
PERGAMON

Neural Networks 15 (2002) 1259–1278

Neural Networks

www.elsevier.com/locate/neunet

Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks)

N. García-Pedrajas[a,*], C. Hervás-Martínez[a], J. Muñoz-Pérez[b]

[a]*Department of Computing and Numerical Analysis, University of Córdoba, 14071 Córdoba, Spain*
[b]*Department of Languages and Computer Science, University of Málaga, 29071 Málaga, Spain*

Received 20 November 2001; accepted 13 June 2002

Abstract

In this paper we present a cooperative coevolutive model for the evolution of neural network topology and weights, called MOBNET. MOBNET evolves subcomponents that must be combined in order to form a network, instead of whole networks. The problem of assigning credit to the subcomponents is approached as a multi-objective optimization task. The subcomponents in a cooperative coevolutive model must fulfill different criteria to be useful, these criteria usually conflict with each other. The problem of evaluating the fitness on an individual based on many criteria that must be optimized together can be approached as a multi-criteria optimization problems, so the methods from multi-objective optimization offer the most natural way to solve the problem.

In this work we show how using several objectives for every subcomponent and evaluating its fitness as a multi-objective optimization problem, the performance of the model is highly competitive. MOBNET is compared with several standard methods of classification and with other neural network models in solving four real-world problems, and it shows the best overall performance of all classification methods applied. It also produces smaller networks when compared to other models.

The basic idea underlying MOBNET is extensible to a more general model of coevolutionary computation, as none of its features are exclusive of neural networks design. There are many applications of cooperative coevolution that could benefit from the multi-objective optimization approach proposed in this paper.

Keywords: Evolutionary computation; Multi-objective optimization; Cooperative coevolution; Topology evolution; Automatic neural network design

1. Introduction

The selection of the architecture of a neural network (Haykin, 1994) suitable to solve a given problem is one of the most important aspects of neural network research. It is known that a network smaller than needed would be unable to learn, and a network larger than needed would probably end in over-training.

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 Yao (1999)). Although the developed theory assures that a neural network with a hidden layer can approximate any function (Cybenko, 1989; Hornik & Stinchombe, 1992), there is no information about how to find such network. It has been proved (Judd, 1988; Sima, 1994) that, even if the optimal architecture of a network is known, the problem of finding a set of weights for which the network performs the desired mapping (known as the *loading problem*) is NP-complete. Based on that theoretical results Blum and Rivest (1992) stated that it is unlikely that any algorithm which simply varies weights on a network of fixed size can learn in polynomial time. Judd (1990) suggested that one of the possibilities to avoid this problem of NP-completeness is to modify the structure of the networks as the learning progresses. Baum (1991) and Baum and Haussler (1989) has given an existence proof that suggests that the dynamic modification of the topology during the learning process can dramatically reduce the training time. So, our model is based on evolving both the topology and the weights of the networks, following those suggestions.

Modular systems are often used in machine learning as an approach for solving complex problems. Moreover, in spite of the fact that small networks are preferred for their better performance, the error surfaces of such networks are

* Corresponding author. Tel.: +34-957-211032; fax: +34-957-218630.
E-mail addresses: npedrajas@uco.es (N. García-Pedrajas), chervas@uco.es (C. Hervás-Martínez), munozp@lcc.uma.es (J. Muñoz-Pérez).

0893-6080/02/$ - see front matter
PII: S0893-6080(02)00095-3

more rugged and have few good solutions (Shang & Wah, 1996). In addition, there is much neuropsychological evidence showing that certain parts of the brain of humans and other animals, especially for peripheral mechanisms, consist of modules, which are subdivided into identifiable parts, each one with its own purpose and function (Cho & Shimohara, 1998). So, we based our study on the evolution of modular neural networks (Caelli, Guan, & Wen, 1999) as in the last few years modular models have shown to perform better than simple neural network models (Caelli, Squire, & Wild, 1993; Guan, Anderson, & Sutton, 1997; Lin, King, & Lin, 1997).

Evolutionary computation (Goldberg, 1989; Michalewicz, 1994) is a set of global optimization techniques that have been widely used in recent years for training and automatically designing neural networks. Some efforts have been made in designing modular neural networks with these techniques (Yao & Liu, 1997), but almost all of them use some a priori knowledge of the problem to be solved, greatly restricting the area of application of those models (for a very interesting review in modularity on neural networks refer to Caelli et al. (1999)).

