

Dolores Barrios · Alberto Carrascal  
Daniel Manrique · Juan Ríos

## Cooperative binary-real coded genetic algorithms for generating and adapting artificial neural networks

Received: ■ / Accepted: ■  
© Springer-Verlag London Limited 2003

**Abstract** This paper describes a genetic system for designing and training feed-forward artificial neural networks to solve any problem presented as a set of training patterns. This system, called GANN, employs two interconnected genetic algorithms that work parallelly to design and train the better neural network that solves the problem. Designing neural architectures is performed by a genetic algorithm that uses a new indirect binary codification of the neural connections based on an algebraic structure defined in the set of all possible architectures that could solve the problem. A crossover operation, known as Hamming crossover, has been designed to obtain better performance when working with this type of codification. Training neural networks is also accomplished by genetic algorithms but, this time, real number codification is employed. To do so, morphological crossover operation has been developed inspired on the mathematical morphology theory. Experimental results are reported from the application of GANN to the breast cancer diagnosis within a complete computer-aided diagnosis system.

**Keywords** Artificial neural networks · Breast cancer diagnosis · Genetic algorithms · Real number codification · Genetic neural design · Genetic neural train

### 1 Introduction

The main research areas in artificial intelligence are, at the present, related to the design of auto-adaptive systems, which are able to transform themselves to solve different kind of problems [1]. Artificial neural networks are applicable to solve a great variety of real-world tasks due to their properties of learning by examples, generalising to unseen data and noise filtering [2, 3]. However, a neural architecture that performs a task accurately is useless to solve a different problem. It is needed to design a new architecture from the starting point: structuring the network connectivity, deciding the number of hidden units and setting the terms in the weight adjustment algorithm, which is not a trivial task even for the most expert practitioners due to the size and complexity of the search space available even for the best understood network models [4]. In these conditions, building a neural network based auto-adaptive intelligent system to solve any problem presented as a set of training patterns becomes unreachable.

This is the reason why several research works have focused on designing new approaches based on different search and optimization techniques to choose the best neural architecture to solve a given problem and to speed up the training process. Some of these studies are the so called incremental algorithms [5], which start from a predefined neural architecture and, then, dynamically add and remove neural connections during the training process. These algorithms present low performance and premature convergence problems depending on the initially chosen architecture, not guaranteeing a good solution [6]. Other approaches are related to how model selection in neural networks can be guided by statistical procedures such as hypothesis tests, information criteria and cross validation [7], while others are based on employing linear programming [8]. The main disadvantage of such methods is their high computational cost.

Other more promising studies are derived from the identification of synergies existing between genetic

D. Barrios · A. Carrascal · D. Manrique · J. Ríos  
Facultad de Informática, Dpto. Inteligencia Artificial

J. Ríos (✉)  
Facultad de Informatica, Campus de Montegancedo s/n  
28660 Boadilla del Monte, Madrid, Spain  
Tel.: + 34 91 336.7417  
Fax: + 34 91 352.4819  
E-mail: jríos@fi.upm.es

algorithms and artificial neural networks that allow to combine them in various ways. Thus, there are works related to the genetic adaptation of the internal structure of the network [9]. Genetic algorithms have been used to replace completely the network learning method [10], while other researchers make this replacement in a partial way [11], applying in first place genetic algorithms to accomplish the global search until a point near the solution is reached and, then, local search with classical gradient descent methods is executed to get the optimum solution. This latter approach has the inconvenience that it is unknown when is the best tune to change from global to local search.

For any evolutive optimization approach chosen, the way the neural networks that make up the search space encoded is the crucial step to accomplish the task of designing them automatically [12]. In the same way, best results in training artificial neural networks with genetic algorithms are obtained when real numbers codification, which represent the weights of the neural connections, is employed. This fact corroborates Michalewicz's assertion: if a problem is real valued in nature the a real number genetic algorithm is faster and more precise that a binary encoded genetic algorithm [13].

There have been several approaches to obtain an efficient codification of ANNs. The first of these is the direct binary encoding of network configuration [14] where each bit determines the presence or absence of a single connection. This approach presents two major problems: first, convergence performance is degraded as the size of the network increases because the search space is much larger. Second, direct encoding methods can not prevent illegal points in the search space. Opposite to the direct encoding methods, there are other approaches to encode ANNs where there is not a direct correspondence between each bit of the string and each connection of the neural architecture. These are called indirect encoding methods, being the graph generation system one of the methods that best results have reported [12, 15]. This approach is based on the binary codification of grammars that describe the architecture of the network and prevent the codification of illegal neural architectures. The problem here is that one bit variation in a string results in a totally different network. This fact degrades the convergence process of the genetic algorithm that uses this codification.

There exist different techniques to train artificial neural networks with real-coded genetic algorithms: Radcliffe's flat crossover [16] chooses parameters for an offspring by uniformly picking parameter values between (inclusively) the two parents parameter values. Later, to avoid the premature convergence problems existing in this operator, BLX- $\alpha$  was proposed [17], which uniformly picks values that lie between two points that contain the two parents and it may extend equally on either side of the interval defined by the parents. This new method, however, is very slow when approximating to the optimum because the extension of the interval defined by the parents is determined by a static user

specified parameter  $\alpha$  fixed at the running start. Other important crossover technique for real-coded genetic algorithms is the UNDX [18] that can optimize functions by generating the offspring using the normal distribution defined by three parents. The problem here is the high computational cost required to calculate the normal distribution.

This paper presents a new system for design a train automatically artificial neural networks by genetic algorithms known as GANN (Genetic Algorithm Neural Networks). This systems employs a different approach for the binary encoding of neural architectures in the design step. This codification method, called basic-architectures codification method, is based on the definition of an Abelian semi-group with neutral element in the set neural architectures. It avoids illegal networks and needs very short encoding length, being optimum when encoding neural networks with one output. The proposed codification encodes any kind of generalised feed-forward neural networks with one hidden-layer. It also has the important feature that one bit variation in the string that represents a network results in a very similar neural architecture, which, as it will be seen in the results section, improves the performance of the genetic algorithm. A specialized binary crossover operator to work with the proposed codification method has been also designed: the Hamming crossover (HX). This operator has better performance in searching for neural architectures than the other crossover operators.

GANN system trains, parallelly, the artificial neural networks obtained in the previous design step, employing real-coded genetic algorithms. To do so, a new crossover operator has been designed and called morphological crossover (MX). It has the important feature that extends adaptively (depending on the progenitors values) the interval schemata defined by the parents, from which the offspring is obtained. MX gives a new interpretation to the morphological gradient operation, very used in digitalized image segmentation [19, 20], to get an on-line genetic diversity measure. This measure increases the local search speed of the genetic algorithm in the training process, the main inconvenient reported by the researchers that prefer to combine genetic algorithms and gradient descent methods, while avoids local optima.

