, J. M., Martínez-Blanco, M. R. & Vega-Carrillo, H. R. (2006). Robust design of artificial neural networks applying the Taguchi methodology and DoE, *Proceedings*

of the Electronics, Robotics and Automotive Mechanics Conference (CERMA'06), IEEE Computer Society, Cuernavaca, Mor., México, pp. 131–136.

Packianather, M. S. & Drake, P. R. (2004). Modeling neural network performance through response surface methodology for classifying wood veneer defects, *Proceedings of the Institution of Mechanical Engineers, Part B* 218(4): 459–466.

Packianather, M. S., Drake, P. R. & H., R. (2000). Optimizing the parameters of multilayered feedforward neural networks through Taguchi design of experiments, *Quality and Reliability Engineering International* 16: 461–473.

Periaux, J. & Winter, G. (1995). *Genetic algorithms in engineering and computer science*, John Wiley & Sons.

Peterson, G. E., St. Clair, D. C., Aylward, S. R. & E., B. W. (1995). Using Taguchi's method of experimental design to control errors in layered perceptrons, *IEEE Transactions on Neural Networks* 6(4): 949–961.

Pongcharoen, P., Chainate, W. & Thapatsuwan, P. (2007). Exploration of genetic parameters and operators through traveling salesman problem, *Science Asia* 33: 215–222.

Rasskin-Gutman, D. (2009). *Chess metaphors, artificial intelligence and the human brain*, The MIT Press.

Reginatto, M. & Zimbal, A. (2008). Bayesian and maximum entropy methods for fusion diagnostic measurements with compact neutron spectrometers, *Review of Scientific Instruments* 79(2): 398–403.

Roberts, N. J. (2007). Investigation of combined unfolding of neutron spectra using the UMG unfolding codes, *Radiation Protection Dosimetry* 126(1-4): 398–403.

Rothlauf, F. (2006). *Representations for Genetic and Evolutionary Algorithms*, Springer.

Taylor, J. G. (1993). *Mathematical approaches to neural networks*, North-Holland Mathematical library.

Thomas, D. J. (2004). Neutron spectrometry for radiation protection, *Radiation Protection Dosimetry* 110(1-4): 141–149.

Vega-Carrillo, H. R., Hernández-Dávila, V. M., Manzanares-Acuña, E., Gallego, E., Lorente, A. & Iñiguez, M. P. (2007b). Artificial neural networks technology for neutron spectrometry and dosimetry, *Radiation Protection Dosimetry* 126(1-4): 408–412.

Vega-Carrillo, H. R., Hernández-Dávila, V. M., Manzanares-Acuña, E., Mercado Sánchez, G. A., Gallego, E., Lorente, A., Perales-Muñoz, W. A. & Robles-Rodríguez, J. A. (2005). Artificial neural networks in neutron dosimetry, *Radiation Protection Dosimetry* 118(3): 251–259.

Vega-Carrillo, H. R., Hernández-Dávila, V. M., Manzanares-Acuña, E., Mercado-Sánchez, G. A., Iñiguez de la Torre, M. P., Barquero, R., Preciado-Flores, S., Méndez-Villafañe, R., Arteaga-Arteaga, T. & Ortiz-Rodríguez, J. M. (2006). Neutron spectrometry using artificial neural networks, *Radiation Measurements* 41: 425–431.

Vega-Carrillo, H. R., Manzanares-Acuña, E., Ortiz-Rodríguez, J. M. & Arteaga-Arteaga, T. (2007a). Neutron spectra re-binning and dose calculation using monte carlo methods, *Revista Mexicana de Física* 53: 1–7.

Vega-Carrillo, H. R., Martínez-Blanco, M. R., Hernández-Dávila, V. M. & Ortiz-Rodríguez, J. M. (2009a). Spectra and dose with ANN of 252Cf, 241AmBe, and 239PuBe, *Journal of Radioanalytical and Nuclear Chemistry* 281: 615–618.

Vega-Carrillo, H. R., Ortiz-Rodríguez, J. M., Hernández-Dávila, V. M., Martínez-Blanco, M. R., Hernández-Almaraz, B., Ortiz-Hernández, A. & Mercado, G. A. (2009b). Different spectra with the same neutron source, *Revista Mexicana de Fśica S* 56(1): 35–39.

Vega-Carrillo, H. R., Ortiz-Rodríguez, J. M., Martínez-Blanco, M. R. & Hernández-Dávila, V. M. (2010). Ann in spectroscopy and neutron dosimetry, *American Institute of Physics Proccedings* 1310: 12–17.

Vylet, V. (2002). Response matrix of an extended Bonner sphere system, *Nuclear Instruments and Methods in Physics Research A* 476: 26–30.

Whiteson, S. & Stone, P. (2006). Evolutionary function approximation for reinforcement learning, *The Journal of Machine Learning Research* 7: 877–917.

Wielunski, M., Wahl, W., EL-Faramawy, N., Ruhm, W., Luszik-Bhadra, M. & Roos, H. (2008a). Development of a Brazilina gamma-neutron dosimeter, *Nuclear Instruments and Methods in Physics Research B* 266(12-13): 3174–3177.

Wielunski, M., Wahl, W., EL-Faramawy, N., Ruhm, W., Luszik-Bhadra, M. & Roos, H. (2008b). Intercomparison exercise Whith MeV neutrons using various electronic personal dosimeters, *Radiation Measurements* 43(2-6): 1063–1067.

Yao, X. (1993). Evolutionary artificial neural networks, *International Journal of Neural Systems* 4(3): 203–222.

Zupan, J. (1994). Introduction to artificial neural network methods: what they are and how to use them, *Acta Chimica Slovenica* 41(3): 327–352.

View publication stats