Cooperative coevolution (Potter & Jong, 2000) is a recent paradigm in evolutionary computation. The objective of cooperative coevolution is to evolve subcomponents that must cooperate in solving a given task, instead of evolving whole individuals. Some works in this area (Giordana & Neri, 1996; Moriarty & Miikkulainen, 1996; Potter & Jong, 2000) have proved that cooperative coevolution is a promising area for developing solutions for solving complex tasks. Cooperative coevolution provides a natural and simple way to implement modularity, as each subcomponent of a cooperative model could be considered as a module in a neural network topology design environment.

One of the key aspects of cooperative coevolution is the estimation of the fitness of the subcomponents that cooperate to solve a task. This is known as the *credit assignment problem*. In a standard evolutionary algorithm the evaluation of the individual is available and the fitness assignment is straightforward. On the other hand, in cooperative coevolution only the whole individual could be evaluated, and no direct evaluation of the subcomponents is available. So it is a very difficult problem to assign each subcomponent a portion of the fitness of the whole individual. Nevertheless, the success of the cooperative model highly depends on this credit assignment. So, our work is focused on the problem of assigning credit to the subcomponents that make up a solution taking into account more than one criterion. Some different solutions have been developed (Giordana & Neri, 1996; Hillis, 1991; Paredis, 1995; Rosin & Belew, 1997), but all these models focus on specific tasks.

The concepts of *subcomponent* and *individual* in a cooperative coevolutionary model must be defined so the difference between them is made clear. We will use the following definitions:

Definition 1 (Subcomponent). A member of a population or subpopulation that is subject to an evolutionary process and that must be combined with other subcomponents to make up a solution to the given task.

Definition 2 (Individual). A member of a population that is subject to an evolutionary process, that is made up of several subcomponents, and constitutes a solution to the given task.

With these definitions it is clear that we can evaluate the performance of an individual; but it is also clear that such evaluation is not possible in a subcomponent.

Some methods have been proposed for the estimation of the fitness of the subcomponents in a cooperative environment. However, most of them focus only on the performance of the whole individuals, disregarding the contribution of each subcomponent to that performance.

In a previous work (García-Pedrajas, Hervás-Martínez, & Muñoz-Pérez, 2001a) we have shown how combining different criteria[1] the assignment of credit to subcomponents could be improved. However, the combination of several criteria to obtain the fitness of each subcomponent has two major problems:

1. The relevance of each criteria must be pondered. This is a complex and time consuming task, involving a long process of trial and error.
2. The ranges of the objectives are different. In many cases if the objectives are not weighted, some of the objectives could be overshadowed by objectives with higher values.

These two problems become more important as more objectives are introduced in the evaluation of the fitness of an individual, preventing the addition of interesting criteria.

On the other hand, if we approach this problem as a multi-objective optimization task, we will benefit from many advantages. First, there is no need to weight the objectives as their relevance and range do not matter any more. Secondly, the solutions based on Pareto optimality guarantee the diversity of the final population. And third, there is an underlying theory applicable to our problem.

Moreover, as the two problems stated above disappear, we can add new objectives to the evaluation of the fitness. So, the use of more objectives allows the designer to apply a priori knowledge in defining the criteria that must fulfill a solution to the problem to be useful. Nevertheless, as the objectives defined in this paper are generic, the absence of any a priori knowledge does not prevent the application of this fitness estimation method.

The principles of multi-objective optimization are quite

[1] We will use the term *criterion* when we refer to a feature that is interesting in a subcomponent, and the term *objective* as the representation of a criterion in a measurable quantity that can be used in an optimization process.

different from those of single objective optimization. The main goal in a single-objective optimization task is to find the global optimal solution. However, in a multi-objective optimization problem, there is more than one objective function, each of which may have a different individual optimal solution. If there is sufficient difference in the optimal solutions corresponding to different objectives, the objective functions are often known as conflicting to each other.

Multi-objective optimization with such conflicting objective functions gives rise to a set of optimal solutions, instead of one optimal solution, as none of the solutions could be considered to be *the best one*, because none of them is better than any other with respect to all objective functions. These optimal solutions have a special name—Pareto-optimal solutions.