---

## 2 General structure of GANN system

The output of GANN is the minimum neural network that gets the mean square error less than a value previously established for a set of training patterns that describes the problem to be solved. This set of training patterns is the input to the system. The structure of GANN consists of two modules that work parallelly (figure 1). The *architectonic design module* employs a genetic algorithm with the basic-architectures codification method to search for the optimum neural architecture that solves the given problem. The *training module* receives a binary-coded neural architecture from

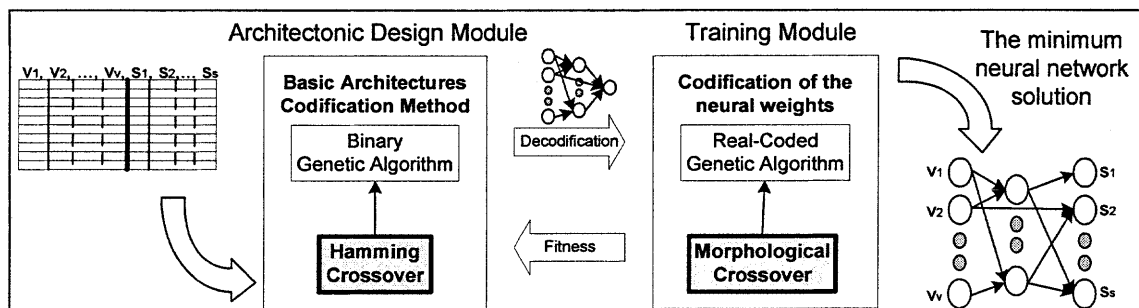


Fig. 1 Overview of GANN structure

the population of the architectonic design module to be evaluated after a decodification process. The training module consists on a real-coded genetic algorithm that employs the morphological crossover to find the values of the weights for the neural connections that minimize the mean square error for the set of training patterns. The mean square error (MSE) obtained after the training process is then used to calculate the fitness of the binary string that represents this neural network in the population handled by the architectonic design module.

For explanation purposes, next two sections describe first, the training module with the morphological crossover and, then, the architectonic design module going deeper into the new codification method of neural architectures and the Hamming crossover.

### 3 Genetic training of artificial neural networks module

The training module employs a general-purpose method of training artificial neural networks with real-coded genetic algorithms. This module trains the neural architectures obtained by the architectonic design module, whose search space is formed by feed-forward neural networks with one hidden layer due to these are the type of neural networks that are permitted by the basic-architectures codification method. This is the reason why the training module only manages these type of networks when working embedded into GANN, although it is a more general module that even allows the development of new neural architectures without the need of designing new algorithms to adjust the network weights and the characteristics implemented in the architecture.

The training module consists on a real-coded genetic algorithm formed by a population of individuals each of which encodes, with real numbers, the set of weights of the connections and biases of the network to be trained. In the notation used,  $W_{ij}(k)$  is the weight of the connection from the  $j$ th neuron of layer  $k$  to the  $i$ th neuron of layer  $k + 1$ , with  $j = 1, \dots, I$  for the input layer,  $j = 1, \dots, H$  for the hidden layer,  $i = 1, \dots, O$  for the output layer and  $k = \{0, 1\}$  representing respectively the input and hidden layer.  $W_{ij}^*$  represents a direct connection from the  $j$ th neuron in the input layer to the  $i$ th neuron of the output layer. Figure 2 shows how this codification is made:

starting from the first input unit, the first genes of each individual represents the weights of the connections from this unit to all hidden units, then, the weights from the second input unit to all hidden units, and so on with all the input neurons. The following genes represent the biases of the hidden units, the weights from the hidden units to the output neurons, the weights of the direct connections from the inputs to the output and, finally, the biases of the output neurons.

The genetic algorithm that employs this codification uses the roulette-wheel method as the selection algorithm. Mutation simply consists on the random variation of one on the genes, randomly chosen, of the individual. The new individuals of the offspring obtained from the crossover operation replace the worst individuals of the population (this is a particular implementation of a steady-state genetic algorithm). The training module employs the morphological crossover as the crossover operation, which has been specially designed to work on real-coded individuals.

#### 3.1 Morphological crossover

Morphological crossover is a general purpose operator for optimization problems with real-coded genetic algo-

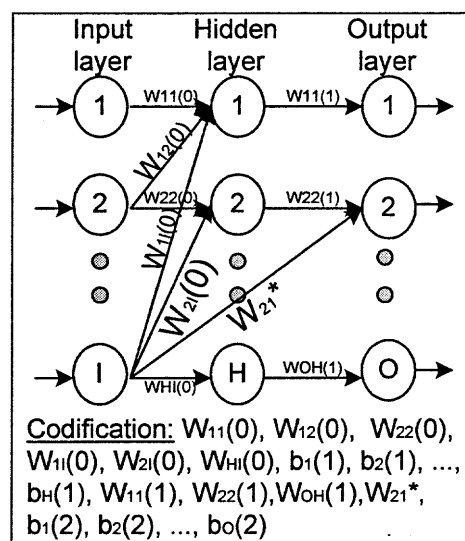


Fig. 2 Real-number codification of weights and biases of a given generalized feed-forward neural network with one hidden layer

rithms. This is just the case of training artificial neural networks where weights and biases of a network are

$$G = \begin{pmatrix} a_{10} & a_{11} & \dots & a_{1l-1} \\ a_{20} & a_{21} & \dots & a_{2l-1} \\ \dots & \dots & \dots & \dots \\ a_{n0} & a_{n1} & \dots & a_{nl-1} \end{pmatrix}$$

coded by the individuals of the population and the optimization problem consists on minimize the mean square error for the set of training patterns.

Let be  $D_{\mathfrak{R}}$  a point in the search space, defined by the string  $s = (a_0, a_1, \dots, a_{l-1})$ , where  $a_i \in \mathfrak{R}$ . This operator works with each gene in the parents independently to obtain the corresponding gene in the two descendants. Let  $s_1, \dots, s_n$  be an odd number of strings chosen from the actual population to be crossed, the  $n$  by  $l$  progenitors matrix,  $G$ , is defined as:

$$\text{where } s_i = (a_{i0}, a_{i1}, \dots, a_{i,l-1}), \quad i = 1, \dots, n.$$

The crossover operator works with each column  $f_i = (a_{1i}, a_{2i}, \dots, a_{ni})$  in matrix  $G$  obtaining genes  $o_i$  and  $o'_i$ . The result of applying the operator to matrix  $G$  is, therefore, two new descendants  $o = (o_0, o_1, \dots, o_{l-1})$  and  $o' = (o'_0, o'_1, \dots, o'_{l-1})$ . The procedure employed by this crossover to generate the new offspring strings  $o$ ,  $o' \in D_{\mathfrak{R}}$  from the parents  $s_1, \dots, s_n$  in matrix  $G$  is accomplished by three steps:

1. Obtaining a genetic diversity measure,  $g_i$ , for each gene  $a_i$

The morphological gradient operator,  $g_b(f_i): D_f \rightarrow \mathfrak{R}$ , is applied on each vector  $f_i$ ,  $i = 0, 1, \dots, l-1$ , with a structuring element  $b: D_b \rightarrow \mathfrak{R}$  defined as:  $b(x) = 0$ ,  $\forall x \in D_b$ ,  $D_b = \{-E(n/2), \dots, 0, \dots, E(n/2)\}$ , being  $E(x)$  the integer part of  $x$ .  $g_i$  is obtained as the value:

$$g_i = g_b(f_i)(E(n/2) + 1) \quad i \in \{0, 1, \dots, l-1\}$$

The morphological gradient is usually applied to image segmentation: it returns high values when sudden transitions in gray levels values are detected, and low values if the pixels covered by the structuring element are similar, so it enhances the borders of the objects in a digital image. The result given by this operator when it is applied on each vector  $f_i$  within the morphological crossover has been reinterpreted:  $g_i$  can be considered as a measure of the heterogeneity of gene  $i$  in the individuals chosen to be crossed. If value  $g_i$  is high, the population is scattered, while if it is low, that means that the values of that gene are converging. So, we have an on-line measure of the genetic diversity of the population with very low computational cost.

