COVNET: A Cooperative Coevolutionary Model for Evolving Artificial Neural Networks

Nicolás García-Pedrajas, *Associate Member, IEEE*, César Hervás-Martínez, and José Muñoz-Pérez

***Abstract*—This paper presents COVNET, a new cooperative coevolutionary model for evolving artificial neural networks. This model is based on the idea of coevolving subnetworks that must cooperate to form a solution for a specific problem, instead of evolving complete networks. The combination of this subnetworks is part of a coevolutionary process. The best combinations of subnetworks must be evolved together with the coevolution of the subnetworks. Several subpopulations of subnetworks coevolve cooperatively and genetically isolated. The individual of every subpopulation are combined to form whole networks. This is a different approach from most current models of evolutionary neural networks which try to develop whole networks. COVNET places as few restrictions as possible over the network structure, allowing the model to reach a wide variety of architectures during the evolution and to be easily extensible to other kind of neural networks. The performance of the model in solving three real problems of classification is compared with a modular network, the adaptive mixture of experts and with the results presented in the bibliography. COVNET has shown better generalization and produced smaller networks than the adaptive mixture of experts and has also achieved results, at least, comparable with the results in the bibliography.**

***Index Terms*—Cooperative coevolution, evolutionary computation, evolutionary programming, genetic algorithms, neural-networks automatic design.**

I. Introduction

IN THE AREA of neural-networks design [1], one of the main problems is finding suitable architectures for solving specific problems. The election of such architecture is very important, as a network smaller than needed would be unable to learn and a network larger than needed would end in overtraining.

The problem of finding a suitable architecture and the corresponding weights of the network is a very complex task (for a very interesting review of the matter the reader can consult [2]). Modular systems are often used in machine learning as an approach for solving these complex problems. Moreover, in spite of the fact that small networks are preferred because they lead to better performance, the error surfaces of such networks are more rugged and have few good solutions [3]. In addition, there is much neuropsychological evidence showing that the brain of humans and other animals consists of modules, which are subdivisions in identifiable parts, each one with its own purpose and function [4]. The objective of this work is the design of such modular neural networks.

Evolutionary computation [5], [6] is a set of global optimization techniques that have been widely used in late years for training and automatically designing neural networks (see Section II). Some efforts have been made in designing modular [7] neural networks with these techniques (e.g., [8]), but in almost all of them the design of the networks is helped by methods outside evolutionary computation, or the application area for those models is limited to very specific architectures.

Cooperative coevolution [9] is a recent paradigm in the area of evolutionary computation focused on the evolution of coadapted subcomponents without external interaction. In cooperative coevolution a number of species are evolved together. The cooperation among the individuals is encouraged by rewarding the individuals based on how well they cooperate to solve a target problem. The work on this paradigm has shown that cooperative coevolutionary models present many interesting features, such as specialization through genetic isolation, generalization and efficiency [10]. Cooperative coevolution approaches the design of modular systems in a natural way, as the modularity is part of the model. Other models need some *a priori* knowledge to decompose the problem *by hand*. In many cases, either this knowledge is not available or it is not clear how to decompose the problem.

This paper describes a new cooperative coevolutionary model called COVNET (some preliminary results on COVNET have been published in [11]). This model develops subnetworks instead of whole networks. These modules are combined forming groups that make up a network. As Potter and de Jong [10] have stated, *"to apply evolutionary algorithms effectively to increasingly complex problems explicit notions of modularity must be introduced to provide reasonable opportunities for solutions to evolve in the form of interacting coadapted subcomponents."*

The most distinctive feature of COVNET is the coevolution of modules without the intervention of any agent external to the evolutionary process and without a gating network or another alternative mechanism for combining subnetworks. Also, the use of an evolutionary algorithm for the evolution of both the weights and the architecture allows the model to be applied to tasks where there is no error function that could be defined (e.g., game playing [12] or control [13]) in order to apply an algorithm based on the minimization of that error, like the backpropagation learning rule.

The most important contributions of COVNET are the following. First, it forms modular artificial neural networks using

Manuscript received October 16, 2000; revised February 25, 2002 and August 5, 2002. This work was supported in part by the Spanish Comisión Interministerial de Ciencia y Tecnología under Project ALI98-0676-CO2-02.

N. García-Pedrajas and C. Hervás-Martínez are with the Department of Computing and Numerical Analysis, University of Córdoba, 14071 Córdoba, Spain (e-mail: ngarcia-pedrajas@acm.org; ma1hemac@uco.es).

J. Muñoz-Pérez is with the Department of Languages and Computer Science, University of Málaga, Málaga 29071, Spain (e-mail: munozp@lcc.uma.es).

Digital Object Identifier 10.1109/TNN.2003.810618

cooperative coevolution. Every module must learn how to combine with the other modules of the evolved network to be useful. Introducing the combination of nodules into the evolutionary process enforces the cooperation among the modules, as independently evolved modules are unlikely to combine well after the evolutionary process have finished.

Second, it develops a method for measuring the fitness of cooperative subcomponents in a coevolutionary model. This method, based on three different criteria, could be applied to other cooperative coevolutionary models not related to the evolution of neural networks. The current methods are based, almost exclusively, on measuring the fitness of the networks where the module appears.

Third, it introduces a new hybrid evolutionary programming algorithm that puts very few restrictions in the subnetworks evolved. This algorithm produces very compact subnetworks and even the evolved subnetworks alone achieved very good performance in the test problems, as it will be shown in the experimental section.

This paper is organized as follows: Section II makes a short overview of the methods for designing neural networks. Section III explains the proposed model. Section IV makes a comparison between our model and a modular neural network in the task of solving three classification problems. Section V makes an experimental analysis of some features of the populations evolved. Section VI presents the conclusions of our work and states some future work that would be interesting to do in order to improve the model.

II. Automatic Design of Artificial Neural Networks

The automatic design of artificial neural networks has two basic sides: parametric learning and structural learning. In structural learning, both architecture and parametric information must be learned through the process of training. Basically, we can consider three models of structural learning: Constructive algorithms, destructive algorithms, and evolutionary computation.

Constructive algorithms [14]–[16] start with a small network (usually a single neuron). This network is trained until it is unable to continue learning, then new components are added to the network. This process is repeated until a satisfactory solution is found. These methods are usually trapped in local minima [17] and tend to produce big networks. Destructive methods, also known as pruning algorithms [18], start with a big network, that is able to learn but usually ends in over-fitting and try to remove the connections and nodes that are not useful. A major problem with pruning methods is the assignment of credit to structural components of the network in order to decide whether a connection or node must be removed.

Both methods, constructive and destructive, limit the number of available architectures, thus introducing constraints in the search space of possible structures that may not be suitable to the problem. Although these methods have been proved useful in simulated data [19], [20], their application to real problems has been rather unsuccessful [21]–[23].

Evolutionary computation has been widely used in the late years to evolve neural-network architectures and weights. There have been many applications for parametric learning [24] and for both parametric and structural learning [8], [17], [25]–[32]. These works fall in two broad categories of evolutionary computation: genetic algorithms and evolutionary programming.

Genetic algorithms are based on a representation independent of the problem, usually the representation is a string of binary, integer, or real numbers. This representation (the genotype) codifies a network (the phenotype). This is a dual representation scheme. The ability to create better solutions in a genetic algorithm relies mainly on the operation of *crossover*. This operator forms offspring by recombining representational components from two members of the population.

The benefits of crossover come from the ability of forming connected substrings of the representation that correspond to above-average solutions [5]. This substrings are called *building blocks*. Crossover is not effective in environments where the fitness of an individual of the population is not correlated with the expected ability of its representational components [33]. Such environments are called *deceptive* [34]. Deception is a very important feature in most representations of neural networks, so crossover should be avoided in evolutionary neural networks [17].

One of the most important forms of deception arises from the many-to-one mapping from genotypes in the representation space to phenotypes in the evaluation space. The existence of networks functionally equivalent and with different encodings makes the evolution inefficient and it is unclear whether crossover would produce more fitted individuals from two members of the population. This problem is usually termed as the *permutation problem* [35], [36], or the *competing conventions problem* [37].

Evolutionary programming [38] is, for many authors, the most suited paradigm of evolutionary computation for evolving artificial neural networks [17]. Evolutionary programming uses a representation natural for the problem. Once the representation scheme has been chosen, mutation operators specific to the representation scheme are defined. Evolutionary programming offers a major advantage over genetic programming when evolving artificial neural networks, the representation scheme allows manipulating networks directly, avoiding the problems associated with a dual representation that we have discussed.

The use of evolutionary learning for designing neural networks dates from no more than two decades (see [2] or [37] for reviews). However, a lot of work has been made in these two decades, leaving many different approaches and working models, for instance, [8], [25], or [30]. Evolutionary computation has been used for learning connection weights and for learning both architecture and connection weights. The main advantage of evolutionary computation is that it performs a global exploration of the search space avoiding to become trapped in local minima as usually happens with local search procedures.

Miller *et al.* [39] proposed that evolutionary computation is a very good candidate to be used to search the space of topologies because the fitness function associated with that space is complex, noisy, nondifferentiable, multimodal, and deceptive.

Almost all the current models try to develop a global architecture, which is a very complex problem. Although, some attempts have been made in developing modular networks [40],

[41], in most cases the modules are combined only after the evolutionary process has finished and not following a cooperative coevolutionary model.

Few authors have devoted their attention to the cooperative coevolution of subnetworks. Some authors have termed this kind of cooperative evolution (where the individuals must cooperate to achieve a good performance) *symbiotic evolution* [42]. More formally, we should speak of *mutualism*, that is, the cooperation of two individuals from different species that benefits both organisms.

Smalz and Conrad [26] developed a cooperative model that had some similarities with COVNET. In this model there are two populations: a population of nodes, divided into clusters and a population of networks that are combinations of neurons, one from each cluster. Both populations are evolved separately.

Whitehead and Choate [29] developed a cooperative-competitive genetic model for radial-basis function (RBF) neural networks. In this work there is a population of genetically encoded neurons that evolves both the centers and the widths of the RBFs. There is just one network that is formed by the whole population of RBFs. The major problem, as in our approach, is to assign the fitness to each node of the population, as the only performance measure available is for the whole network. This is well known as the "credit apportionment problem"[1] [9], [26]. The credit assignment used by Whitehead and Choate is restricted to RBF-like networks and very difficult to adapt to other kind of networks.

Opitz and Shavlik [44] developed a model closer to COVNET, called Accurate anD Diverse Ensemble Maker giving United Predictions (ADDEMUP). They evolved a population of networks by means of a genetic algorithm and combined the networks in an ensemble with a linear combination. The competition among the networks is encouraged with a diversity term added to the fitness of each network.

Moriarty and Miikkulainen [32], [42] developed an actual cooperative model, called SANE, that had some common points with Smalz and Conrad [26]. In that paper, they propose two populations: one of nodes and another of networks that are combinations of the individuals from the population of nodes. Zhao *et al.* [45] proposed a framework for cooperative coevolution and applied that framework to the evolution of RBF networks. Nevertheless, their work, more than a finished model, is an open proposal that aims at the definition of the problems to be solved in a cooperative environment.

Cho and Shimohara [4] developed a modular neural network evolved by means of genetic programming. Each network is a complex structure formed by different modules which are codified by a tree structure.

Yao and Liu [31] use the final population of networks developed using the EPNet [8] model to form ensembles of neural networks. The combination of these networks produced better results than any isolated network. Nevertheless, the cooperation among the networks takes place only after the evolutionary process has finished. So, the model is neither cooperative nor coevolutive.

[1]This problem can be traced back to the earliest attempts to apply machine learning to playing the game of checkers by Samuel [43] in 1959.

III. COVNET: Cooperative Coevolutionary Model

COVNET is a cooperative coevolutionary model, that is, several species are coevolved together. Each species is a subnetwork that constitutes a partial solution of a problem; the combination of several individuals from different species makes up the network that must be applied to the specific problem. The population of subnetworks, that are called *nodules*, is made up by several subpopulations[2] that evolve independently. Each one of these subpopulations constitutes a species. The combination of individuals from these different subpopulations that coevolve together is the key factor of our model.

The evolution of coadapted subcomponents must address four major issues: problem decomposition, interdependence among subcomponents, credit assignment and maintenance of diversity. Cooperative coevolution gives a framework where these issues could be faced in a natural way. The problem decomposition is intrinsic in the model. Each population will evolve different species that must cooperate in order to be rewarded with high fitness values. There is no need to any *a priori* knowledge to decompose the problem *by hand*. The interdependence among the subcomponents comes from the fact that the fitness of each individual depends on how well the individual works together with the members of other species. Credit assignment is made using a model developed in this work (see Section III-C). The diversity is maintained along the evolution due to the fact that each species is evolved without exchanging genetic material among them. This is an important aspect of our cooperative coevolutionary model. Exchanging genetic material between two different species (that it, subpopulations) will usually produce nonviable offspring. Moreover, the mixing of genetic material might reduce the diversity of the populations.