As the concept of optimality in multi-objective optimization deals with several solutions, we must define some new concepts to characterize a solution to a multi-objective optimization problem.

The basic concept in multi-objective optimization is the concept of domination. Being $x^{(i)}$ a solution and $\mathbf{f}(x^{(i)}) = (f_1(x^{(i)}), f_2(x^{(i)}), \ldots, f_M(x^{(i)}))$ the evaluation of that solution for every objective, this concept is defined as follows:

Definition 3 (Domination). A solution $x^{(1)}$ is said to dominate another solution $x^{(2)}$ if both the following conditions are true:

1. The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives, that is, $f_j(x^{(1)}) \nprec f_j(x^{(2)})$ for all $j = 1, 2, \ldots, M$ objectives, where $\prec$ denotes worse.
2. The solution $x^{(1)}$ is strictly better than the solution $x^{(2)}$ in at least one objective, that is, $f_{\bar{j}}(x^{(1)}) \succ f_{\bar{j}}(x^{(2)})$ for at least one $\bar{j} \in \{1, 2, \ldots, M\}$, where $\succ$ denotes better.

Similar to the definitions of local and global optimal solutions in single-objective optimization, the following definitions are their equivalent in multi-objective optimization:

Definition 4 (Local Pareto-optimal set). If for every member x in a set $\underline{P}$, there exists no solution y satisfying $\|y - x\|_\infty \leq \epsilon$, where ϵ is a small positive number, which dominates any member in the set $\underline{P}$, then the solutions belonging to the set $\underline{P}$ constitute a local Pareto-optimal set.

Definition 5 (Global Pareto-optimal set). If there exists no solution in the search space which dominates any member in the set $\underline{P}$, then the solutions belonging to the set $\underline{P}$ constitute a global Pareto-optimal set.

There are two primary goals that a multi-objective optimization algorithm must achieve:

1. Guide the search towards the global Pareto-optimal region.
2. Maintain the population diversity in the Pareto-optimal front.

Although a discussion about multi-objective evolutionary algorithm is outside the scope of this paper there are many reviews on the matter for the interested reader (Deb, 1999; Fonseca & Flemming, 1995; Zitzler, 1999). Many classical algorithms have been proposed for solving multi-objective problems (Steuer, 1986). However, evolutionary computation offers some interesting features to solve this kind of problems (Deb, 1999).

The multi-objective evolutionary algorithms are based on the concept of non-domination, and almost all of them follow the principles sketched by Goldberg (1989).[2] Among the most successful algorithms are the multi-objective genetic algorithm of Fonseca and Flemming (1993), the niched Pareto genetic algorithm of Horn, Nafploitis, and Goldberg (Horn, Goldberg, & Deb, 1994), the strength Pareto evolutionary algorithm of Zitzler and Thiele (1998) and the non-dominated sorting genetic algorithm of Srinivas and Deb (1994). The algorithm we have applied is basically an adaptation of the latter to evolutionary programming.

This paper is organized as follows. Section 2 discusses the related work found in the bibliography; Section 3 describes the proposed model, MOBNET; Section 4 explains the application of multi-objective optimization to the problem of fitness estimation; Section 5 shows the experimental results obtained applying MOBNET to real world problems; and finally Section 6 states the conclusions of our work.

2. Related work

There are some different methods for estimating the fitness of the subcomponents that make up an individual in cooperative coevolution. For a short review the reader could refer to García-Pedrajas, Hervás-Martínez, Muñoz-Pérez (submitted). Nevertheless, very few authors have considered the application of multi-objective methods to this estimation. We have not found any work applying multi-objective optimization to cooperative coevolution in the bibliography.

Perhaps, the most related paper is the application of Pareto-optimality to competitive coevolution of Ficici and Pollack (2001). In their work each member of the population is considered as an objective to be optimized. The population is divided into teachers and learners. An agent in the population may act as both of them. As an agent interacts with the rest of the agents of the population, each one is considered as an objective to be optimized. So the teachers *create gradients* that the learners must *follow*. They

[2] Although the first practical implementation was suggested by Schaffer (1984), and the first ideas can be traced back to the work of Rosenberg (1967).

use a multi-objective method close to NSGA (Srinivas & Deb, 1994).