2. Calculating the crossover interval:

Let  $\varphi: \mathfrak{R} \rightarrow \mathfrak{R}$  be a given function. The maximum gene is defined as:

$$g_{\text{imax}} = \max(f_i) + \varphi(g_i)$$

Likewise, the minimum gene is defined as:

$$g_{\text{imin}} = \min(f_i) + \varphi(g_i)$$

$g_{\text{imax}}$  and  $g_{\text{imin}}$  determine the crossover interval  $C_i = [g_{\text{imin}}, g_{\text{imax}}]$ , from which the  $i$  th genes  $o_i$  and  $o'_i$  will be taken in the third step.

The function  $\varphi$  determines a rule, depending on the genetic diversity  $g_i$ , that allows to dynamically control the range of the crossover interval  $C_i$  to avoid falling in local minima and to get a high convergence speed. When the individuals to be crossed are diverse (which implies a high value of the gradient  $g_i$ ) the crossover interval must be narrower according to the interval defined by the values  $\max(f_i)$  and  $\min(f_i)$ , thus allowing to explore its interior searching for the optimum much faster. On the other hand, if the individuals to be crossed are very similar (gradient  $g_i$  close to zero), which means that the population is converging, then it is advisable to expand the interval  $[\min(f_i), \max(f_i)]$  to allow the exploration of new points in the domain, thus avoiding the possible convergence to a local optimum.

The figure 3 shows the shape of function  $\varphi$  that is defined in domain  $[0, 1]$ , so the population had to be normalized in the same range, and depends on four parameters  $a$ ,  $b$ ,  $c$  and  $d$ . This shape allows morphological crossover to have the following desirable features:

- This function only performs one multiplication, so it is very efficient, as it allows the generation of a new individual with only  $l$  multiplications (the length if the individuals).
- $\varphi(g_i)$  is positive when  $g_i$  is strictly greater than  $c$ , making the crossover interval narrower in this case. The greatest narrowing takes place in  $\varphi(1) = d$ .
- $\varphi(g_i)$  takes negative values if  $g_i \leq c$ , that is, the crossover interval is expanded, and the maximum expansion is produced in  $\varphi(c) = b$ . Starting from this point the function takes values each time smaller as the gradient value diminishes, until reaching value  $a$  in  $g_i = 0$ . Otherwise the crossover interval would be too much wide compared with the crossover interval  $[g_{\text{imin}}, g_{\text{imax}}]$ .

Parameters  $a$ ,  $b$ ,  $c$  and  $d$  have been obtained by a binary genetic algorithm that searches for the values that get the highest convergence speed while avoid falling in local optima in training different neural

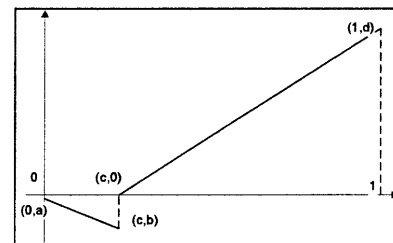


Fig. 3 Shape of function  $\varphi$  used by morphological crossover

networks to solve different benchmark tests such as the two-spiral problem [21]. Best results were obtained for  $a = -0.01$ ,  $b = -0.05$ ,  $c = 0.2$  and  $d = 0.4$ , so the analytical expression for function  $\varphi$  is the following:

$$\varphi(g_i) = \begin{cases} -(0.2 \cdot g_i) - 0.01 & \text{if } g_i \leq 0.2 \\ -(0.5 \cdot g_i) - 0.1 & \text{otherwise} \end{cases}$$

### 3. Obtaining the offspring:

For each crossover interval  $C_i = [g_{\min}, g_{\max}]$  calculated in the previous step the  $i$ th gen  $o_i$  belonging to the descendant  $o = (o_0, o_1, \dots, o_{l-1})$  is obtained by picking a value randomly from within  $C_i$ . The other  $i$ th  $o'_i$  gen that belongs to the second descendant  $o' = (o'_0, o'_1, \dots, o'_{l-1})$  is also taken from the crossover interval  $C_i$  using the following formula:

$$o'_i = g_{\max} + g_{\min} - o_i$$

This way, the two new  $i^{\text{th}}$  genes of the offspring are symmetric with respect to the central point of the crossover interval.

## 4 Architectonic design module

The architectonic design module consists on a binary genetic algorithm that searches for the generalized feed-forward neural network that best fits the set of training patterns given as the input to GANN system. The search space of this genetic algorithm is formed by the set of artificial neural networks that can solve the problem, so each individual of the population represents a neural architecture with  $I$  input neurons, one hidden layer as much with  $H$  hidden units and  $O$  outputs. The architectonic design module employs the basic architectures codification method, which will be deeper explained in the next subsection, to encode the search space. The Hamming crossover, subsection 4.2, is a new crossover operator employed with the proposed codification to exploit its features and advantages giving a high performance to the global system.

### 4.1 Basic architectures codification method

**Definition 4.1** A generalized feed-forward architecture  $r$  with  $I$  input neurons,  $H$  units in only one hidden layer and  $O$  output neurons is defined as  $r \subset (\tilde{I} \times \tilde{H}) \cup (\tilde{H} \times \tilde{O}) \cup (\tilde{I} \times \tilde{O})$  whereas  $\tilde{I} = \{i_1, i_2, \dots, i_I\}$  denotes the set of  $I$  input neurons,  $\tilde{H} = \{h_1, h_2, \dots, h_H\}$  is the set of  $H$  units in the hidden layer and  $\tilde{O} = \{o_1, o_2, \dots, o_O\}$  as the set of  $O$  output neurons. If  $(a, b) \in r$  then the neuron  $a$  is connected to the neuron  $b$ . The cartesian product of input and hidden neurons is  $\tilde{I} \times \tilde{H}$  that represents the set of all possible connections from the input layer to the hidden layer,  $\tilde{H} \times \tilde{O}$  is the set of all connections from the hidden layer to the output and  $\tilde{I} \times \tilde{O}$  represents the set of direct connections from the input to the output.

The set of all generalised feed-forward neural architectures with a maximum of  $I$  input neurons,  $H$  hidden units and  $O$  output units is denoted by  $R_{I,H,O}$ . There is a special case, the null architecture, defined as  $n = \emptyset$ , where there are not connected neurons. From the set of all existing neural architectures  $R_{I,H,O}$ , we are only interested in the subset  $V_{I,H,O} \subseteq R_{I,H,O}$  of all valid neural architectures as  $V_{I,H,O}$ , only contains points that can solve a given problem. The set  $V_{I,H,O}$  does not include illegal architectures. The basic architectures codification method only encodes points belonging to set  $V_{I,H,O}$ .

**Definition 4.2** A neural architecture  $v \in R_{I,H,O}$  with  $v \subset (\tilde{I} \times \tilde{H}) \cup (\tilde{H} \times \tilde{O}) \cup (\tilde{I} \times \tilde{O})$  is called *valid* neural architecture, and so  $v \in V_{I,H,O}$  if and only if for all  $(i_r, h_s) \in \tilde{I} \times \tilde{H} \cap v$  there exists  $o_p \in \tilde{O}$  such that  $(h_s, o_p) \in \tilde{H} \times \tilde{O} \cap v$  and, reciprocally, for all  $(h_s, o_p) \in \tilde{H} \times \tilde{O} \cap v$  there exists  $i_r \in \tilde{I}$  such that  $(i_r, h_s) \in \tilde{I} \times \tilde{H} \cap v$ . It can be deduced from this definition that the null architecture  $n \in V_{I,H,O}$ . The figure 4 shows, on the left a valid neural architecture, while there is an illegal architecture on the right because the second neuron in the hidden layer is connected to the input, but not to the output.

**Definition 4.3** Let be  $v$  and  $v' \in V_{I,H,O}$ , the superimposition operation between  $v$  and  $v'$  is defined as  $v \oplus v' = v \cup v'$ . The figure 5 shows two valid neural architectures, and the result obtained after superimposition operation has been applied.