A nodule is made up of a variable number of nodes with free interconnection among them (see Fig. 1), that is, each node could have connections from input nodes, from other nodes of the nodule and to output nodes. More formally, a nodule could be defined as follows.

Definition 1: (*Nodule*) A nodule is a subnetwork formed by: a set of nodes with free interconnection among them, the connection of these nodes from the input and the connections of the nodes to the output. It cannot have connections with any node belonging to another nodule.

The input and output layers of the nodules are common, they are the input and output layers of the network. It is important to note that the genotype of the nodule has a one-to-one mapping to the phenotype, as the many-to-one mapping between them is one of the main sources of deception and the permutation problem [17].

In the same way, we define a network as a combination of nodules. The definition more formally is as follows.

Definition 2: (*Network*) A network is the combination of a finite number of nodules. The output of the network is the sum of the outputs of all the nodules that constitute the network.

In practice, all the networks of a population must have the same number of nodules and this number, N, is fixed along the evolution.

[2]Each subpopulation evolves independently, so we can talk of subpopulations or species indistinctly, as each subpopulation will constitute a different species.

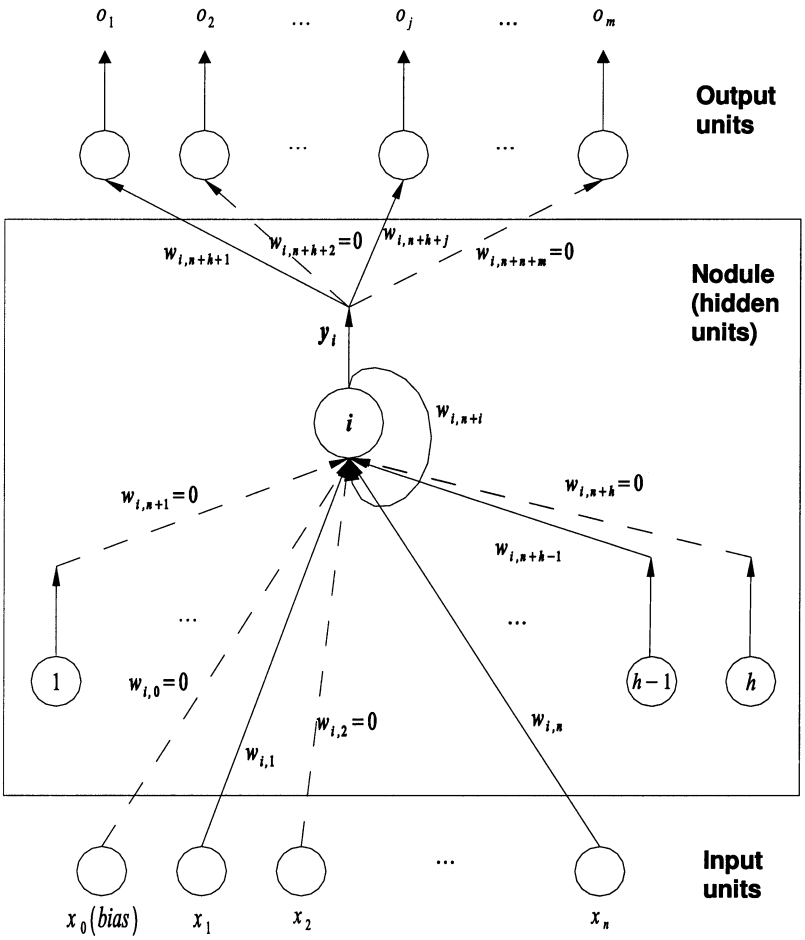

Fig. 1. Model of a nodule. As a node has only connections to some nodes of the nodule, the connections that are missing are represented with dashed lines. The nodule is composed by the hidden nodes and the connections of these nodes from the input and to the output.

Some parameters of the nodule are given by the problem and, for that reason, they are common to all the nodules:

n number of inputs;
m number of outputs;
$\mathbf{x} = (\mathbf{1}, \mathbf{x_1}, \ldots, \mathbf{x_n})$ input vector;
f^{output} transfer function of the output layer.

These parameters are fixed for all nodules. The rest of the parameters depend on each nodule:

h number of (hidden) nodes of the nodule;
f^i transfer function of node i;
p^i partial output of node i (see explanation below);
y^i output of the node i;
$\mathbf{w}_i$ weight vector of node i.

As the node has a variable number of connections we have considered, for simplicity, that the connections that are not present in the node have weight 0, so we can use a weight vector of fixed length for all nodes. A node could have connections from input nodes, from other nodes and to output nodes. The weight vector is ordered as follows:

$$\mathbf{w}_i = (\overbrace{w_{i,0}}^{\text{bias}}, \overbrace{w_{i,1}, \ldots, w_{i,n}}^{\text{input}}, \overbrace{w_{i,n+1}, \ldots, w_{i,n+h}}^{\text{hidden}}, \overbrace{w_{i,n+h+1}, \ldots, w_{i,n+h+m}}^{\text{output}}). \quad (1)$$

As there is no restriction in the connectivity of the nodule the transmission of the impulse along the connections must be defined in a way that avoids recurrence as the aim of these work is the cooperative coevolution of feed-forward neural networks. The transmission has been defined in three steps.

Step 1) Each node generates its output as a function of only the inputs of the nodule (that is, the inputs of the whole network)

$$p_i = f^i \left(\sum_{j=0}^{n} w_{i,j} x_j \right) \quad (2)$$

this value is called *partial output*.

Step 2) These partial outputs are propagated along the connections. Then, each node generates its output as a function of all its inputs

$$y_i = f^i \left(\sum_{j=0}^{n} w_{i,j} x_j + \sum_{j=1}^{h} w_{i,n+j} p_j \right). \quad (3)$$

Step 3) Finally, the output layer of the nodule generates its output

$$o_j = f^{\text{output}} \left(\sum_{i=1}^{h} w_{i,n+h+j} y_i \right). \quad (4)$$

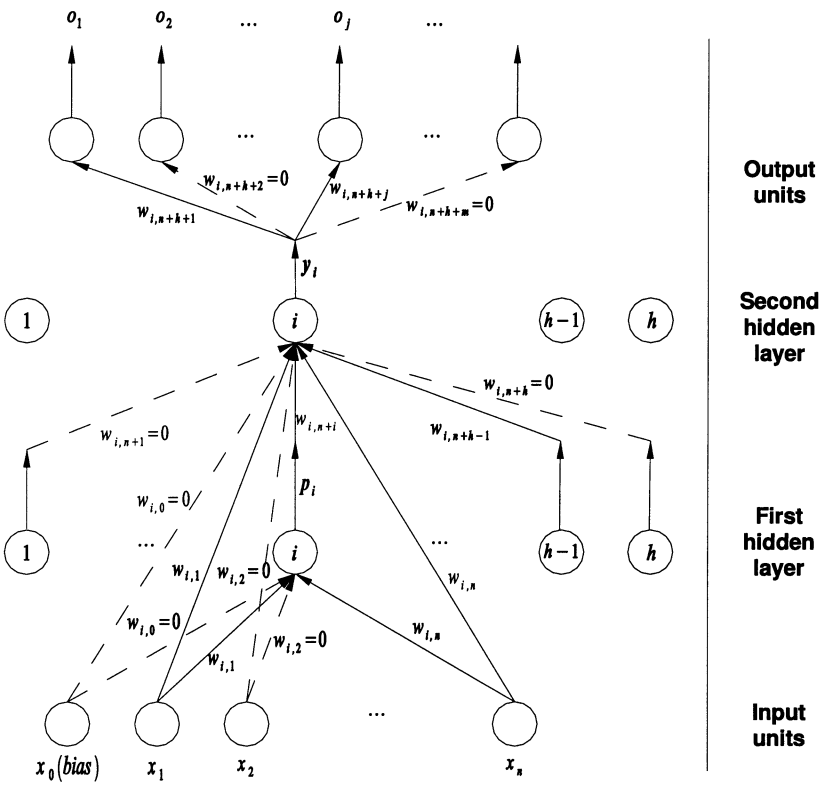

Fig. 2. Equivalent model with two hidden layers. Every connection from an input node represents two connections, as the input value is used in two steps [see (2) and (3)]. Every connection from another node of the nodule represents a connection between the first and second hidden layer [see (3)].

These three steps are repeated over all the nodules. The actual output vector of the network is the sum of the output vectors generated by each nodule.

Defined in this way a nodule is equivalent to a subnetwork of two hidden layers with the same number of nodes in both layers. This equivalent model is shown on Fig. 2. So, the nodule of Fig. 1 could be seen as the genotype of a nodule whose phenotype is the subnetwork shown on Fig. 2. This difference is important, as the model of Fig. 1 considered as a phenotype would be a recurrent network. In this representation, the mapping from genotype to phenotype is one-to-one, so the deception problem above mentioned does not appear.

As the nodules must coevolve to develop different behaviors we have N_P independent subpopulations of nodules[3] that evolve separately. The network will always have N_P nodules, each one from one different subpopulation of nodules. Our task is not only developing cooperative nodules but also obtaining the best combinations. For that reason we have also a population of networks. This population keeps track of the best combinations of nodules and evolves as the population of nodules evolves. The whole evolutionary process is shown on Fig. 3.

Niche creation is implicit, as the subpopulations must coevolve complementary behaviors in order to get useful networks, as the combination of several nodules with the same behavior when they receive the same inputs would not produce networks with a good fitness value. So, there is no need of calculating a *fitness sharing* measure that can bias the evolutionary process.

[3]In order to maintain a coherent nomenclature we talk of one population of networks and another population of nodules. The population of nodules is divided into N_P genetically isolated subpopulations that coevolve together.

In the next two sections, we will explain in depth the two populations and their evolutionary process.

A. Nodule Population

The nodule population is formed by N_P subpopulations. Each subpopulation consists of a fixed number of nodules codified directly as subnetworks, that is, we evolve the genotype of Fig. 1 that is a one-to-one mapping to the phenotype of Fig. 2. The population is subject to the operations of replication and mutation. Crossover is not used due to its disadvantages in evolving artificial neural networks [17]. With these features the algorithm falls in the class of evolutionary programming [38].

There is no limitation in the structure of the nodule or in the connections among the nodes. There is only one restriction to avoid unnecessary complexity in the resulting nodules, there can be no connections to an input node or from an output node.

The algorithm for the generation of a new nodule subpopulation is similar to other models proposed in the bibliography, such as GNARL [17], EPNet [8], or the genetic algorithm developed by Bebis *et al.* [30]. The steps for generating the subpopulations are the following.

- The nodules of the initial subpopulation are created randomly. The number of nodes of the nodule, h, is obtained from a uniform distribution: $0 \leq h \leq h_{\max}$. Each node is created with a number of connections, c, taken from a uniform distribution: $0 \leq c \leq c_{\max}$. The initial value of the weights is uniformly distributed in the interval $[w_{\min}, w_{\max}]$.

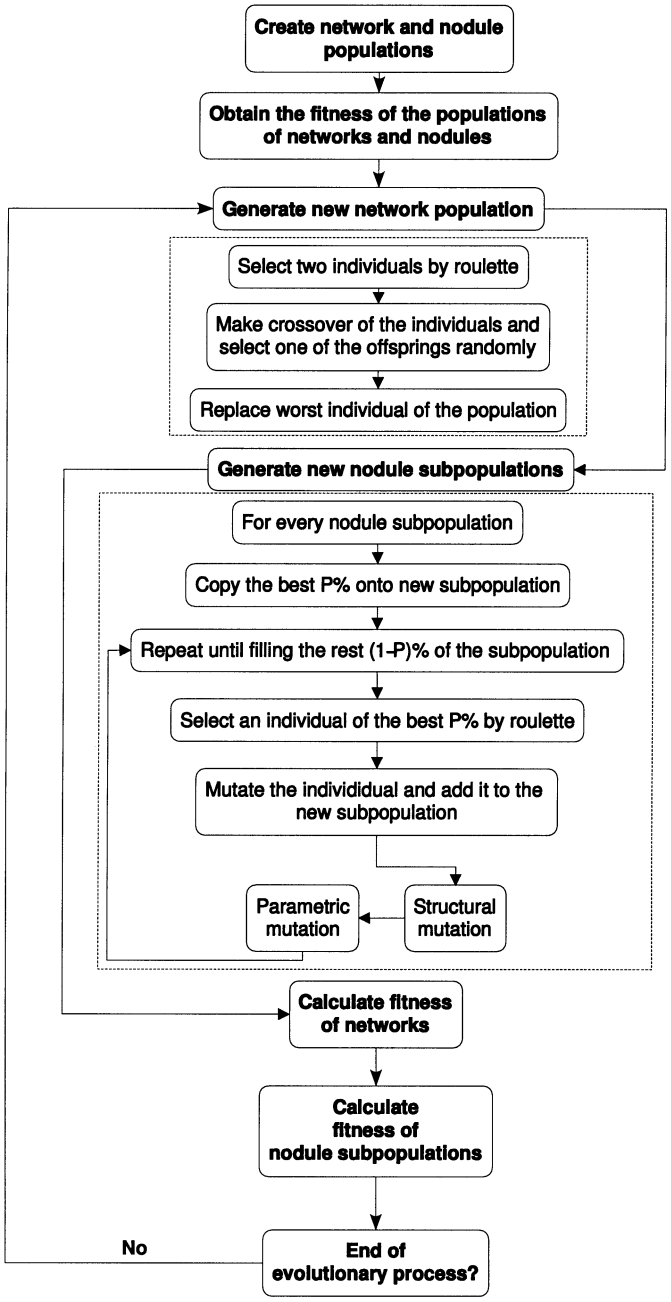

Fig. 3. Evolutionary process of both populations. The generation of a new population for both populations, networks and nodules, is represented in detail. The fitness of the individuals of the nodule subpopulations are evaluated in parallel, allowing the model to be run in a distributed system more efficiently.