2.1. Subcomponents fitness estimation

As we have stated, the estimation of the fitness of the subcomponents of an individual in cooperative coevolution is a key aspect. It is also a difficult task, as it is not clear how to measure the relevance of each subcomponent.

Nevertheless, most of the cooperative models based the estimation of this fitness just on the performance of the individual made up of different subcomponents, as it is the only one that can be evaluated in terms of the task. When an individual performs well, every one of its subcomponents receives an equal portion of its fitness. This method rewards subcomponents that are not playing an important part in the performance of the individual, as there is no difference in the assignment of fitness to the different subcomponents of an individual.

This is exemplified by the model proposed by Potter and Jong (2000) where the fitness of each subcomponent is obtained combining it with the best subcomponent of every other population. This method, however, does not reward combinations of suboptimal subcomponents that could yield to better individuals when combined.

Moriarty and Miikkulainen (1996) coevolve one population of nodes that are combined into another population of networks. Each node receives a fitness that is the averaged performance of the networks where it participates.

Other coevolutionary models estimate the fitness of the subcomponents with methods closer to the problem they are applied. R. Smalz and Conrad (1994) developed a model with several subpopulations of nodes for a multi-layer neural network. The credit assignment process rewards those nodes within each competing subpopulation that are more compatible with the networks that perform better with respect to a specific external input. Whitehead and Choate (1996) evolved a population of RBF units that made up a network. The fitness of each individual is assigned depending on the spatial disposition of its weight vector.

Haynes and Sen (1997) developed a model of coadaptation of a team using genetic programming (Koza, 1994) where the fitness of each individual[3] is estimated evaluating three different criteria. This evaluation shares our basic idea of considering all potentially interesting aspects in evaluating an individual.

A model more similar to MOBNET is developed by Puppala, Sen, and Gordin (1998). They evolve two populations of cooperating agents and keep track of the best combinations in a shared memory segment. However, they do not evolve these combinations. As in other works the fitness of the subcomponents is evaluated taking into account only the performance of the combinations. In this paper each individual receives the fitness of the performance of the best combination where it is present.

A model that shares our basic idea of combining different objectives in assigning credit to individuals is the coevolutionary model proposed by Yao and Shi (1995). They evolve a population of rules that generate a network, so the assignment of credit to each rule is not straightforward. At every generation the fitness of the network is obtained from some different criteria: network payoff, number of epochs, and number of nodes and connections. The fitness of the rules that do not participate in generating the network is unchanged. The rules that contribute to the network update their fitness following also different criteria.

This kind of model could benefit from incorporating the multi-objective concepts in the evolutionary process that we use in our model, as many other, such as the evolution of a cooperative population of networks using mutual information (Liu, Yao, Zhao, & Higuchi, 2001), or the models of negative correlated ensembles of neural networks (Liu & Yao, 1999; Liu, Yao, & Higuchi, 2000).

3. MOBNET

In this section we will explain the proposed model in depth. As we have stated, MOBNET is a cooperative coevolutive model. Instead of trying to evolve just one individual to solve a given problem, many subcomponents are evolved together, and a combination of them constitutes a solution to the given task. MOBNET shares some of the design principles of COVNET, another cooperative model (García-Pedrajas et al., 2001a) that does not use multi-objective optimization techniques. The new idea of MOBNET is the introduction of multi-objective optimization in the process of subcomponent fitness assignment.

The evolution operates over two different populations: one population of subnetworks or modules (known in our model as *nodules*) and another population of networks. The two populations evolve concurrently. Each network is a combination of a fixed number of nodules. More formally, a nodule can be defined as follows:

Definition 6 (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 nodule is shown in Fig. 1.

A network is the combination of a number of nodules. The output of the network is the sum of the outputs of each nodule. So a network is defined as:

Definition 7 (Network). A network is the combination of a finite number of nodules. The output of the network is the

[3] The members of the population are whole individuals that collaborate, not subcomponents that must be combined.