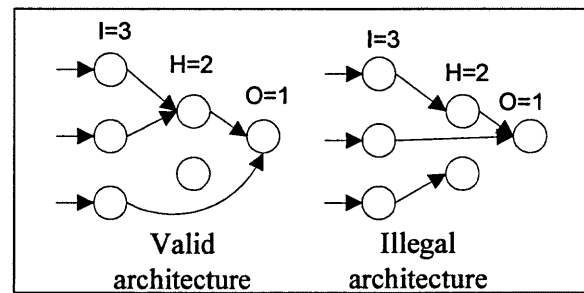


Fig. 4 An example of one valid and one illegal neural architectures

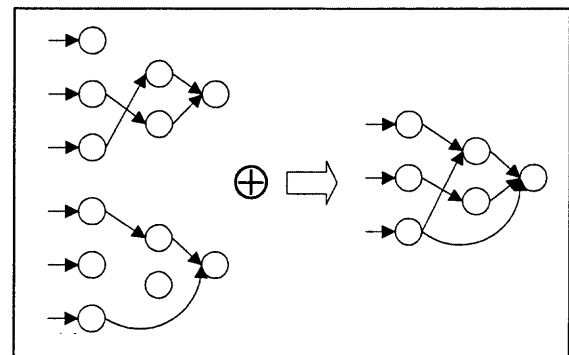


Fig 5 The superimposition operation  $\oplus$

With all of these definitions it is possible to establish that the set  $V_{I,H,O}$  of valid architectures with the operation superimposition  $\oplus$  is an Abelian semi-group with neutral element (the null architecture)  $(V_{I,H,O}, \oplus)$  which is easy to prove, as  $\oplus$  is based on the operation union between two sets.

**Definition 4.4** A valid neural architecture  $b \in V_{I,H,O}$  is called *basic neural architecture*, and so,  $b \in B_{I,H,O}$  if and only if  $\#b \leq 2$  and if  $\#b = 2$  then  $b = \{(i_r, h_s), (h_s, o_p)\}$  with  $(i_r, h_s) \in \check{I} \times \check{H}$  and  $(h_s, o_p) \in \check{H} \times \check{O}$ .  $\#b$  denotes the cardinal of the set  $b$ . The subset  $B_{I,H,O} \subseteq V_{I,H,O}$  of all basic neural architectures has important features because they allow building any valid neural architectures from these basic structures.

Since basic neural architectures are also valid neural architectures, if  $\#b = 0$  then  $b = \emptyset$ , which corresponds to the null architecture. If  $\#b = 1$  then  $b = \{(i_r, o_p)\}$ , with  $(i_r, o_p) \in (\check{I} \times \check{O})$ ;  $b$  has only one direct connection from the input to the output layer. If  $\#b = 2$ , then there is one connection from the input to one hidden unit, and other from this unit to the output neuron. The figure 6 shows three examples of basic neural architectures in the set  $B_{3,2,1}$ : the null architecture on the left, an architecture with  $\#b = 1$  and, on the right, an architecture with  $\#b = 2$ .

**Definition 4.5** Let be  $v \in V_{I,H,O}$  and  $B = \{b_1, \dots, b_k\} \subseteq B_{I,H,O}$ . If  $v = b_1 \oplus \dots \oplus b_k$  then  $B$  is called *decomposition* of  $v$ .

**Theorem 4.1**  $\forall v \in V_{I,H,O}$  there exists, at least, one subset  $B = \{b_1, \dots, b_k\} \subseteq B_{I,H,O}$  such that  $B$  is a decomposition of  $v$ . In other words, any architecture in  $V_{I,H,O}$  can be obtained from superimposition of elements taken from  $B_{I,H,O}$ .

*Proof* Let be  $v \in V_{I,H,O}$  with  $v \subset (\check{I} \times \check{H}) \cup (\check{H} \times \check{O}) \cup (\check{I} \times \check{O})$ .  $\check{I} \times \check{O} \cap v$  represents the set of direct connections from the input to the output neurons, so  $\forall (i_r, o_p) \in \check{I} \times \check{O} \cap v, \{(i_r, o_p)\} \in B_{I,H,O}$  because each  $b_i = \{(i_r, o_p)\}$  is a basic neural architecture with  $\#b_i = 1$ . As it was seen in the definition of valid neural architectures (Definition 4.2), each pair of sets  $b = \{(i_r, h_s), (h_s, o_p)\}$  is also a basic neural architecture with  $\#b = 2$ . The decomposition of the null architecture  $v = \emptyset$  is itself. So  $v$  may be expressed as the superimposition of all basic neural architectures in the way  $\{(i_r, o_p)\}$  such that  $(i_r, o_p) \in \check{I} \times \check{O} \cap v$  with the superimposition of all basic

neural architectures in the way  $\{(i_r, h_s), (h_s, o_p)\}$  such that  $(i_r, h_s) \in \check{I} \times \check{H} \cap v$  and  $(h_s, o_p) \in \check{H} \times \check{O} \cap v$ .  $\square$

**Corollary 4.1** If  $B = \{b_1, \dots, b_k\} \subseteq B_{I,H,O}$  and  $B' = \{b'_1, \dots, b'_{k'}\} \subseteq B_{I,H,O}$  are decompositions of  $v \in V_{I,H,O}$  then  $B \cup B'$  is a decomposition of  $v$ .

*Proof*  $B \cup B' = \{b_1, \dots, b_k, b'_1, \dots, b'_{k'}\}$ . From the definition of decomposition  $v = b_1 \oplus \dots \oplus b_k$  and  $v = b'_1 \oplus \dots \oplus b'_{k'}$ .  $\forall b_i \in B \cup B' b_i = \{(i_r, o_p)\}$  with  $(i_r, o_p) \in \check{I} \times \check{O} \cap v$ , or  $b_i = \{(i_r, h_s), (h_s, o_p)\}$  with  $(i_r, h_s) \in \check{I} \times \check{H} \cap v$  and  $(h_s, o_p) \in \check{H} \times \check{O} \cap v$ , so  $v = b_1 \oplus \dots \oplus b_k \oplus b'_1 \oplus \dots \oplus b'_{k'}$  and thus,  $B \cup B'$  is another decomposition of  $v$ .

**Definition 4.6**  $\forall v \in V_{I,H,O}$ , the decomposition  $M \subseteq B_{I,H,O}$  is called *maximum decomposition* if and only if  $M = B_1 \cup B_2 \cup \dots \cup B_n$ , being  $B_1, \dots, B_n \subseteq B_{I,H,O}$  possible decompositions of  $v$ .

**Remark 4.1** From Definition 4.6, it is clear that every  $v \in V_{I,H,O}$  has one and only one maximum decomposition.