- The new subpopulation is generated replicating the best $P\%$ of the former population. The remaining $(1-P)\%$ is removed and replaced by mutated copies of the best $P\%$. An individual of the best $P\%$ is selected by roulette selection and mutated. This mutated copy substitutes one of the worst $(1-P)\%$ individuals.
- There are two types of mutation: parametric and structural. The severity of the mutation is determined by the relative fitness, F_r, of the nodule. Given a nodule ν its relative fitness is defined as

$$F_r = e{-}\alpha F(\nu) \tag{5}$$

where $F(\nu)$ is the fitness value of nodule ν.

Parametric mutation consists of a local search algorithm in the space of weights, a simulated annealing algorithm [46]. This algorithm performs random steps in the space of weights. Each random step affects all the weights of the nodule. For every weight w_{ij} of the nodule the following operation is carried out:

$$w_{ij} = w_{ij} + \Delta w_{ij} \qquad \forall w_{ij} \in \nu \tag{6}$$

where

$$\Delta w_{ij} \in N(0, \beta F_r(\nu)) \tag{7}$$

where β is a positive value that must be set by the user in order to avoid large steps in the space of weights. The value of β used in all our experiments has been $\beta = 0.75$, anyway COVNET is quite robust regarding this parameter.

Then, the fitness of the nodule is recalculated and the usual simulated annealing criterion is applied. Being ΔF the difference in the fitness function before and after the random step.

- If $\Delta F \geq 0$ the step is accepted.
- If $\Delta F < 0$ then the step is accepted with a probability

$$P(\Delta F) = e^{-(\Delta F/T)}$$

where T is the current temperature. T starts at an initial value T_0 and it is updated at every step, $T(t+1) = \gamma T(t), 0 < \gamma < 1$. The number of steps of the algorithm that are carried out on each parametric mutation is very low. Performing many steps is computationally very expensive and the probability of being trapped on local minima would increase.

Parametric mutation is always carried out after structural mutation, as it does not modify the structure of the network.

Structural mutation is more complex because it implies a modification of the structure of the nodule. The behavioral link between parents and their offspring must be enforced to avoid generational gaps that produce inconsistency in the evolution. There are four different structural mutations:

Addition of a node. The node is added with no connections to enforce the behavioral link with its parent. As many authors have stated, [8], [17], maintaining the behavioral link between parents and their offsprings is of the utmost importance to get a useful algorithm.

Deletion of a node. A node is selected randomly and deleted together with its connections.

Addition of a connection. A connection is added, with weight 0, to a randomly selected node. There are three types of connection: from an input node, from another hidden node and to an output node. The selection of the type is made according to the relative number of each type of nodes: input, output, and hidden. Otherwise, when there is a significant difference among these three types the connections may end highly biased.

Deletion of a connection. A connection is selected, following the same criterion of the addition of connections and it is removed.

All the above mutations are made in the mutation operation on the nodule. For each mutation there is a minimum value, Δ_m and a maximum value, Δ_M. The number of elements (nodes or connections) involved in the mutation is calculated as follows:

$$\Delta = \Delta_m + F_r(\nu)(\Delta_M - \Delta_m). \tag{8}$$

TABLE I
PARAMETERS OF NODULE STRUCTURAL MUTATIONS COMMON TO ALL THE EXPERIMENTS CARRIED OUT

Mutation	Δ_m	Δ_M
Add node	0	1
Delete node	0	2
Add connection	1	4
Delete connection	0	3

So, before making a mutation the number of elements, Δ, is calculated, if $\Delta = 0$ the mutation is not actually carried out. The values of nodule mutation parameters used in all our experiments are shown on Table I.

There is no migration among the subpopulations. So, each subpopulation must develop different behaviors of their nodules, that is, different species of nodules, in order to compete with the other subpopulations for conquering its own niche and to cooperate to form networks with high fitness values.

B. Network Population

The network population is formed by a fixed number of networks. Each network is the combination of one nodule of each subpopulation of nodules. So the networks are strings of integer numbers of fixed length. The value of the numbers is not significant as they are just labels of the nodules. The relationship between the two populations can be seen in Fig. 4. It is important to note that, as the chromosome that represents the network is ordered, the permutation problem we have discussed cannot appear.

The network population is evolved using the *steady-state* genetic algorithm [47], [48]. This term may lead to confusion as it has been proved that shows higher variance [49] and is a more aggressive and selective selection strategy [50] than the standard genetic algorithm. This algorithm is selected because we need a population of networks that evolves more slowly than the population of nodules, as the changes in the population of networks have a major impact in the fitness of the nodules. The steady-state genetic algorithm avoids the negative effect that this drastic modification of the population of networks could have over the subpopulations of nodules. It has been also shown by some works in the area [51], [52] that the steady-state genetic algorithm produces better solutions and is faster than the standard genetic algorithm.

This algorithm has three features that are different from the standard genetic algorithm.

- The crossover generates just one individual. Two parents are chosen by means of a roulette selection algorithm. One of the two offsprings is selected randomly.
- The selected offspring replaces the worst individual of the population instead of replacing one of its parents.
- Fitness is assigned to the members of the population in function of their rank and not as their absolute fitness value. In our model, this feature has been ignored and the absolute value of the fitness has been used.

The algorithm allows adding mutation to the model, always at very low rates. Usually mutation rate ranges from 1% to 5%.

In our model we have modified this standard algorithm allowing the replacement of the n worst individuals instead of replacing just the worst one. In our experiments $n = 2$.

Crossover is made at nodule level, using a standard two-point crossover. So the parents exchange their nodules to generate their offspring. Mutation is also carried out at nodule level. When a network is mutated one of its nodules is selected and is substituted by another nodule of the same subpopulation selected by means of a roulette algorithm.

During the generation of the new nodule population some nodules of every population are removed and substituted. The removed nodules are also substituted in the networks. This substitution has two advantages: first, poor performing nodules are removed from the networks and substituted by potentially better ones and second, the new nodules have the opportunity to participate in the networks immediately after their creation.

C. Fitness Assignment

The assignment of fitness to networks is straightforward. Each network is assigned a fitness in function of its performance in solving a given problem. If the model is applied to classification, the fitness of each network is the number of patterns of the training set that are correctly classified; if it is applied to regression, the fitness is the sum of squared errors and so on. In the test problems below the classification is made following the criterion of the maximum:[4] the pattern is assigned to the class whose corresponding output is the highest one. Ties are resolved arbitrarily assigning the pattern to a *default class*.[5]

Assigning fitness to the nodules is a much more complex problem. In fact, the assignment of fitness to the individuals that form a solution in cooperative evolution is one of its key topics. The performance of the model highly depends on that assignment. A discussion of the matter can be found in the Introduction of [9].

Our credit assignment must fulfill the following requirements to be useful.

- It must enforce competition among the subpopulations to avoid two subpopulations developing similar responses to the same features of the data.
- It must enforce cooperation. The different subpopulations must develop complementary features that together could solve the problem.
- It must measure the contribution of a nodule to the fitness of the network and not only the performance of the networks where the nodule is present. A nodule in a good network must not get a high fitness if its contribution to the performance of the network is not significant. Likewise, a nodule in a poor performing network must not be penalized if its contribution to the fitness of the network is positive. Otherwise, a good nodule that is temporarily assigned to poor rated networks could be lost in the evolution of the subpopulations of nodules.

[4]This is similar to the Bayesian criterion in the problems of classification in two classes.

[5]Ties only happen during the first steps of evolution, as the networks evolve ties hardly or never occur.

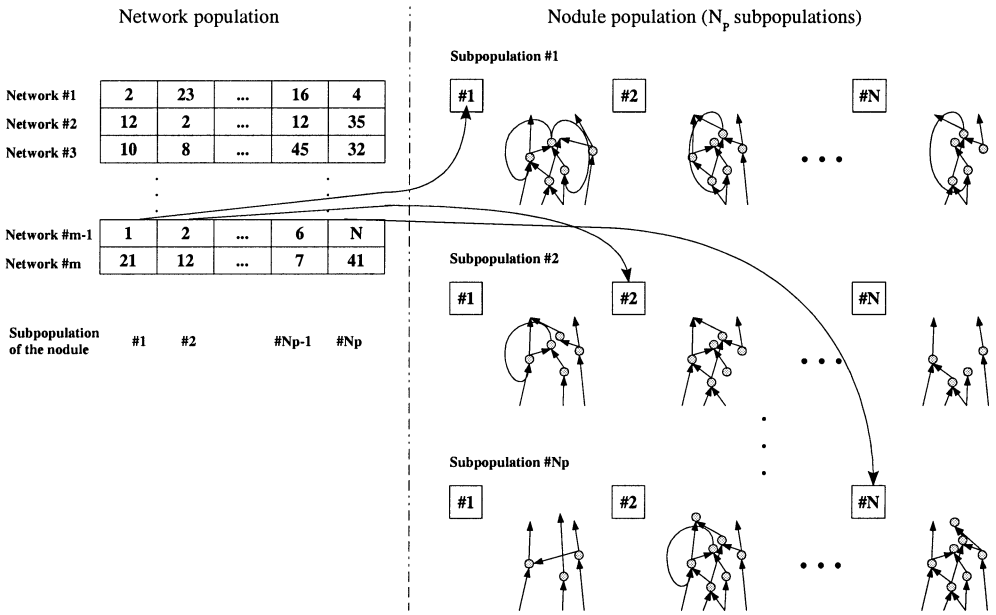

Fig. 4. Populations on networks and nodules. Each element of the network is a reference to, or a label of, an individual of the corresponding subpopulation of nodules. So the network is a vector where the first component refers to a nodule of subpopulation 1, the second component to a nodule of subpopulation 2 and so on.

Some methods for calculating the fitness of the nodules have been tried. The best one consists of the weighted sum of three different criteria. These criteria, for obtaining the fitness of a nodule ν in a subpopulation π, are as follows.

Substitution (σ). k networks are selected using an elitist method, that is, the best k networks of the population. In these networks the nodule of subpopulation π is substituted by the nodule ν. The fitness of the network with the nodule of the population π substituted by ν is measured. The fitness assigned to the nodule is the averaged difference in the fitness of the networks with the original nodule and with the nodule substituted by ν. This criterion enforces competition among nodules of the same subpopulation, as it tests if a nodule could achieve better performance than the rest of the nodules of its subpopulation.

The interdependencies among nodules could be a major drawback in the *substitution* criterion, but it does not mean that this criterion is useless. In any case, the criterion has two important features.

— It encourages the nodules to compete within the subpopulations, rewarding the nodules most *compatible* with the nodules of the rest of the subpopulation. This is true even for a distributed representation, because it has been shown that such representation is also modular. Moreover, as the nodules have no connection among them, they are more independent than in a standard network.

— As many of the nodules are competing with their parents, this criterion allows to measure if an offspring is able to improve the performance of its parent.

In addition, the neuropsychological evidence showing that certain parts of the brain consist of modules, that we discussed above, would support this objective.

Difference (δ). The nodule is removed from all the networks where it is present. The fitness is measured as the difference in performance of these networks. This criterion enforces competition among subpopulations of nodules preventing more than one subpopulation from developing the same behavior. If two subpopulations evolve in the same way, the value of this criterion in the fitness of their nodules will be near zero and the subpopulations will be penalized.

Best k (β_k). The fitness is the mean of the fitness values of the best k networks where the nodule ν is present. Only the best k are selected because the importance of the worst networks of the population must not be significant. This criterion rewards the nodules in the best networks and does not penalize a good nodule if it is in some poor performing networks.

Considered independently none of these criteria is able to fulfill the three desired features above mentioned. Nevertheless, when the weighted sum of all of them is used they have proved to give a good performance in the problems used as tests, as will be shown on Section IV. Typical values of the weights of the components of the fitness used in our experiment are ($\lambda_\delta \simeq 2\lambda_\sigma \simeq 60\lambda_{\beta_n}$). The values of these coefficients must not only weight the importance of each criteria but also correct the differences in range of them.