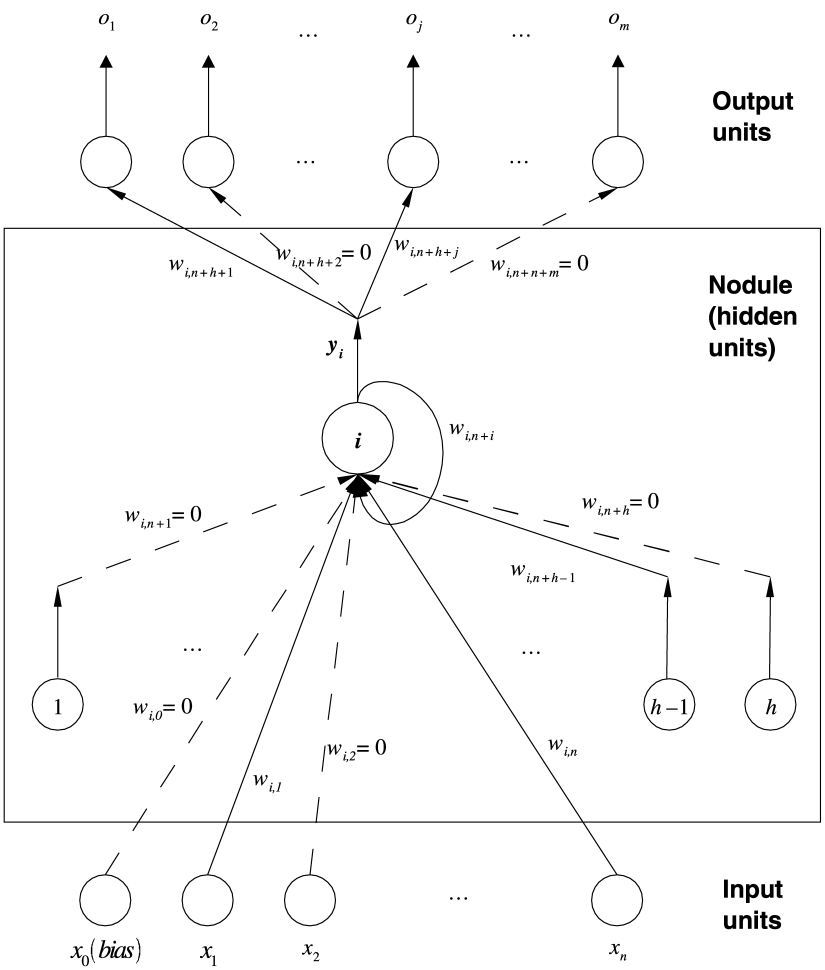

Fig. 1. Model of a nodule. As a node has only connections to some nodes of the nodule, the missing connections are represented with dashed lines. The nodule is composed by the hidden nodes and the connections of these nodes from the input nodes and to the output nodes, and the connections among hidden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

sum of the outputs of all the nodules that constitute the network.

As we have defined the concept of nodule and network, the network is recurrent. As we are constructing feed-forward neural networks, we define the transmission of impulse in three steps to avoid this recurrence. These steps are the following for a nodule with n inputs and h nodes:

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

$$p_i = f^i\left(\sum_{j=0}^{n} w_{i,j}x_j\right), \tag{1}$$

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:

$$y_i = f^i\left(\sum_{j=0}^{n} w_{i,j}x_j + \sum_{j=1}^{h} w_{i,n+j}p_j\right). \tag{2}$$

Step 3.
Finally, the output layer of the nodule generates its output:

$$o_j = f^{\text{output}}\left(\sum_{i=1}^{h} w_{i,n+h+j}y_i\right). \tag{3}$$

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. With this definition of the transmission of impulse, the nodule depicted in Fig. 1 is equivalent to a network of two hidden layers that is shown in Fig. 2.

The population of nodules consists of a fixed number of subpopulations that evolve without interchanging genetic information among them. The isolation of the populations is