4.1.1 The cardinal of set  $B_{I,H,O}$

The cardinal of set  $B_{I,H,O}$ ,  $\#B_{I,H,O}$  subset of  $V_{I,H,O}$  is now calculated because this result will be used in the next subsection. Let us consider first, the case of basic architectures with no hidden units,  $\#b = 1$ : There is only one direct connection from one of the inputs to one of the outputs. So there is a total of I-O ANNs with these conditions. If  $\#b = 2$  then there are not direct connections, only one output is connected to one input throughout one hidden neuron, existing I-H-O possible combinations with these features. Then, the cardinal of  $B_{I,H,O}$ , adding the null net,  $\#b = 0$ , is:

$$\#B_{I,H,O} = I \cdot O \cdot (H + 1) + 1$$

4.1.2 The binary codification of set  $V_{I,H,O}$

It has been seen how it is possible to build any valid architecture from some basic structures called basic neural architectures, and the existence of exactly  $I \cdot O \cdot (H + 1) + 1$  basic neural architectures. The null architecture is unable to build more complex valid neural architectures but itself because it is the neutral element in the structure defined. However, there are other  $I \cdot O \cdot (H + 1)$  basic neural architectures that can be superimposed to build more complex structures. Thus, the set of basic neural architectures excluding the null net:  $\{b_1, \dots, b_{I \cdot O \cdot (H + 1)}\}$  can be combined in  $2^{I \cdot O \cdot (H + 1)}$  different ways to build valid neural architectures. So, there exists a one-to-one correspondence between the set of all possible decompositions of all valid ANNs and the set  $V_{I,H,O}^b$  of all binary strings that encode each decomposition of valid architectures.

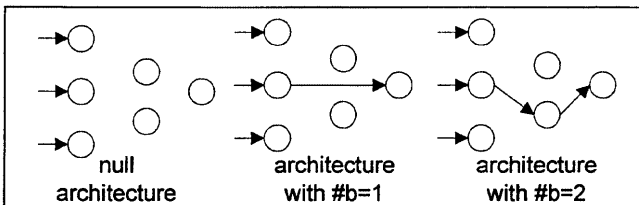


Fig. 6 Three basic neural architectures

With all these premises, the codification of all the points in the search space  $V_{I,H,O}$  will be based on the codification of the set of basic architectures  $B_{I,H,O} = \{b_0, b_1, b_2, \dots, b_i, \dots, b_{I \cdot O(H+1)}\}$  with binary strings of  $I \cdot O(H + 1)$  bit length as shown in the table 1.

$b_0, b_1, \dots, b_{I \cdot O(H+1)}$  may be ordered in any way, but from now on should preserve the same sequence that has been chosen, bearing in mind that the null net  $b_0$  is always encoded as a string of  $I \cdot O(H + 1)$  zeros. The

**Table 1** Binary codification of the basic neural architectures

Basic neural architecture	Binary codification	Comments
$b_0$	$0, 0, \dots, 0, \dots, 0$	The null architecture
$\dots$	$\dots$	
$b_i$	$0, 0, \dots, 1, \dots, 0$	The 1 is set in the $i^{th}$ position
$\dots$	$\dots$	
$b_{I \cdot O(H+1)}$	$0, 0, \dots, 0, \dots, 1$	

**Table 2** Table of correspondences for the binary codification of  $B_{2,1,1}$

Basic neural architecture	Binary codification
	$b_0 = 0000$
	$b_1 = 1000$
	$b_2 = 0100$
	$b_3 = 0010$
	$b_4 = 0001$

**Fig. 7** Codification and decodification processes

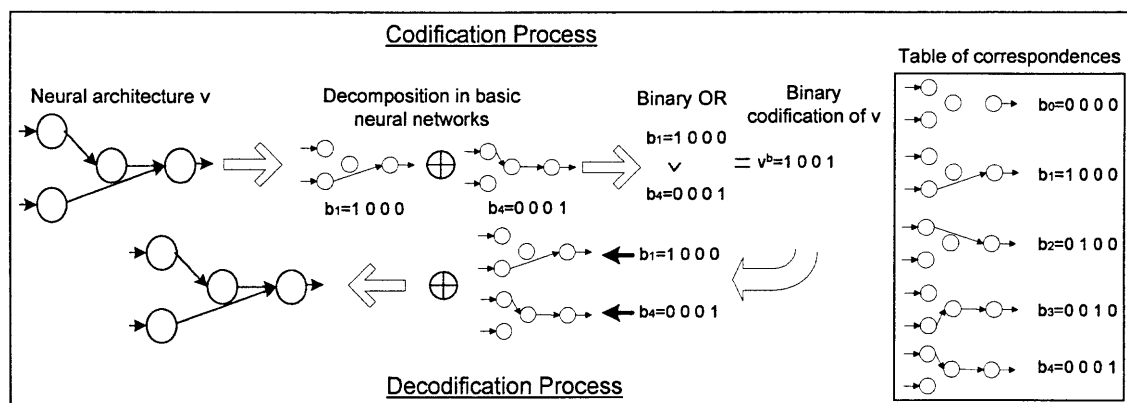


table 2 shows the table of correspondences chosen for the set  $B_{2,1,1}$ .

Once all basic neural architectures have been encoded, it is possible to build any valid architecture included in the search space  $V_{I,H,O}$  by applying the binary-OR operator ( $\vee$ ) to the encoded basic nets; this permits obtaining all binary strings included in the set  $V_{I,H,O}^b$ . Once the table of correspondences is stored, as shown in the example of table 2 between basic neural architectures and their binary codifications, it is easy to encode any architecture  $v \in V_{I,H,O}$  by simply calculating one of the possible decompositions of  $v$  and, starting from a string of  $I \cdot O(H + 1)$  zeros, switching the  $i$  th bit to 1 if the  $i$  th basic neural architecture of the table appears in the decomposition of  $v$ . Figure 7 shows the process of codification and decodification of a concrete neural architecture  $v \in V_{2,1,1}$ .

When this codification is used, the set  $V_{I,H,O}^b$  has two important features: first, the search space defined with the codification proposed yields only possible solutions to the problem, that is, there are not illegal neural architectures. Second, there exist several binary strings that codify the same valid architecture because the set  $V_{I,H,O}^b$  encodes the set of all decompositions of all valid neural architectures and, as it was shown in theorem 4.1, any valid architecture has at least one decomposition. This feature is very desirable when working with genetic algorithms to find faster the best architecture that solves any problem because, generally, several codifications of the best architecture are spread in the search space.

#### 4.2 The Hamming crossover

The basic-architectures codification method has the following feature: given a binary string that represents a valid neural architecture, the variation of one bit in such string results in a very similar neural architecture. These two architectures are only differentiated by the basic neural architecture represented by the modified bit. Thus, only one or connections and one neuron have changed. This feature is very important when working with genetic algorithms because its local search capability is potentiated. The Hamming crossover has been especially designed to make use of this characteristic.

The Hamming crossover works with binary strings of length  $l$ , and it is based on the definition of Hamming distance [22],  $d_H(s,s')$  between two binary strings  $s = (a_0, a_1, \dots, a_{l-1})$  and  $s' = (a'_0, a'_1, \dots, a'_{l-1})$ , where  $a_i, a_j \in \{0,1\}$ . From the set  $G = \{s_1, \dots, s_n\}$  of progenitor strings, which have been chosen from the actual population, two new descendants are obtained,  $o$  and  $o'$ , through the application of the following steps:

- The maximum Hamming distance,  $h$ , between two progenitor strings is calculated: If  $s_{\min}, s_{\max} \in G$  are such that  $d_H(s_{\min}, s_{\max}) \geq d_H(s_i, s_j) \forall s_i, s_j \in G$ , then  $h = d_H(s_{\min}, s_{\max})$ .
- The genetic diversity measure of the population,  $g$ , is calculated:  $g = h/l, g \in [0, 1]$ .
- The offspring is obtained: as it happened in the case of morphological crossover, the genetic diversity measure,  $g$ , guides the behavior of the Hamming crossover adaptatively. If  $g$  takes values near to zero, the genetic diversity of the population is increased to avoid falling in local minima. In the other hand, if  $g$  takes larger values, then the local search capability is increased by generating new strings similar to the progenitors in terms of Hamming distance. This feature is got by employing the function  $\varphi$ , as it was used in the morphological crossover and shown in figure 5. Function  $\varphi$  gives the maximum number of bits,  $n$ , to be modified in the two descendants using the following formula:

$$n = E[l \cdot \varphi(g)]$$

Given  $h = d_H(s_{\min}, s_{\max})$ , the *minimum strings set*, denoted by  $G_{\min}$ , is defined as the set of binary strings at a Hamming distance  $|n|$  from  $s_{\min}$  and  $h-n$  from  $s_{\max}$ :

$$\begin{aligned} G_{\min} &= \{s_1, \dots, s_m\}, d_H(s_{\min}, s_i) \\ &= |n| \text{ y } d_H(s_{\max}, s_i) = h - n, \forall s_i \in G_{\min} \end{aligned}$$