In order to encourage small nodules we have included a regularization term in the fitness of the nodule. Being n_n the number of nodes of the nodule and n_c the number of connections, the *effective fitness*,[6] f_i', of the nodule is calculated as follows:

$$f_i' = f_i - \rho_n n_n - \rho_c n_c. \tag{9}$$

The values of the coefficients must be in the interval $0 < \rho_n, \rho_c \ll 1$ in order to avoid the regularization term introducing a high bias in the learning process.

[6]It is called *efective fitness* because it is the actual value used as the fitness of the nodule in the generation of a new subpopulation.

So, the equation of the effective fitness of the nodule ν of subpopulation π is the following:

$$f_\nu^\pi = \lambda_\sigma \sigma + \lambda_\delta \delta + \lambda_{\beta_k} \beta_k - \rho_n n_n - \rho_c n_c \qquad (10)$$

if the expression above is negative for any of the nodules of a subpopulation, then the fitness values of all the nodules of that subpopulation are shifted, as we have mentioned above, as follows:

$$f_\nu^\pi = f_\nu^\pi - \min\{f_i^\pi\}_{i=1}^N \qquad (11)$$

where N is the number of nodules of the nodule subpopulation.

D. Stop Criterion

The stop criterion can have a dramatic impact on the performance of the model, as the over-training effect depends highly on it. Most models use cross-validation to avoid this effect. In COVNET we have evolved the population with and without a validation set. In both cases the evolution of the system depends only on the fitness of the network population. The system is evolved until *the average fitness of the network population stops growing*. The fitness is measured over the training set, unless a validation set is used, in this case the fitness is measured over this validation set.

When we stop the evolution of the population due to the stagnation of the average fitness of the networks, an individual with a higher fitness than the current best one could be reached if the evolution process is continued. But the generalization ability of such individual is likely to be worse in most cases. The same effect is likely to occur in the generalization ability of the whole population of networks during this additional stage of evolution.

E. Election of the Best Individual

The election of the best individual of the population is straightforward. The best network in terms of training error is chosen (or in terms of validation error if a validation set is used). In case of a tie the smallest network is preferred, if some of them are of equal size, one of them is chosen at random.

IV. Performance Evaluation

The performance of the developed model is tested in three classification problems with different features. In order to get a clear idea of the performance of the model we have compared our model with a modular network, the *adaptive mixture of local experts* [53]. Each *expert* is a multilayer perceptron (MLP) trained with standard backpropagation [54] and a momentum term.[7] We have also compared COVNET with the results in the bibliography.

For the design and training of the modular networks we have used the *NeuralWorks Professional II/Plus* [56] simulator. We also tried some pruning algorithms that are implemented in the *Stuttgart Neural Network Simulator* (SNNS)[8] (OBD [57], OBS [22], and Skeletonization [58]), but always with worse results.

[7]We tested also the performance of local experts using the EDBD rule [55] but the generalization results were worse.

[8]This package could be obtained by anonymous ftp from ftp://ftp.informatik.uni-stuttgart.de/pub/SNNS.

TABLE II
COVNET'S PARAMETERS COMMON TO ALL THE EXPERIMENTS

Parameter	*Value*
Number of networks	100
Number of nodules on each subpopulation	40
Networks to replace on each generation	2.0%
Mutation rate on network population	5.0%
Initial value of weights	(-0.5, 0.5)
Nodule elitism	70%
Input scaling interval	$[-2.5, 2.5]$
Number of nodule subpopulations	5
Initial maximum number of nodes	3
Initial maximum number of connections	15
Nodule fitness components	$\lambda_\sigma = 3.50$
	$\lambda_\delta = 1.45$
	$\lambda_{\beta_3} = 0.05$
Regularization term	$\rho_n = 0.25$
	$\rho_c = 0.025$
Simulated annealing	$T_o = 5.0$
	$\alpha = 0.95$
	$n = 25$
Minimum improvement (stop criterion)	10%

The design of the modular network was made with different architectures and learning algorithms. COVNET has been programmed in C under the Linux Operating System. All the tools and programs used for its development are licensed under the GNU General Public License. COVNET's code[9] is also under the GNU General Public License.

All the parameters of COVNET are common to all the data sets used in the experiments. Such parameters are shown in Table II. Setting the parameters for each problem specifically improves the performance of COVNET but using the same parameters for all the problems shows the robustness of the model regarding the parameter's setting.

The regularization term is either used with the parameters shown in Table II or is removed, setting the parameters to zero. This second option is used when no over-training effect is observed and the resulting networks are small enough for the purposes of a specific task.

As in any other evolutionary algorithm this set of parameters must be tuned in order to get a useful model. A study of the sensibility of COVNET to this set of parameters is beyond the scope of this paper. Nevertheless, we will give in this section

[9]Available by anonymous ftp from ftp://ftp.ayrna.org/pub/COVNET .

some guidance that could help the parameter's setting for any problem.

The parameters of the population, number of networks, number of nodule subpopulations and number of nodules per subpopulation, can have a variety of values. However, increasing the values shown in this paper will not improve the performance and will increase the computational cost of the evolution.

Elitism in the nodule population must be high. Values near 50% produce very good and very bad individuals, making the evolution unstable.

The weight of the nodule fitness subcomponents must be fixed in a way that corrects the differences among their ranges. The values given in our experiments follows this idea. In a specific problem could be interesting considering any of the subcomponent more important than the others, but that can only be tested by trial and error.

Regularization parameters must be set in function of the importance of parsimony in our task. Increasing the values shown in this paper will evolve smaller network, but also will decrease the performance of the networks as the regularization restriction becomes more critical.

A. Experimental Setup

The tests were conducted following the guidelines of Prechelt [59]. Each set of available data was divided into three sets: 50% of the patterns were used for learning, 25% of them for validation and the remaining 25% for testing the generalization of the individuals.

The populations of COVNET were evolved using together the training set and the validation set, that is, no validation was used. At the end of the evolution the best network, in terms of training error, was selected as the result of the evolution. The test set was then used to obtain the generalization of this network.

For the training of the modular networks we used the method of cross-validation and early-stopping [60]. The networks were trained until the error over the validation set started to grow. Nevertheless, the results obtained with early-stopping were worse that the ones obtained when the validation set was added to the training set. Only the results with the latter configuration are shown.

For each data set three different random permutations of the patterns were made. For each permutation the evolutionary process was repeated ten times. The fitness of the individuals of the population of networks was measured as the number of patterns correctly classified.

In all the tables we show, for each permutation of the data sets, the averaged error of classification over ten repetitions on each permutation of the data set, the standard deviation, the best and worst individuals and the averaged number of nodes and connections of the best networks of each experiment. The measure of the error is the following:

$$E = \frac{1}{P}\sum_{i=1}^{P} e_i \tag{12}$$

where P is the number of patterns and e_i is zero if pattern i is correctly classified and 1 otherwise. Each permutation of the data set is labeled I, II, and III in the tables. The averaged results over the three permutations are labeled All.

TABLE III
COMPARISON BETWEEN COVNET AND MLP NETWORKS IN TERMS OF AVERAGED NETWORK SIZES FOR THE THREE PROBLEMS

Data set	Model	#nodes	#connections
Pima	Modular	17	180
	COVNET	4.57	24.60
Heart disease	Modular	25	360
	COVNET	4.77	33.07
Credit card	Modular	14	750
	COVNET	3.67	34.23

TABLE IV
P-VALUES OF STATISTICAL TESTS ON PIMA INDIAN ERRORS

Test	Null hypothesis	COVNET	Modular
K-S	Normal distribution	0.4364	0.744
F	Equality of variances	0.2408	
t	Equality of means	1.091e-05	

B. Classification of the Pima Indian Data Set

This data set is from the UCI machine learning repository. The data set contains data of 768 individuals, all of them females at least 21 years old of Pima Indian heritage. The patterns are divided into two classes. The class of each pattern shows whether the patient shows signs of diabetes according to the World Health Organization criteria.

There are eight attributes for each pattern, all of them are real valued. Former results can be found in [61]. Following this previous work and the recommendations of Prechelt ([59], [62]) we have divided the data set in 384 patterns for training, 192 patterns for validation and 192 patterns for generalization. The data set contains 500 instances of class 1 and 268 instances of class 2. Three permutations of the data set are used and ten experiments are carried out on each one.

The modular network was formed by four local expert of three hidden nodes, the gating network had five hidden nodes. The error was measured using the *451 045 criterion*, that is, an output is 1 if it is the interval [0.55, 1.0] and it is 0 if it belongs to the interval [0.0, 0.45]. If the output is in the interval (0.45, 0.55) it is undefined.

The parameters for applying COVNET to this problem are shown in Table II. As we can see on Table V COVNET outperformed the modular network of adaptive mixture of experts.

The size of the networks evolved by COVNET was smaller than the size of the modular networks designed by hand. The number of nodes and connections of the different models is shown on Table III. Networks developed by COVNET are very compact and far smaller than the modular network needed to obtain comparative performance.

TABLE V
ERROR RATES FOR PIMA INDIAN DATASET

Model	Set	Training				Generalization			
		Mean	StD	Best	Worst	Mean	StD	Best	Worst
Modular network	I	0.2528	0.0135	0.2413	0.2847	0.2458	0.0211	0.2135	0.2865
	II	0.2460	0.0109	0.2326	0.2604	0.2370	0.0268	0.2083	0.2813
	III	0.2540	0.0124	0.2413	0.2795	0.2068	0.0181	0.1771	0.2448
	All	0.2509	0.0124			*0.2299*	0. 0274		
COVNET	I	0.2226	0.0073	0.2135	0.2344	0.1995	0.0284	0.1615	0.2448
	II	0.2269	0.0070	0.2170	0.2396	0.2063	0.0236	0.1771	0.2396
	III	0.2243	0.0042	0.2188	0.2326	0.1911	0.0082	0.1771	0.2031
	All	0.2246	0.0063			*0.1990*	0.0220		

TABLE VI
COMPARISON OF COVNET'S RESULTS WITH OTHER MODELS IN TERMS OF AVERAGED TESTING ERROR IN THE PIMA INDIAN DATASET

Model	Author	Error rate	Runs (#sets)	Network size
COVNET	-	0.1990	30(3)	4.8
BP	L. Prechelt[59]	0.2437	60(3)	32
Logdisc	D. Michie[67]	0.2230	12 (12)	N/A
EPNet	X. Yao *et al.*[8]	0.2237	30 (1)	3.4
EPNet ensemble	X. Yao *et al.*[31]	0.2220	30 (1)	86
acasper	N. K. Treadgold *et al.*[68]	0.2314	50 (1)	3.02
acascor	N. K. Treadgold *et al.*[68]	0.2453	50 (1)	9.78
MPyramid	R. Parekh *et al.*[16]	0.232	10 (10)	6.5
MTiling	R. Parekh *et al.*[16]	0.229	10 (10)	5.7

In order to verify the true difference in performance between COVNET and the modular network we conducted three statistical tests (a summary of the results is shown on Table IV). First, we corroborated that the distribution of the errors was normal by means of a Kolmogorov—Smyrnov test [63]. Next, we tested the hypothesis that the errors from the experiments with COVNET and modular networks had the same variance performing an F test [64]. Finally, we performed a t test that allowed us to ascertain that the error obtained with COVNET was significatively lower than the error obtained with the modular network with a significance level of 5%. All tests were made with the R statistical package [65].

Although direct comparison with other papers is difficult because the algorithms and methods of obtaining the generalization of the models are different, it is interesting to compare the results we have obtained with the latest published in the bibliography. The most important results for the diabetes problem are shown on Table VI. The results of COVNET are better than those obtained with the other methods, even taking into account that Michie *et al.* used for testing the error rate a 12-fold cross-validation and Parekh *et al.* a ten-fold cross-validation which are more optimistic methods than the one we have used. Yao and Liu achieved a testing error of 0.2237 with a population of networks evolved with their EPNet algorithm [8]. The evolved networks are slightly more compact than those evolved with COVNET, but their testing error is significatively worse. They achieved a better result, 0.2220 testing error, with an ensemble of 20 networks evolved with EPNet [31], but the complexity of the ensemble did not pay the improvement of the testing error. Prechelt [59] found a hand-designed neural network with an averaged testing error of 0.2437 which is worse than the worst network evolved by COVNET. More recent results with the algorithms MPyramid and MTiling [16], developed by Parekh *et al.*, have not been able to improve the results above. Their best model achieved a testing error of 0.229 with a complexity of the networks comparable to COVNET. Treadgold and Gedeon tried this data set with a modification of the cascade-correlation network architecture [66]. They achieved a

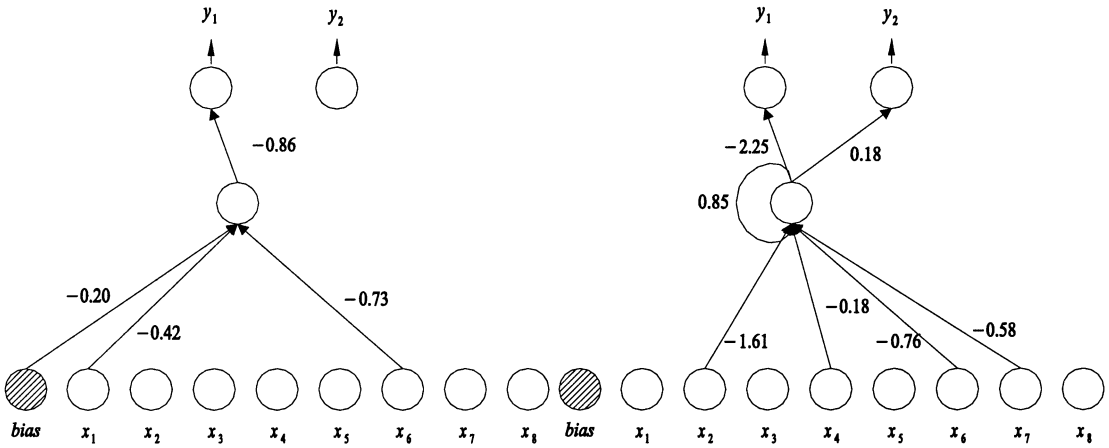

Fig. 5. Two nodules that constitute the best network evolved for Pima data set, it achieves a generalization error of 0.1719.

TABLE VII
ERROR RATES FOR HEART DISEASE DATASET

Model	Set	Training				Generalization			
		Mean	StD	Best	Worst	Mean	StD	Best	Worst
Modular network	I	0.0500	0.0135	0.0347	0.0743	0.1853	0.0210	0.1618	0.2206
	II	0.0490	0.0147	0.0347	0.0792	0.1794	0.0331	0.1324	0.2353
	III	0.0639	0.0118	0.0495	0.0842	0.2176	0.0248	0.1765	0.2500
	All	0.0543	0.0143			*0.1941*	0.0310		
COVNET	I	0.1257	0.0063	0.1139	0.1337	0.1500	0.0135	0.1324	0.1765
	II	0.1366	0.0142	0.1089	0.1584	0.1206	0.0217	0.0882	0.1471
	III	0.1312	0.0075	0.1238	0.1436	0.1574	0.0318	0.1176	0.2059
	All	0.1312	0.0106			*0.1426*	0.0279		

testing error of 0.2314 with 50 runs over one permutation of the data. Although the testing error is worse than the results of other models, including COVNET, the obtained networks are the smallest in all the bibliography.

To show a typical network evolved by COVNET, Fig. 5 represents the best network evolved in terms of testing error (it achieved a 0.1719 testing error). We can see that not only it has a low number of nodes but the connections are also sparse.

For the Pima data set, we can conclude that COVNET shows the best results in terms of generalization error rate with a complexity that is similar to the best results in the bibliography and significantly lower than that of modular neural networks.

C. Classification of Heart Disease Problem

This data set comes from the Cleveland Clinic Foundation and was supplied by Robert Detrano of the V.A. Medical Center, Long Beach, CA. The database contains 13 attributes, which have been extracted from a larger set of 75, that correspond to the results of various medical tests carried out on a patient. The goal is the prediction of the presence or absence of heart disease in those patients. The original data set had five classes, considering four degrees of heart disease. The database originally contained 303 examples but six of them had missing values and 27 of the remaining were retained in case of dispute, leaving a final total of 270 (the problem is described more deeply in [69]).

It is a very interesting data set because it has real valued attributes (1, 4, 5, 8, 10, and 12), ordered (11), binary valued (2, 6, and 9) and nominal (7, 3, and 13), making its classification more difficult; and the number of available patterns is small. There are two outputs determining whether the patient has a heart disease.

The results obtained with COVNET are shown on Table VII together with the results of a modular network made up by four local experts each one having five hidden nodes. The parameters used in COVNET's evolution are shown on Table II.

As in the former problem we have checked the bibliography in order to compare our results with other papers working with this dataset. The most important results are shown on Table VIII. Roy *et al.* used an RBF network of 24 Gaussians to achieve a testing error of 0.1818. They used the original data set with

TABLE VIII
COMPARISON OF COVNET'S RESULTS ON HEART DISEASE DATASET WITH OTHER MODELS IN TERMS OF AVERAGED TESTING ERROR

Model	Author	Error rate	Runs (#sets)	Network size
COVNET	-	0.1426	30(3)	7.97
MSM1	K. P. Bennet*et al.*[70]	0.1653	N/A	N/A
RBF (GM)	A. Roy*at al.*[72]	0.1818	30 (1)	30
EPNet	X. Yao *et al.*[8]	0.1677	30 (1)	4.1
EPNet ensemble	X. Yao *et al.*[31]	0.1510	30 (1)	94
acasper	N. K. Treadgold*et al.*[68]	0.1921	50 (1)	0.10
acascor	N. K. Treadgold*et al.*[68]	0.1989	50 (1)	2.64

297 cases (198 for training and 99 for testing the generalization of the network). The paper reported the average over 30 runs of the algorithm with the best configuration. Bennet and Mangasarian [70] reported a testing error of 0.1653 with their MSM1 method. Yao and Liu [8] achieved a testing error of 0.1677 with their EPNet evolutionary model. The experimental setup of their test is very similar to ours, except that they used just one permutation of the dataset. Later results [31] show an averaged testing error of 0.1510, obtained by an ensemble of 20 networks evolved with the EPNet algorithm and combined with the RLS algorithm [71]. Nevertheless, the whole evolutionary process of such networks is very complex and the resulting networks are quite big. Treadgold and Gedeon [68] reported a testing error of 0.1921 and 0.1989 with their algorithms *acasper* and *acascor* respectively. These algorithms are based on the cascade-correlation network of Fahlman and Lebiere [66] and they produced very small networks, despite the fact that their testing error was much higher than the method cited above and the results obtained by COVNET. The best hand-designed neural network [59] achieved 0.1478 testing error, which is slightly better than the averaged testing error of COVNET.

The same statistical tests performed over the results of Pima data have been carried out in the results of this problem. The results are shown on Table IX. The test shows, at a significance level $\alpha = 0.5$, that the there are significant differences in mean between the two samples of errors.

D. Credit Card Problem

This data set is also from the UCI Machine Learning Repository. The set contains data from applications to an Australian bank to get a credit card. There are two classes, meaning whether the application was granted (44.5% of the patterns) or denied (55.5%). Each record has 14 attributes, for confidentiality all attributes and values are not explained in the original data set.

This data set is very interesting because it has two important features. First, it contains many missing values (there are missing values in 5% of the records). Second, the attributes are of very different kind: continuous (1, 2, 10, 13, and 14), binary (0, 8, 9, and 11), and nominal (3, 4, 5, 6, and 12). The binary and nominal attributes have been codified using a 1-out-of-n code so the dataset used for training the networks had 51 inputs and two outputs.

TABLE IX
P-VALUES OF STATISTICAL TESTS ON HEART DISEASE ERRORS

Test	Null hypothesis	COVNET	Modular
K-S	Normal distribution	0.06898	0.3502
F	Equality of variances	0.5762	
t	Equality of means	7.493e-09	

The parameters of the evolution of COVNET are shown on Table II. The results obtained with COVNET and with a modular neural network of three experts each one with three hidden nodes are shown on Table X. The comparison in terms of network size is shown on Table III.

This dataset has been used in many other works (see Table XI). The best results are obtained by Yao and Liu [31] with an ensemble of networks evolved by the EPNet algorithm, obtaining a testing error of 0.093 over 30 runs of the algorithm. Our result is a little below this mark, however, the complexity of the ensemble of networks is much bigger than the comparatively very small networks obtained by COVNET. The ensemble is the combination of 20 networks with an average size of 4.5 nodes and 82.8 connections. Slightly poorer results (test set error of 0.095) are obtained with an ensemble of ten networks. In [67] several algorithms are used over this data set, the testing error being measured by means of ten-fold cross-validation. The best algorithm produced a test set error of 0.131. This value is worse than the error obtained by COVNET, even though ten-fold cross-validation is a more optimistic method in estimating errors than the method we used.

As in previous sets we have performed three statistical tests to assess that the results of COVNET are better than the ones obtained by a modular network. These tests are shown on Table XII.

TABLE X
ERROR RATES FOR CREDIT CARD DATASET

Model	Set	Training				Generalization			
		Mean	StD	Best	Worst	Mean	StD	Best	Worst
Modular network	I	0.1120	0.0097	0.0907	0.1236	0.1401	0.0043	0.1337	0.1453
	II	0.1371	0.0356	0.1178	0.2355	0.1395	0.0131	0.1279	0.1686
	III	0.1237	0.0160	0.1042	0.1602	0.1326	0.0116	0.1163	0.1512
	All	0.1243	0.0247			*0.1374*	0.0106		
COVNET	I	0.1375	0.0075	0.1255	0.1506	0.1198	0.0088	0.1105	0.1395
	II	0.1431	0.0092	0.1236	0.1564	0.1105	0.0119	0.0930	0.1337
	III	0.1411	0.0059	0.1313	0.1486	0.1169	0.0089	0.1047	0.1337
	All	0.1405	0.0078			*0.1157*	0.0104		

TABLE XI
COMPARISON OF COVNET'S RESULTS ON CREDIT CARD DATASET WITH OTHER MODELS IN TERMS OF AVERAGED TESTING ERROR

Model	Author	Error rate	Runs (#sets)	Network size
COVNET	-	0.1157	30(3)	2.47
Cal5	D. Michie et al.[67]	0.131	10 (10)	N/A
EPNet	X. Yao et al.[8]	0.115	30 (1)	4.83
EPNet ensemble	X. Yao et al.[31]	0.093	30 (1)	90
acasper	N. K. Treadgold *et al.[68]*	0.1372	50 (1)	0.12
acascor	N. K. Treadgold *et al.[68]*	0.1358	50 (1)	1.07

TABLE XII
P-VALUES OF STATISTICAL TESTS ON CREDIT CARD ERRORS

Test	Null hypothesis	COVNET	Modular
K-S	Normal distribution	0.5624	0.6407
F	Equality of variances	0.9019	
t	Equality of means	6.278e-11	

V. ANALYSIS OF COVNET

A. Capability and Necessity Tests

The key idea of our model is the combination of subnetworks to form better solutions than the solution that could be achieved by just the evolution of a whole network. In order to test this idea we repeat our experiments over the data sets described above with networks formed by only one nodule. The results are shown on Table XIII.

The results obtained prove that the combination of several modules forming an ensemble clearly improves the performance of the networks. To compare the results with 1 and 5 nodules we have carried out the statistical test whose results are shown on Table XIV.

We can also observe that the evolution with the combination of several nodules has considerably lower variance of the testing error, being this feature very important, as it means a more robust evolutionary process.

An additional capability test was carried out over the experiments performed on the datasets. For the best individual of the final population, we chose every nodule in turn and measured its performance over the learning and generalization sets. This gave an idea of the capability of the evolved nodules. The results are shown on Table XV. This test has been carried out only in the first partition of the three problems. The nodules that evolved to void nodules are marked with a "–. " The table shows that the error value of the best nodule of each network is clearly worse than the error of the network. That enforces the idea of the combination of parts, as the nodules cannot achieve the performance of the networks (that are combination of them) whether they are evolved separately or combined.

The necessity test is made in order to measure the individual importance of each single nodule in the evolved networks. This

TABLE XIII
TESTING ERROR RATES FOR NETWORKS WITH ONE NODULE FOR ALL THE THREE PROBLEMS

Problem	Set	Training				Generalization			
		Mean	StD	Best	Worst	Mean	StD	Best	Worst
Pima	I	0.2564	0.0349	0.2274	0.3247	0.2479	0.0312	0.2135	0.3177
	II	0.2712	0.0286	0.2396	0.3194	0.2510	0.0581	0.1927	0.3281
	III	0.2724	0.0377	0.2274	0.3403	0.2438	0.0369	0.1927	0.3073
	All	0.2670	0.0336			*2476*	0.0422		
Heart	I	0.1772	0.0188	0.1386	0.1980	0.2000	0.0242	0.1618	0.2353
	II	0.2109	0.0254	0.1733	0.2475	0.1765	0.0563	0.1029	0.2794
	III	0.1817	0.0239	0.1485	0.2228	0.1941	0.0397	0.1324	0.2500
	All	0.1899	0.0268			*1902*	0.0419		
Card	I	0.1680	0.0453	0.1351	0.2761	0.1477	0.0625	0.1105	0.3081
	II	0.1544	0.0566	0.1448	0.1911	0.1180	0.0167	0.1047	0.1512
	III	0.2164	0.0566	0.1274	0.2703	0.2186	0.0675	0.1221	0.2907
	All	0.1706	0.0492			*0.1614*	0.0675		

TABLE XIV
p-VALUES OF STATISTICAL TESTS COMPARING THE RESULTS WITH ONE NODULE AND n NODULES IN THE TEST SETS FOR ALL THE THREE PROBLEMS

Set	Test	Null hypothesis	#Nodules	
			1	5
Pima	K-S	Normal distribution	0.4595	0.4364
	F	Equality of variances	0.0007559	
	t	Equality of means	1.351e-06	
Heart	K-S	Normal distribution	0.8423	0.06898
	F	Equality of variances	0.03191	
	t	Equality of means	4.042e-06	
Credit card	K-S	Normal distribution	0.03427	0.5624
	F	Equality of variances	–	
	t	Equality of means	–	

test has been carried out only in the first partition of the three problems. In order to test this aspect, we removed in turn every nodule from the best individual of the final population. Then, we measured the error of the network without this nodule. The results are shown on Table XVI. The nodules that evolved to void nodules are marked with a "–." The table shows that removing a nodule does not cause a dramatic effect on the performance of the network, so the model is quite robust to the elimination of some parts of the network. This is very interesting if we consider the hardware implementation of the networks. The tables also shows the cooperation among the nodules, as the majority of nodules that did not evolve to void nodules have their share in the performance of the network. This aspect is less evident in the Card problem. For this problem the evolutionary process obtains useless nodules more frequently.