In the same way, the *maximum strings set*, denoted by  $G_{\max}$ , is defined as the set of binary strings at a Hamming distance  $|n|$  from  $s_{\max}$  and  $h-n$  from  $s_{\min}$ :

$$\begin{aligned} G_{\max} &= \{s'_1, \dots, s'_m\}, d_H(s_{\max}, s'_i) \\ &= |n| \text{ y } d_H(s_{\min}, s'_i) = h - n, \forall s'_i \in G_{\max} \end{aligned}$$

The so defined sets  $G_{\min}$  and  $G_{\max}$  assure that:

$$\forall s_i \in G_{\min}, \forall s'_i \in G_{\max}, d_H(s_i, s'_i) = h - 2n$$

Let be  $m \in \{0, \dots, h-2n\}$ , the *offspring set*,  $O_m = \{o_1, \dots, o_p\}$ , is defined as the set of binary strings such that:

- $\forall s \in G_{\min}, \forall s' \in G_{\max}: d_H(o_i, s) = m, d_H(o_i, s') = h-2n-m$ , with  $o_i \in O_m$ , or,
- $\forall s \in G_{\min}, \forall s' \in G_{\max}: d_H(o_i, s) = h-2n-m, d_H(o_i, s') = m$ , with  $o_i \in O_m$ .

Given the offspring set  $O_m = \{o_1, \dots, o_p\}$ , the symmetric offspring set,  $O'_m = \{o'_1, \dots, o'_q\}$ , is defined as the set of binary strings such that:

- If for any string  $s'' \in G_{\min}$ ,  $d_H(o_i, s'') = m$ , with  $o_i \in O_m$ , then  $\forall s \in G_{\min} \forall s' \in G_{\max}: d_H(o'_i, s) = h-2n-m, d_H(o'_i, s') = m, \forall o'_i \in O'_m$ .
- If for any string  $s \in G_{\min}$ ,  $d_H(o_i, s) = h-2n-m$ , with  $o_i \in O_m$ , then  $\forall s \in G_{\min}, \forall s' \in G_{\max}: d_H(o'_i, s) = m, d_H(o'_i, s') = h-2n-m, \forall o'_i \in O'_m$ .

The Hamming crossover operator randomly chooses an offspring set  $O_m$ , from which one of its strings,  $o$ , is taken as the first descendant individual. Then, the symmetric offspring set,  $O'_m$ , is calculated, from which one of its strings,  $o'$ , is randomly selected as the second descendant individual. These two descendants constitute the result given by this operator.

## 5 Results

The experimental results accomplished to illustrate GANN system are related to the convergence speed and size of the networks given as solution. The performance of morphological crossover in training artificial neural networks has been already compared to backpropagation with momentum factor and other real-coded crossovers like Radcliffe's flat crossover and blend crossover [17]. The results reported clearly shows the superiority of morphological crossover in terms of speed of convergence and lower probability of falling in local optima [10]. In this work, three type of tests have been accomplished:

- The basic-architectures codification method with the Hamming crossover, employed in the architectures design module, has been compared to the direct codification method and graph generation system.
- The performance of the Hamming crossover operator is compared to one point, two points, generalized and uniform crossovers. In all cases, the basic architectures method has been used.
- GANN system has been employed in one real-world task: the generation of artificial neural networks for the breast cancer diagnosis. This task is part of a complete system that detect and diagnosis suspicious masses in the breast tissue of being a carcinoma, taken as input a complete set of views (oblique, crano-caudal and lateral) of digitalized mammograms from both breasts. The results given by GANN are compared to GANNET system [23]. This system, similarly to GANN, generates and trains artificial neural networks by genetic algorithms and has been chosen to be compared to GANN because GANNET is one of the evolutionary systems that better results has reported.

Similar genetic settings have been chosen for the three different experiments. Thus, we used a proportional reproduction strategy in which reproduction probability is decided according to the fitness of each individual. The probability of mutation has been set to 0.05, and the crossover operator has been used with a probability of

0.6. For each experiment we ran 100 trials of 1000 generations and the mean value calculated. The fitness of each neural network of the population is calculated as:

$$f = \text{MSE}_{it} \frac{C_a}{C_t}$$

being  $C_a$  the number of connections existing in the actual neural network and  $C_t$  the maximum number of connections allowed by the codification.  $\text{MSE}_{it}$  is the mean square error given by the network after  $it$  learning iterations have been run. Morphological crossover have been employed as the learning algorithm.

### 5.1 The basic-architectures codification

GANN system employs the basic-architectures codification method to design neural architectures. The coder/decoder problem has been studied to show the speed of convergence and the size of the neural architectures obtained as solutions to this problem when the codification method proposed is used in comparison to the direct and graph generation system encoding methods.

Figure 8 shows the results, in terms of convergence speed, of searching for the best network that solves the encoder/decoder problem encoding architectures of a maximum of four input and output units and up to 8 hidden neurons (4-8-4). The fitness of the individuals has been calculated after  $it = 50,000$  learning iterations have been run and the averaged MSE of the ten best individuals of the population is plotted against the generation.

Apart from the fact that the proposed method clearly outperforms the other methods, it is important to notice that, in these experiments, the evolution line of our model tends to converge much faster after several iterations have occurred. The table 3 shows the final solu-

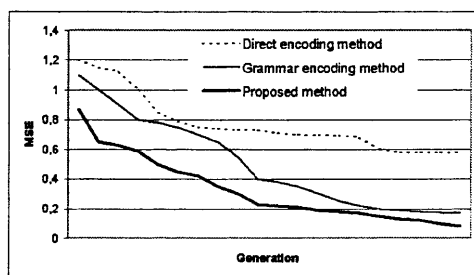


Fig. 8 Convergence process in the coder/decoder problem

Table 3 Final solutions of the coder/decoder

	Direct encoding	Grammer encoding	Proposed method
4-0-4	42%	58%	74%
4-1-4	41%	32%	15%
4-2-4	17%	10%	11%

tions given by each method. From this table, it can be observed that the proposed method not only converges faster, but also gets smaller neural networks. Using the proposed method, architectures 4-0-4 and 4-1-4 with direct connections from the inputs to the outputs are obtained in the 89% of the executions. This is computationally very desirable when these networks are working as part of an intelligent system to solve the problem they are designed to.

### 5.2 The Hamming crossover

The Hamming crossover operator is used by the architecture design module to increase the convergence speed and to avoid falling in local optima with respect to other crossover operators when basic-architectures codification is employed. Test results are shown in solving the coder/decoder problem, encoding architectures of a maximum of four input and output units and up to 8 hidden neurons. The genetic parameters are similar the ones used in previous section. Figure 9 shows the convergence process of GANN towards the optimum neural network using Hamming, one point, two points, uniform and generalized crossover operators. For each of them, the averaged fitness after  $it = 50,000$  learning iterations of the ten best individuals of the population is plotted against the generation.