B. Overtraining Effect

In our experiments we have not used validation sets, either for early stopping of the evolutionary process or for choosing the best individual of the population (both these uses are common in [31]).

Not using a validation set for early stopping can produce over-training and bad generalization. For testing the possibility of appearance, we kept track of the generalization of the networks along the evolutionary process. Fig. 6 represents the behavior of the error of the population along the evolution on nine experiments over the Pima Indian dataset (the first three experiments carried out over each permutation of the dataset). The figure represents the averaged training error of the population, that is, the criterion for stopping the evolution; the generalization error of the best individual of the population in terms of training error, that is, how well generalizes the individual that learns best; and the averaged generalization error of the whole population.

From Fig. 6, we can see that the averaged testing error of the population hardly suffered from over-training, being this measure the more interesting to measure overtraining, as the generalization ability of the best individual is subject to important changes along the evolution. The averaged testing error remained constant during the last generations but it did not increase. The best individual did suffer from over-training. The figure shows how the testing error of the first individual decreased during, approximately, 40 generations and then started to grow slowly until the end of the evolutionary process. This

TABLE XV
CAPABILITY TEST FOR PIMA, HEART, AND CARD PROBLEMS. THE VALUES SHOW THE GENERALIZATION ERROR OF EACH NODULE OF THE BEST INDIVIDUAL OF THE POPULATION WHEN IT IS CONSIDERED ITSELF AS A NETWORK. THE LAST COLUMN SHOWS THE GENERALIZATION ERROR OF THE WHOLE NETWORK

Pima

Run	Nodule number 1	2	3	4	5	Network
1	0.2917	–	–	–	*0.2292*	0.1719
2	0.2500	–	*0.1927*	–	–	0.1719
3	–	0.2969	*0.2135*	–	0.2708	0.1823
4	*0.2240*	0.2448	0.2604	–	–	0.2031
5	–	*0.2552*	0.2708	0.2865	–	0.2240
6	0.7031	*0.2448*	0.2969	–	–	0.2448
7	0.7031	*0.2240*	–	–	–	0.2240
8	–	–	–	*0.1979*	0.3125	0.1615
9	–	–	–	0.3438	*0.1562*	0.1875
10	*0.2552*	0.2969	–	0.3125	–	0.2240

Heart

Run	Nodule number 1	2	3	4	5	Network
1	0.4412	–	–	0.2059	*0.1765*	0.1471
2	*0.1618*	–	–	0.2794	–	0.1618
3	–	*0.1912*	–	0.3382	0.4412	0.1324
4	–	*0.1765*	0.1912	–	0.2500	0.1324
5	*0.1765*	–	0.1912	–	–	0.1471
6	0.2059	0.2353	–	–	*0.1618*	0.1471
7	*0.1765*	–	–	–	0.3088	0.1618
8	–	–	–	*0.1912*	0.1471	0.1765
9	–	*0.1618*	0.3088	0.1765	–	0.1471
10	0.2206	–	*0.1765*	–	0.4412	0.1471

Card

Run	Nodule number 1	2	3	4	5	Network
1	0.5174	0.5174	*0.1105*	0.5174	0.2674	0.1221
2	–	0.5174	–	0.2384	*0.1221*	0.1221
3	*0.1221*	0.5174	0.5174	0.5174	–	0.1163
4	*0.1105*	0.2151	0.5174	0.4826	0.5174	0.1163
5	0.5174	0.5174	*0.1105*	0.4826	0.4477	0.1279
6	0.7674	0.5174	–	0.4826	*0.1105*	0.1105
7	*0.1163*	–	–	–	0.5174	0.1163
8	–	*0.1337*	–	0.4884	0.5174	0.1395
9	0.5174	0.4826	0.5174	*0.1512*	0.1337	0.1105
10	0.2733	0.1279	*0.1105*	0.5174	–	0.1163

TABLE XVI
NECESSITY TEST FOR PIMA, HEART, AND CARD PROBLEMS. THE VALUES SHOW THE INCREMENT ON THE GENERALIZATION ERROR OF THE NETWORK WHEN THE CORRESPONDING NODULE IS REMOVED. THE LAST COLUMN SHOWS THE GENERALIZATION ERROR OF THE WHOLE NETWORK

Pima

Run	Nodule number 1	2	3	4	5	Network
1	0.0573	–	–	–	*0.1198*	0.1719
2	0.0208	–	*0.0781*	–	–	0.1719
3	–	0.0156	*0.0990*	–	0.0260	0.1823
4	0.0052	*0.0208*	0.0052	–	–	0.2031
5	–	0.0104	*0.0677*	-0.0417	–	0.2240
6	0.0000	*0.0521*	0.0000	–	–	0.2448
7	0.0000	*0.4792*	–	–	–	0.2240
8	–	–	–	*0.1510*	0.0365	0.1615
9	–	–	–	-0.0312	*0.1562*	0.1875
10	*0.0885*	0.0104	–	0.0312	–	0.2240

Heart

Run	Nodule number 1	2	3	4	5	Network
1	0.0000	–	–	0.0294	*0.0588*	0.1471
2	*0.1176*	–	–	0.0000	–	0.1618
3	–	*0.2059*	–	0.0588	0.0000	0.1324
4	–	-0.0147	*0.0294*	–	0.0000	0.1324
5	*0.0147*	–	0.0000	–	–	0.1471
6	*0.0147*	-0.0147	–	–	-0.0147	0.1471
7	*0.1029*	–	–	–	-0.0294	0.1618
8	–	–	–	-0.0147	*0.0294*	0.1765
9	–	*0.0294*	0.0147	0.0147	–	0.1471
10	0.0147	–	*0.0588*	–	0.0000	0.1471

Card

Run	Nodule number 1	2	3	4	5	Network
1	0.0000	0.0000	*0.1337*	0.0000	-0.0116	0.1221
2	–	0.0000	–	0.0000	*0.1163*	0.1221
3	*0.4012*	0.0058	0.0000	0.0000	–	0.1163
4	*0.1105*	-0.0058	0.0000	0.0000	0.0000	0.1163
5	0.0000	0.0000	*0.3547*	-0.0174	0.0058	0.1279
6	0.0000	0.0000	–	0.0000	*0.6802*	0.1105
7	*0.4012*	–	–	–	0.0000	0.1163
8	–	*0.3488*	–	-0.0058	0.0000	0.1395
9	0.0000	0.0000	0.0000	*0.0233*	0.0174	0.1105
10	-0.0058	0.0000	*0.0233*	0.0000	–	0.1163

effect could not be removed using a validation set. The results with a validation set were far poorer than the results without it. However, the effect of over-training is not dramatic, as the testing error does not increase more than 3%–4%.

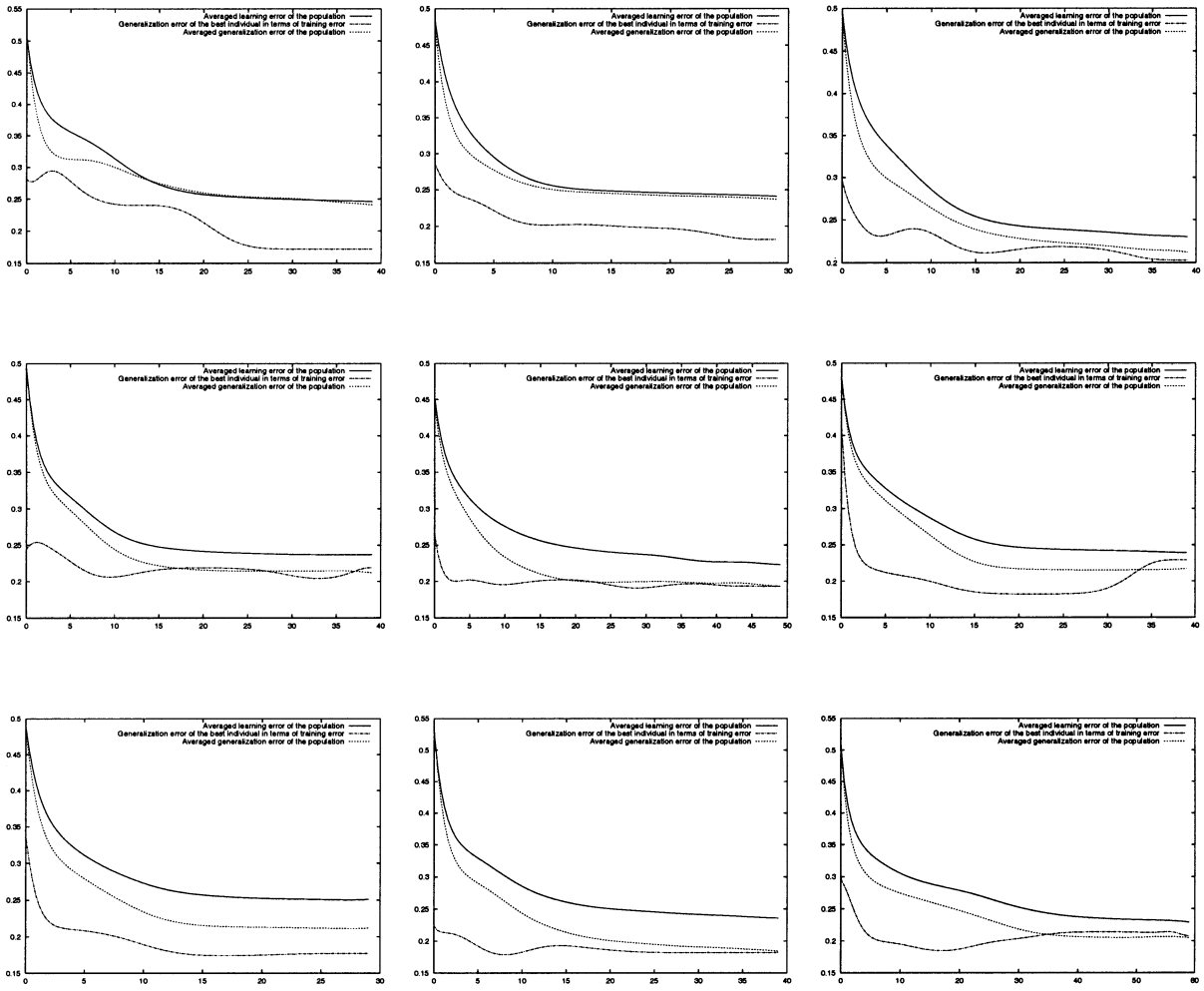

Fig. 6. Training and testing error of the population of networks along the evolutionary process in the classification of Pima Indians dataset. For each partition the first three experiments are shown.

TABLE XVII
SENSIBILITY ANALYSIS FOR PIMA INDIAN DATASET

Case	Training				Generalization				t-test
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	
-	0.2246	0.0063	0.2135	0.2396	*0.1990*	0.0220	0.1615	0.2448	-
$\sigma = 0$	0.2249	0.0057	0.2118	0.2396	*0.2082*	0.0146	0.1875	0.2500	0.0618
$\delta = 0$	0.2146	0.0067	0.2049	0.2361	*0.2036*	0.0169	0.1771	0.2396	0.3599
$\beta = 0$	0.2083	0.0052	0.1979	0.2188	*0.2127*	0.0184	0.1771	0.2500	0.0112
$\delta = \beta = 0$	0.2086	0.0072	0.1910	0.2205	*0.2052*	0.0193	0.1771	0.2448	0.2476
$\sigma = \beta = 0$	0.2212	0.0046	0.2118	0.2326	*0.2094*	0.0163	0.1823	0.2552	0.0419
$\sigma = \delta = 0$	0.2508	0.0112	0.2309	0.2708	*0.2323*	0.0298	0.1823	0.3073	0.0000
$\rho_n = \rho_c = 0$	0.2128	0.0051	0.2014	0.2240	*0.2069*	0.0181	0.1719	0.2396	0.1308

C. Analyzing the Relevance of Criteria

One of the most important aspect of our model is the evaluation of the fitness of the subcomponents using a three criteria method. It is interesting to test if the proposed criteria contribute to the performance of the model in a positive way. In order to test the relevance of each criterion we have performed the following experiment: for every criterion we have evaluated the performance of the model without using

TABLE XVIII
SENSIBILITY ANALYSIS FOR HEART DATASET

Case	Training				Generalization				t-test
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	
-	0.1312	0.0106	0.1089	0.1584	*0.1426*	0.0279	0.0882	0.2059	-
$\sigma = 0$	0.1381	0.0101	0.1188	0.1584	*0.1529*	0.0320	0.0882	0.2206	0.1896
$\delta = 0$	0.1173	0.0093	0.0990	0.1436	*0.1632*	0.0287	0.1324	0.2500	0.0065
$\beta = 0$	0.1168	0.0116	0.0941	0.1485	*0.1588*	0.0322	0.1029	0.2353	0.0418
$\delta = \beta = 0$	0.1219	0.0098	0.0990	0.1386	*0.1574*	0.0292	0.1029	0.2206	0.0508
$\sigma = \beta = 0$	0.1417	0.0099	0.1287	0.1782	*0.1564*	0.0249	0.1029	0.2059	0.0489
$\sigma = \delta = 0$	0.2360	0.0203	0.1931	0.2871	*0.2304*	0.0441	0.1176	0.3235	0.0000
$\rho_n = \rho_c = 0$	0.1249	0.0109	0.1089	0.1485	*0.1608*	0.0296	0.1029	0.2353	0.0177

TABLE XIX
SENSIBILITY ANALYSIS FOR CARD DATASET

Case	Training				Generalization				t-test
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	
-	0.1405	0.0078	0.1236	0.1564	*0.1157*	0.0104	0.0930	0.1395	-
$\sigma = 0$	0.1432	0.0130	0.1255	0.1988	*0.1254*	0.0250	0.1047	0.2442	0.0573
$\delta = 0$	0.1305	0.0102	0.1042	0.1486	*0.1207*	0.0179	0.0988	0.1802	0.1885
$\beta = 0$	0.1215	0.0099	0.1004	0.1429	*0.1260*	0.0148	0.0988	0.1512	0.0029
$\delta = \beta = 0$	0.1283	0.0095	0.1062	0.1448	*0.1200*	0.0145	0.0930	0.1453	0.1973
$\sigma = \beta = 0$	0.1358	0.0077	0.1236	0.1564	*0.1231*	0.0126	0.0988	0.1453	0.0165
$\sigma = \delta = 0$	0.1661	0.0286	0.1429	0.1429	*0.1376*	0.0448	0.0930	0.2907	0.0137
$\rho_n = \rho_c = 0$	0.1273	0.0102	0.1100	0.1506	*0.1196*	0.0115	0.0930	0.1337	0.1773

TABLE XX
NETWORK SIZES FOR THE THREE PROBLEMS WITHOUT USING THE REGULARIZATION TERM

Data set	#nodes	#connections
Pima	6.77	41.47
Heart disease	5.57	33.07
Credit card	4.73	44.63

the criterion and with the criterion alone. The objective is to test if the criterion is useful to the system and test its capability if it is considered as the only one. The results for Pima, Heart, and Card problems are summarized in Tables XVII, XVIII, and XIX, respectively.

From these results we can obtain very interesting information about the three criteria. First, we can see that all the problems are solved better using the three criteria, while all the experiments removing any of the criteria performed worse. However, the difference in performance is not always statistically significant.

Second, for each criterion we can conclude the following remarks:

Substitution. Removing this criterion has the effect of decreasing the performance of the model in two out of the three problems with a confidence level of 10%. Considered isolated, its performance is quite bad for two out of the three problems. The main reason is that this criterion only encourages competition among the nodules in the same subpopulation and when it is considered isolated the cooperation is assured only by the network population, so the performance of the model is seriously affected.

Difference. It is the criterion that performs best in isolation. The reason is that this criterion enforces both competition among the members of the same subpopulation and cooperation among the members of different subpopulations.

Best n. There is a very interesting result for this criterion. The performance of the criterion when considered alone is very low,

TABLE XXI
p-VALUES OF ANOVA ANALYSIS FOR PIMA DATASET. THE RESULTS OF THE TEST THAT MEASURES THE EFFECT OF THE COEFFICIENT ON THE GENERALIZATION ERROR AND THE RESULTS OF THE MULTIPLE COMPARISON OF THE THREE LEVELS OF EACH COEFFICIENT. THE NULL HYPOTHESIS IS THAT THERE IS NO EFFECT ON THE GENERALIZATION ERROR

Coefficient		Substitution				Difference				Best n		
Effect on error		0.901				0.789				0.883		
		3.15	3.50	3.85		1.30	1.45	1.60		0.04	0.05	0.06
	3.15	−−	1.000	1.000	1.30	−−	1.000	1.000	0.04	−−	1.000	1.000
	3.50	1.000	−−	1.000	1.45	1.000	−−	1.000	0.05	1.000	−−	1.000
Multiple comparison	3.85	1.000	1.000	−−	1.60	1.000	1.000	−−	0.06	1.000	1.000	−−

TABLE XXII
p-VALUES OF ANOVA ANALYSIS FOR HEART DATASET. THE THE RESULTS OF THE TEST THAT MEASURES THE EFFECT OF THE COEFFICIENT ON THE GENERALIZATION ERROR AND THE RESULTS OF THE MULTIPLE COMPARISON OF THE THREE LEVELS OF EACH COEFFICIENT

Coefficient		Substitution				Difference				Best n		
Effect on error		0.335				0.083				0.952		
		3.15	3.50	3.85		1.30	1.45	1.60		0.04	0.05	0.06
	3.15	−−	0.417	1.000	1.30	−−	1.000	0.109	0.04	−−	1.000	1.000
	3.50	0.417	−−	1.000	1.45	1.000	−−	0.262	0.05	1.000	−−	1.000
Multiple comparison	3.85	1.000	1.000	−−	1.60	0.109	0.262	−−	0.06	1.000	1.000	−−

with a generalization error far worse than the obtained by the other two criteria. This is specially important, as it is the only criterion commonly used for evaluating the subcomponents in a cooperative environment.

It is also relevant the dramatic impact of removing this criterion on the performance of the model. The reason is not only in the contribution of the criterion to the fitness of the nodule, but also in a very useful *side-effect* of this criterion. The nodules of the networks with the highest fitness have a high value in this criterion and its fitness value is also very high, so its probability of surviving along the evolutionary process is increased. In this way, the probability of removing a nodule of a high-performing network decreases, allowing the best networks to survive and mate.

Finally, we have made a test of the relevance of the regularization term over the size of the network and its generalization ability. We can see on the last experiment of the previous tables that the generalization error increases and the training error decreases. This is a typical effect of the absence of a regularization term. The evolved networks are also bigger as it is shown on Table XX (the size of networks with the regularization term were shown on Table III).

D. Analysis of Sensibility to Criteria Coefficients

Finally, we conclude the study of the different aspects of the model testing the sensibility of the evolution to a modification of the coefficients of the criteria. For each criterion we have defined an interval of variation of 20% around the value used in the experiments. We have taken the bounds and center of the interval, considering for each criterion the following values: $\sigma = \{3.15, 3.50, 3.85\}$, $\delta = \{1.30, 1.45, 1.60\}$, $\beta = \{0.04, 0.05, 0.06\}$. We have evaluated the performance of the model for the three problems with the 27 combinations of these three values of the coefficients. With the results of these experiments we have performed an ANOVA 3 analysis in order to measure the sensibility of the model to the coefficient variations and an ANOVA 1 analysis to test the influence of the variations of each coefficient separately. All the tests were made using the SPSS statistical package [73].

The results of the ANOVA analysis for the problems Pima, Heart, and Card are shown on Table XXI, Table XXII, and Table XXIII, respectively. These tables show a summary of the results of the analysis. First, they show the p-values of the test that measures the influence of the variations of the coefficients over the generalization error. The null hypothesis is that the generalization error does not depend on the value of the coefficient within the selected bounds. Second, they show the p-values of the test that measures whether there are significant differences in the generalization error when each one of the three possible values are selected for each coefficient. The null hypothesis is that the error mean is the same.

For the three problems we can see that the influence of the values of the coefficients within this interval of 20% over the generalization error is almost negligible. That shows the robustness of the model to moderate variations of the coefficients of the criteria.

TABLE XXIII
p-VALUES OF ANOVA ANALYSIS FOR CARD DATASET. THE THE RESULTS OF THE TEST THAT MEASURES THE EFFECT OF THE COEFFICIENT ON THE GENERALIZATION ERROR AND THE RESULTS OF THE MULTIPLE COMPARISON OF THE THREE LEVELS OF EACH COEFFICIENT

Coefficient	Substitution			Difference			Best n		
Effect on error	0.486			0.683			0.206		
	3.15	3.50	3.85	1.30	1.45	1.60	0.04	0.05	0.06
3.15	−−	0.705	0.600	1.30	−−	0.766	0.971	0.04	−−	0.331	0.350
3.50	0.705	−−	0.998	1.45	0.766	−−	0.948	0.05	0.331	−−	1.000
Multiple comparison 3.85	0.600	0.998	−−	1.60	0.971	0.948	−−	0.06	0.350	1.000	−−

VI. CONCLUSION AND FUTURE WORK

We have developed a cooperative coevolutionary model for the design of artificial neural networks. This model is based on the coevolution of several species of subnetworks (called nodules in our model) that must cooperate to form networks for solving a given problem. Instead of trying to evolve whole networks, a task that is not feasible in many problems or ends up with poorly performing neural networks, we evolve these subnetworks that must cooperate in solving the given task. The nodules coevolve in several independent subpopulations that evolve to different species. A population of networks that is evolved by means of a steady-state genetic algorithm keeps track of the best combinations of nodules for solving the problem.

We have also developed a new method for assigning credit to the individuals of the different species that cooperate to form a network. This method is based on the combination of three criteria. The criteria enforce competition within species and cooperation among species. The same idea underlying this method could be applied to other models of cooperative coevolution.

This model has proved to perform better than standard algorithms in two real problems of classification. Moreover, it has shown better results than the methods of training modular neural networks by means of gradient descent, e.g., the backpropagation learning rule and it has achieved better results for two of the three tested problems than the results reported in the bibliography and comparable results in the other problem and with less complexity than most models.

Networks evolved by COVNET are very compact and have few sparsely distributed connections. These networks are appropriate for hardware implementation. Moreover, the robustness to the damage of some parts of the network (i.e., the removal of a whole nodule as we have shown on Section V-A) is also a very interesting feature for hardware implemented neural networks.

COVNET is intended for environments where the time for evolution is not the critical feature, as in real-time problems. Searching in a space with very few restrictions allows more solutions to be explored. But also more time is spent on the evolutionary process.

Our current work on COVNET focus on two problems: the improvement of the credit assignment to nodules and number of subpopulation (species) to coevolve. The credit assignment to nodules must be modified to achieve a better fulfillment of the objectives stated above. We have observed that sometimes the cooperation of the nodules is not as good as desired. In those cases, each nodule tries to solve the problem by itself. The result is a poor combination of nodules and networks with low performance.

For this problem we have also worked in a different point of view that is considering the assignment of fitness to the nodules a multiobjective problem [74], [75]. The optimization of each criterion would be approached by a multiobjective evolutionary algorithm [76]. Each one of the three criteria discussed above together with a regularization term could be seen as different objectives for optimization.

As for the second major problem to address, the number of subpopulations (species), in our model this number is fixed and must be discovered through experimentation by the researcher. It would be interesting that this number were adaptive along the coevolution. Preliminary experiments have been carried out reinitializing the subpopulations that are not able to develop useful nodules, but with poor results.

It would be also very interesting to carry out a fine tuning of the parameters of the model using an evolutionary strategy. Currently, it is not feasible because of the enormous time that such experiment would take.

Finally, as in this paper we have experimentally proved the validity of our model in classification problems, the obvious next step will be the application of COVNET to regression. As the features of regression are very different from classification, the application of COVNET to such problems could be considered an almost independent task.

ACKNOWLEDGMENT

The authors would like to acknowledge Dr. F. Herrera-Triguero, R. Moya-Sánchez, and E. Sanz-Tapia for helping with the final version of this paper.

REFERENCES

[1] S. Haykin, *Neural Networks—A Comprehensive Foundation*. New York: Macmillan, 1994.
[2] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, pp. 1423–1447, Sept. 1999.
[3] Y. Shang and B. W. Wah, "Global optimization for neural networks training," *IEEE Comput.*, vol. 29, pp. 45–54, Mar. 1996.
[4] S.-B. Cho and K. Shimohara, "Evolutionary learning of modular neural networks with genetic programming," *Appl. Intell.*, vol. 9, pp. 191–200, 1998.
[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[6] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1994.