Table 4 reports the neural networks obtained to solve the coder/decoder problem with each of the five crossover operators used with the basic-architectures codification method. The Hamming crossover clearly outperforms the other operators. 84% of the executions run with the Hamming crossover gave as a result 4-0-4 and 4-1-4 neural networks. The generalized crossover, which gave better results after Hamming operator, obtains the two best architectures the 68% of the cases.

### 5.3 Breast cancer diagnosis with GANN

GANN system has been employed to a real-world problem: the diagnosis of suspicious masses presented in the breast tissue that could be a carcinoma. This application is part of a whole project, still in progress, for the automatic detection and diagnosis in real time of breast pathologies. The system built takes as input a complete set of views of digitalized mammograms from both patient's breasts and searches for microcalcifications and suspicious masses, the two main abnormalities that can be found in a mammography. The figure 10 shows a general scheme of the whole system where it can be seen how it is divided into two subsystems already finished that work parallelly: the microcalcifications detection subsystem and the masses detection subsystem. Each of these subsystems give as result a set of characteristics of each abnormality found. In this work, we focus on the masses detection subsystem [24, 25]. The output it gives for each mass found consists on a vector of ten real

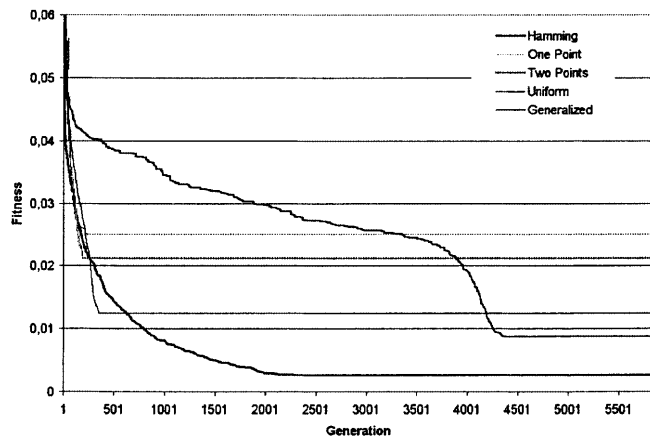


Fig. 9 Convergence process in the coder/decoder problem using different crossovers

Table 4 Final solutions of the coder/decoder for each of the five crossover operators

	Hamming	One point	Two points	Uniform	Generalized
4-0-4	74%	0%	0%	0%	38%
4-1-4	15%	0%	0%	0%	30%
4-2-4	11%	0%	0%	6%	18%
4-3-4	0%	0%	0%	13%	12%
4-4-4	0%	19%	23%	26%	2%
Others	0%	81%	77%	55%	0%

values meaning: first, the radius of the mass (mean of distances from center to points on the perimeter), the texture (standard deviation of gray-scale values), the perimeter, the area, compactness ( $\text{perimeter}^2 / \text{area} - 1$ ), concavity (severity of concave portions of the contour), symmetry of the mass, standard deviation of the distances taken from the center of the mass to the perimeter, the biggest distance and the most concave portion of the perimeter. These features have been chosen after several interview sessions have been arranged with expert radiologists that work in the project.

GANN system takes a training set of 330 patterns with ten real valued inputs (one for each mass characteristic) and one output with two possible values:  $-1$  if the mass is considered as benign, and  $+1$  if it is considered as malign. GANN must give the minimum artificial neural network trained to solve such problem. The results given by GANN in terms of speed of convergence and size of the neural networks given as solution are compared to GANNET. In both cases the genetic parameters used are the same as in previous sections.

The figure 11 the convergence progress of GANN and GANNET towards the optimum neural network. For each of them, the averaged fitness of the ten best individuals of the population is plotted against the generation. For both systems, architectures of a maximum of ten input units, one output and up to one

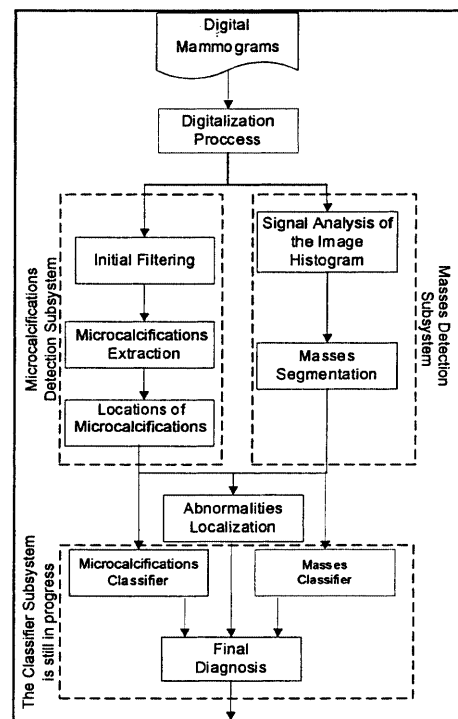


Fig. 10 General Scheme for computerized detection of masses and microcalcifications in digitalized mammograms

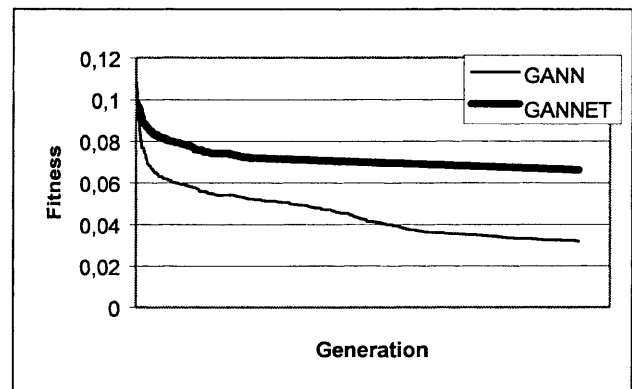


Fig. 11 Convergence process in the cancer diagnosis task for GANN and GANNET

hundred hidden neurons have been encoded. This means, in the case of the basic architectures codification, that the string length of the individuals is about one thousand bits.

Table 5 shows the number of connections of the neural networks obtained as solution for the 100 trials run. In the third row of this table, the interval [25–30] represents the neural networks obtained as solution having between 25 and 30 connections. The high performance and quality of the solutions given by GANN system is very clear in this experiment. The differences with respect to other algorithms are bigger because the

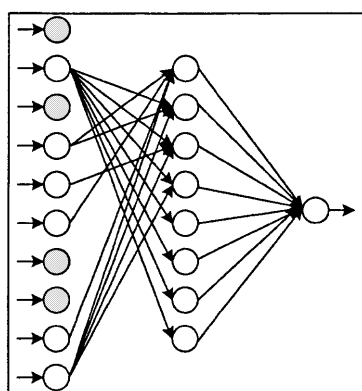
dimension of the search space and the amount of data to handle have increased substantially.

The figure 12 shows the minimum neural network that solves the problem, which has been obtained the 62% of the trials run as seen in table 5. It is important to notice that direct connections from the inputs to the output were not needed and how GANN has eliminated four inputs. GANN system can make, implicitly, a sensitivity analysis of the problem and eliminate all those variables that does not have or very little importance for the output. This important feature of GANN is given by the basic architectures codification method that implements. In this case, the input variables eliminated were the first, corresponding to the radius of the mass, the third, which is the perimeter, the seventh corresponding to the symmetry of the mass and, finally, the eighth, which is the standard deviation of the distances from the centre to the perimeter. GANNET is unable to erase input neurons and so, its solutions are bigger.

These variables eliminated by GANN have a clear relation: the radius and the perimeter represent the size of the mass, while the other two variables represent the shape of the border of the mass. The radiologists that work in the project agreed that the size of a mass has little relevance, although sometimes can give additional information. The system has not completely eliminated the information related to the size of the mass because the area, which has not been eliminated, can substitute the radius and perimeter with this goal. Something similar occurs with the other two variables eliminated: the symmetry and the standard deviation of the radiuses, which can be substituted by the concavity.