[7] T. Caelli, L. Guan, and W. Wen, "Modularity in neural computing," *Proc. IEEE*, vol. 87, pp. 1497–1518, Sept. 1999.

[8] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Networks*, vol. 8, pp. 694–713, May 1997.

[9] M. A. Potter, "The design and analysis of a computational model of cooperative coevolution," Ph.D. dissertation, George Mason Univ., Fairfax, VA, 1997.

[10] M. A. Potter and K. A. de Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, no. 1, pp. 1–29, 2000.

[11] N. García-Pedrajas, C. Hervás-Martínez, and J. Muñoz-Pérez, "Symbiont: A cooperative evolutionary model for evolving artificial neural networks for classification," in *Proc. 8th Int. Conf. Information Processing Management Uncertainty Knowledge Based Systems*, Madrid, Spain, July 2000, pp. 298–305.

[12] K. Chellapilla and D. B. Fogel, "Evolving neural networks to play checkers without relying on expert knowledge," *IEEE Trans. Neural Networks*, vol. 10, pp. 1382–1391, Nov. 1999.

[13] C.-T. Lin and C.-P. Jou, "Controlling chaos by GA-based reinforcement learning neural network," *IEEE Trans. Neural Networks*, vol. 10, pp. 846–859, July 1999.

[14] S. Gallant, *Neural-Network Learning and Expert Systems*. Cambridge, MA: MIT Press, 1993.

[15] V. Honavar and V. L. Uhr, "Generative learning structures for generalized connectionist networks," *Inform. Sci.*, vol. 70, no. 1/2, pp. 75–108, 1993.

[16] R. Parekh, J. Yang, and V. Honavar, "Constructive neural-network learning algorithms for pattern classification," *IEEE Trans. Neural Networks*, vol. 11, pp. 436–450, Mar. 2000.

[17] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 54–65, Jan. 1994.

[18] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, July 1993.

[19] J. Depenau and M. Moller, "Aspects of generalization and pruning," in *Proc. World Congr. Neural Networks*, vol. III, 1994, pp. 504–509.

[20] H. H. Thodberg, "Improving generalization of neural networks through pruning," *Int. J. Neural Syst.*, vol. 1, no. 4, pp. 317–326, 1991.

[21] Y. Hirose, K. Yamashita, and S. Hijiya, "Backpropagation algorithm which varies the number of hidden units," *Neural Networks*, vol. 4, pp. 61–66, 1991.

[22] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," *Advances in Neural Information Systems 5*, 1993.

[23] R. Kamimura and S. Nakanishi, "Weight-decay as a process of redundancy reduction," in *Proc. World Congr. Neural Networks*, vol. III, 1994, pp. 486–489.

[24] A. J. F. van Rooij, L. C. Jain, and R. P. Johnson, *Neural Networks Training Using Genetic Algorithms*, Singapore: World Scientific, 1996, vol. 26, Series in Machine Perception and Artificial Intelligence.

[25] S. V. Odri, D. P. Petrovacki, and G. A. Krstonosic, "Evolutional development of a multilevel neural network," *Neural Networks*, vol. 6, pp. 583–595, 1993.

[26] R. Smalz and M. Conrad, "Combining evolution with credit apportionment: A new learning algorithm for neural nets," *Neural Networks*, vol. 7, no. 2, pp. 341–351, 1994.

[27] V. Maniezzo, "Genetic evolution of the topology and weight distribution of neural networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 39–53, Jan. 1994.

[28] M. V. Borst, "Local structure optimization in evolutionary generated neural networks architectures," Ph.D. dissertation, Leiden Univ., Leiden, The Netherlands, 1994.

[29] B. A. Whitehead and T. D. Choate, "Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Trans. Neural Networks*, vol. 7, July 1996.

[30] G. Bebis, M. Georgiopoulos, and T. Kasparis, "Coupling weight elimination with genetic algorithms to reduce network size and preserve generalization," *Neurocomputing*, vol. 17, pp. 167–194, 1997.

[31] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Trans. Syst. Man, Cybern. B*, vol. 28, pp. 417–425, June 1998.

[32] D. E. Moriarty and R. Miikkulainen, "Forming neural networks through efficient and adaptive coevolution," *Evol. Comput.*, vol. 4, no. 5, 1998.

[33] D. E. Goldberg, "Genetic algorithms and walsh functions: Part 2, deception and its analysis," *Complex Syst.*, vol. 3, pp. 153–171, 1989.

[34] ——, "Genetic algorithms and walsh functions: Part 1, a gentle introduction," *Complex Syst.*, vol. 3, pp. 129–152, 1989.

[35] R. K. Belew, J. McInerney, and N. N. Schraudolph, "Evolving networks: Using genetic algorithms with connectionist learning," Comput. Sci. Eng. Dept., Univ. California, San Diego, Tech. Rep. CS90-174, 1991.

[36] P. J. B. Hancock, "Genetic algorithms and permutation problems: A comparison of recombination operators for neural net structure specification," in *Proc. Int. Workshop Combinations Genetic Algorithms Neural Networks*, L. D. Whitley and J. D. Schaffer, Eds., Los Alamitos, CA, 1992, pp. 108–122.

[37] J. D. Schaffer, L. D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *Proc. Int. Workshop Combinations Genetic Algorithms Neural Networks*, L. D. Whitley and J. D. Schaffer, Eds., Los Alamitos, CA, 1992, pp. 1–37.

[38] D. B. Fogel, "Evolving artificial intelligence," Ph.D. dissertation, Univ. California, San Diego, CA, 1992.

[39] G. F. Miller, P. M. Todd, and S. U. Hedge, "Designing neural networks," *Neural Networks*, vol. 4, pp. 53–60, 1991.

[40] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, no. 10, pp. 1399–1404, Dec. 1999.

[41] B. E. Rosen, "Ensemble learning using decorrelated neural networks," *Connection Sci.*, vol. 8, no. 3, pp. 373–384, Dec. 1996.

[42] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Machine Learning*, vol. 22, pp. 11–32, 1996.

[43] A. L. Samuel, "Some studies in machine learning using the game of checkers," *J. Res. Development*, vol. 3, no. 3, pp. 210–229, 1959.

[44] D. W. Opitz and J. W. Shavlik, "Actively searching for an effective neural network ensemble," *Connection Sci.*, vol. 8, no. 3, pp. 337–353, 1996.

[45] Q. F. Zhao, O. Hammami, K. Kuroda, and K. Saito, "Cooperative co-evolutionary algorithm—How to evaluate a module?," in *Proc. 1st IEEE Symp. Evol. Comput. Neural Networks*, San Antonio, TX, May 2000, pp. 150–157.

[46] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[47] D. Whitley and J. Kauth, "Genitor: A different genetic algorithm," in *Proc. Rocky Mountain Conf. Artificial Intell.*, Denver, CO, 1988, pp. 118–130.

[48] D. Whitley, "The genitor algorithm and selective pressure," in *Proc 3rd Int. Conf. Genetic Algorithms*, Los Altos, CA, 1989, pp. 116–121.

[49] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 2–9.

[50] D. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*, G. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 94–101.

[51] D. Whitley and T. Starkweather, "Genitor II: A distributed genetic algorithm," *J. Experimental Theoretical Artificial Intelligence*, pp. 189–214, 1990.

[52] G. Syswerda, "A study of reproduction in generational and steady-state genetic algorithms," in *Foundations of Genetic Algorithms*, G. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 94–101.

[53] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, pp. 79–87, 1991.

[54] D. Rumelhart, G. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, D. Rumelhart and J. McClell, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[55] A. A. Minai and R. D. Williams, "Acceleration of back-propagation through learning rate and momentum adaptation," in *Proc. Int. Joint Conf. Neural Networks*, vol. 1, Jan. 1990, pp. 676–679.

[56] *Neural Computing: A Technology Handbook for Professional II/Plus*, NeuralWare Inc., Pittsburgh, PA, 1993.

[57] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing*, D. S. Touretzky, Ed. Denver, CO, 1990, pp. 598–605.

[58] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing (1)*, D. S. Touretzky, Ed. Denver, CO, 1989, pp. 107–155.

[59] L. Prechelt, "Proben1—A set of neural network benchmark problems and benchmarking rules," Fakultät für Informatik, Univ. Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94, 1994.

[60] W. Finnoff, F. Hergert, and H. G. Zimmermann, "Improving model selection by nonconvergent methods," *Neural Networks*, vol. 6, pp. 771–783, 1993.

[61] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Using the adap learning algorithm to forecast the onset of diabetes mellitus," in *Proc. Symp. Comput. Applicat. Medical Care*, 1988, pp. 261–265.

[62] L. Prechelt, "A quantitative study of experimental evaluation of neural network learning algorithms," *Neural Networks*, vol. 9, pp. 457–462, 1996.
[63] W. J. Conover, *Practical Nonparametric Statistics*. New York: Wiley, 1971.
[64] T. W. Anderson, "An introduction to multivariate statistical analysis," in *Wiley Series in Probability and Mathematical Statistics*, 2nd ed. New York: Wiley, 1984.
[65] "An Introduction to R," R Development Core Team, 2000.
[66] S. E. Fahlman and C. Lebiere, "The cascade-correlation architecture," in *Advances in Neural Information Systems 2*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kauffman, 1990, pp. 524–532.
[67] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*. London, U.K.: Ellis Horwood, 1994.
[68] N. K. Treadgold and T. D. Gedeon, "Exploring constructive cascade networks," *IEEE Trans. Neural Networks*, vol. 10, pp. 1335–1350, Nov. 1999.
[69] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee, and V. Froelicher, "International application of a new probability algorithm for the diagnosis of coronary artery disease," *Amer. J. Cardiology*, vol. 64, pp. 304–310, 1989.
[70] K. P. Bennet and O. L. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization Methods Software*, vol. 3, pp. 728–734, 1992.
[71] B. Mulgrew and C. F. N. Cowan, *Adaptive Filters and Equalizers*. Boston, MA: Kluwer, 1988.
[72] A. Roy, S. Govil, and R. Miranda, "An algorithm to generate radial basis function (RBF)-like nets for classification problems," *Neural Networks*, vol. 8, no. 2, pp. 179–201, 1995.
[73] "SPSS ©9.0 Advanced Models," SPSS Inc., Chicago, IL, 1999.
[74] N. García-Pedrajas, E. Sanz-Tapia, and C. Hervás-Martínez, "Introducing multi-objective optimization in cooperative coevolution of neural networks," in *Connectionist Models of Neurons, Learning Processes and Artificial Intelligence*, J. Mira and A. Prieto, Eds. Heidelberg, Berlin: Springer-Verlag, 2001, vol. 2084, Lecture Notes in Computer Science, pp. 645–652.
[75] N. García-Pedrajas, C. Hervás-Martínez, and J. Muñoz-Pérez, "Multi-objective cooperative coevolution of artificial neural networks," *Neural Networks*, vol. 15, no. 10, pp. 1255–1274, Nov. 2002.
[76] K. Deb, "Evolutionary algorithms for multi-criterion optimization in engineering design," in *Proc. Evolutionary Algorithms Engineering Computer Science*, Jyväskylä, Finland, June 30, 1999, pp. 135–161.

Nicolás García-Pedrajas (M'01–A'01) was born in Córdoba, Spain, in 1970. He received the B.S. degree in computing and the Ph.D. degree, both from the University of Málaga, Málaga, Spain, in 1993 and 2001, respectively.

He is a Professor in the Department of Computing and Numerical Analysis in the Area of Computer Science and Artificial Intelligence, University of Córdoba, Córdoba, Spain. His dissertation research involves the automatic design of artificial neural networks using both genetic algorithms and evolutionary programming. His current research interests include neural networks, evolutionary computation, and game playing.

Dr. García-Pedrajas is a member of several technical societies including the ACM, the INNS, and the IEEE Computer Society.

César Hervás-Martínez was born in Cuenca, Spain. He received the B.S. degree in statistics and operating research from the Universidad Complutense, Madrid, Madrid, Spain, in 1978 and the Ph.D. degree in mathematics from the University of Seville, Seville, Spain, in 1986.

He is a Professor with the University of Córdoba in the Department of Computing and Numerical Analysis in the area of computer science and artificial intelligence and an Associate Professor in the Department of Quantitative Methods in the School of Economics. His current research interests include neural networks, evolutionary computation, and modeling of natural systems.

José Muñoz-Pérez was born in Cazorla, Spain. He received the B.S. degree in mathematics from the University of Granada, Granada, Spain, in 1974 and the Ph.D. degree in mathematics at the University of Seville, Seville, Spain, in 1986.

He is a Professor with the Department of Computation Sciences in the Area of Computing and Artificial Intelligence, University of Malaga, Malaga, Spain, and is Director of Image Processing Institute in the Technological Park of Andalusia. His current research interests include neural networks and image processing.

Dr. Muñoz-Pérez is member of the Spanish Association for Artificial Intelligence.