**Table 5** Number of connections of the neural networks solution

Number of connections	GANN	GANNET
24	62%	23%
[25–30]	31%	14%
[31–40]	7%	31%
More than 40	0%	32%



**Fig. 12** The minimum neural network that can solve the problem

The neural network solution, shown in figure 12 has been tested using 240 testing patterns that have not been presented to the network during its training process with a 3.3% of error rate. From the 240 testing patterns, 180 were malignant masses and the other 60 were benignant. Only one from the 180 malignant cases was incorrectly classified by the network, while, in case of the benignant cases the network made 7 mistakes. It is very important for radiologists that the probability of error classifying malignant cases be lower than in the case of benignant cases. This is because it is much more dangerous to say to a patient that the mass found in her breast tissue is benignant when it is really a cancer than in the opposite case.

## 6 Conclusions

GANN is a system capable of designing and training one-hidden layer generalized feed-forward neural networks. The basic-architectures codification method proposed to be used in the architectonic design module allows a neural network topology to be represented through elemental structures known as basic neural architectures. This approach exhibits several advantages. Firstly, it provides a clear representation of the structure of the networks encoded. The proposed scheme requires short codification length, being the optimum when one output neural networks are encoded. This is due, partially, to the fact that illegal neural networks are not codified and so, the search space is smaller. Finally, the proposed approach generates regular patterns, preserving the meaningful subcircuits discovered to improve the networks existing in the population. This is possible because one bit variation of the binary string results in a very similar neural architecture where only one or two connections have been changed. This feature is potentiated by employing the Hamming crossover, which generates the offspring near the progenitors in terms of the definition of Hamming distance, improving the fine local search near the optimum architecture and, this way, increasing the speed of convergence and reducing the size of the networks obtained as solutions.

The training module of GANN system implements real-coded genetic algorithms through the morphological crossover. This operator, which can be applied on any optimization problem, is used by GANN to minimize the MSB in the training process of the networks generated by the architectonic design module. Morphological gradient operation has been reinterpreted to build this crossover, giving a heterogeneity measure of the population in order to, dynamically, extend or make narrower the interval defined by the parents. This effect balances adequately the exploration and exploitation capabilities of the genetic algorithm, allowing high speed searching and avoiding falling in local optima.

The results section showed that GANN obtains the smallest architecture that solves the problem. Smaller

neural networks allow, in general, higher generalization capabilities than larger ones. Run time response speed is also higher when using small neural networks because they have less number of processing elements. GANN also accomplishes sensitivity analysis, removing all those input variables that have no effect to the outputs. This feature could be seen in the problem of generating a neural network to diagnose the breast cancer where four input neurons were eliminated. These capabilities of finding the smallest neural network and sensitivity analysis are given to GANN by the basic architectures codification method and the morphological crossover, which trains the neural networks candidates to solve the problem. GANN also exhibits high speed of convergence. This feature is given by the combination of the application of the basic architectures codification method and the Hamming crossover. As it was seen in the results, if one of these two techniques are changed by any other, then the convergence speed is seriously decreased.

---

## References

1. Konar A. (2000) *Artificial Intelligence and Soft Computing*. CRC Press, Boca Raton, Florida
2. Mars P, Chen JR, Nambiar R (1996) *Learning Algorithms: Theory and Applications in Signal Processing, Control and Communications*. CRC Press, New York
3. Principe JC, Euliano NR, Lefebvre WC (2000) *Neural and adaptive systems, fundamentals through simulations*. Wiley & Sons, New York
4. Manrique D (2001) *Neural networks design and new optimization methods by genetic algorithms*. PhD thesis, Universidad Politécnica de Madrid, Madrid
5. Sato A, Yamada K, Tsukumo J, Temma T (1991) *Neural network models for incremental learning*. Artificial Neural Networks. Elsevier Science Publishers
6. Musavi MT, Ahmed W, Chan KF, Paris KB, Hummels DM (1992) On the training of radial basis function classifiers. *Neural Networks* 5:595–603
7. Anders U, Korn O (1999) Model selection in neural networks. *Neural Networks* 12:309–323
8. Sweatman C, Mulgrew B, Gibson G (1998) Two algorithms for neural-network design and training with application to channel equalization. *IEEE Transactions on Neural Networks* 9(3):533–542
9. Braun H (1996) On optimizing large neural networks (multi-layer perceptrons) by learning and evolution. *Zeitschrift für Angewandte Mathematik und Mechanik* 76(1):211–214
10. Barrios D, Carrascal A, Manrique D, Ríos J (2000) Neural network training using real-coded genetic algorithms. In: *Proceedings of the 5th Ibero-American Symposium on Pattern Recognition* Lisbon, Portugal pp. 337–346
11. Brown AD, Card HC (2000) Cooperative coevolution of neural representations. *International Journal of Neural Systems* 10(4):311–320
12. Hussain TS, Browse RA (1998) Including control architecture in attribute grammar specifications of feedforward neural networks. In: *Proceedings of the 1998 Joint Conference on Information Sciences: 2nd International Workshop on Frontiers in Evolutionary Algorithms*, North Carolina, USA vol. 2 pp. 432–436
13. Michalewicz Z (1999) *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York
14. Dorado J (1999) *Cooperative Strategies to Select Automatically Training Patterns and Neural Architectures with Genetic Algorithms*. PhD Thesis, University of La Coruña, Spain
15. Kitano H (1990) *Designing neural networks using genetic algorithms with graph generation system*. *Complex Systems* 4:461–476
16. Radcliffe NJ (1990) *Genetic neural networks on MIMD computers*. PhD thesis, University of Edinburgh, Edinburgh, UK
17. Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms and interval-schemata. *Foundations of Genetic Algorithms* 2:187–202
18. Ono I, Kobayashi S (1997) A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover. In: *Proceedings of 7th International Conference on Genetic Algorithms* pp. 246–253
19. Crespo J (1993) *Morphological connected filters and intra-region smoothing for image segmentation*. PhD thesis, Georgia Institute of Technology, Atlanta
20. D'alotto LA, Giardina CR (1998) *A unified signal algebra approach to two-dimensional parallel digital signal processing*. Marcel Dekker, New York
21. Barrios D, Manrique D, Porras J, Ríos J (2000) Real-Coded genetic algorithms based on mathematical morphology. In: *Proceedings of the 3rd International Conference on Statistical Techniques in Pattern Recognition*, Alicante, Spain. *Lecture Notes in Computer Science: Advances in Pattern Recognition*. Ed. Springer-Verlag pp. 706–715
22. Hamming RW (1950) Error detecting and error correcting codes. *Bell System, Technical Journal* 29:147–160
23. Robbins GE, Plumbley MD, Hughes JC, Fallside F, Pager R (1993) Generation and adaptation of neural networks by evolutionary techniques (GANNET). *Neural Computing and Applications* 1:23–31
24. Giménez V, Manrique D, Ríos J, Vilarrasa A (1999) Iterative Method for Automatic Detection of Masses in Digital Mammograms for Computer-Aided Diagnosis. In: *Proceedings of SPIE'99 on Image Processing*, San Diego, CA, USA pp. 1086–1093
25. Manrique D, Porras J, Ríos J, Vilarrasa A (1999) Nodule Border Detection Algorithm in Digital Mammograms for Computer-Aided Diagnosis. In: *Proceedings of Computer Assisted Radiology and Surgery CARS99*, Paris, France pp. 1015