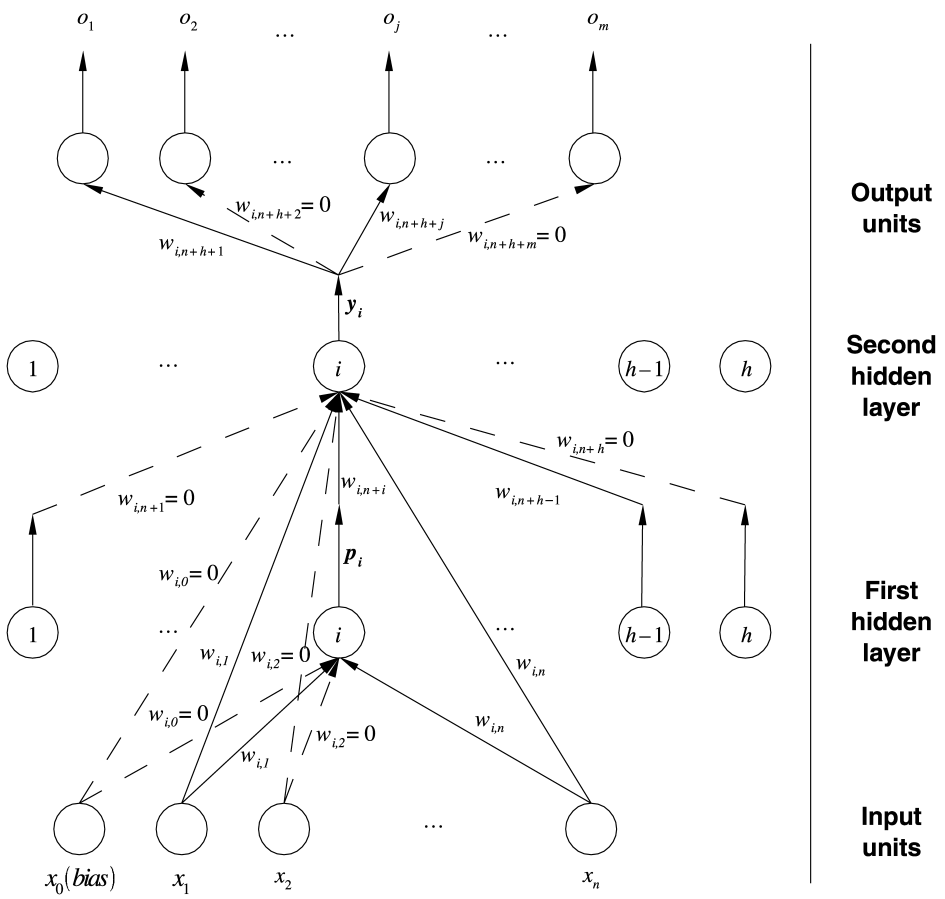

Fig. 2. Equivalent model with two hidden layers. Every connection from an input node represents two connections, as the input value is used in two steps (see Eqs. (1) and (2)). Every connection from another node of the nodule represents a connection between the first and second hidden layer (see Eq. (2)).

a natural solution to the problem of maintaining the diversity needed for coadapting subcomponents. The isolation also avoids the negative effect of cross-species mating (Potter & Jong, 2000), as it has been demonstrated in nature (Smith, 1989) that the viability of the offspring is highly correlated with the genetic distance of the parents.

Each subpopulation is formed by a fixed number of nodules. The evolution of each subpopulation is shown in Fig. 3. This evolution is within the definition of evolutionary programming. The most important steps of the evolutionary process are the following:

1. Generate the initial members of every subpopulation 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 $[w_{\min}, w_{\max}]$.
2. At every generation the best $p\%$ of the population is replicated. The rest $(1-p)\%$ is deleted.
3. The deleted $(1-p)\%$ of the population is refilled selecting an individual from the $p\%$ best of the population by means of a roulette selection. This individual is mutated and added to the new subpopulation.
4. The fitness of the individuals of the new subpopulation is calculated and the process is repeated until the stop criterion is reached.

This algorithm is similar to other evolutionary algorithms in the bibliography (Angeline, Saunders, & Pollack, 1994; Bebis, Georgiopoulos, & Kasparis, 1997; Yao & Liu, 1997).

As there is no migration among the subpopulations, 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 performance (Moriarty & Miikkulainen, 1996). As we have stated, it is known that migration among different species is prone to produce non-viable offspring, and the mixing of genetic material might reduce the diversity of the subpopulations.

Evolutionary programming relies on mutation as the only genetic operator. Many authors agree (Angeline et al., 1994; Yao & Liu, 1997) in the negative effect of the cross-over operator over the evolution of neural networks. There are two forms of mutation: parametric mutation and structural mutation.

Parametric mutation only affects the weights of the nodule. The mutation operator consists of a simulated

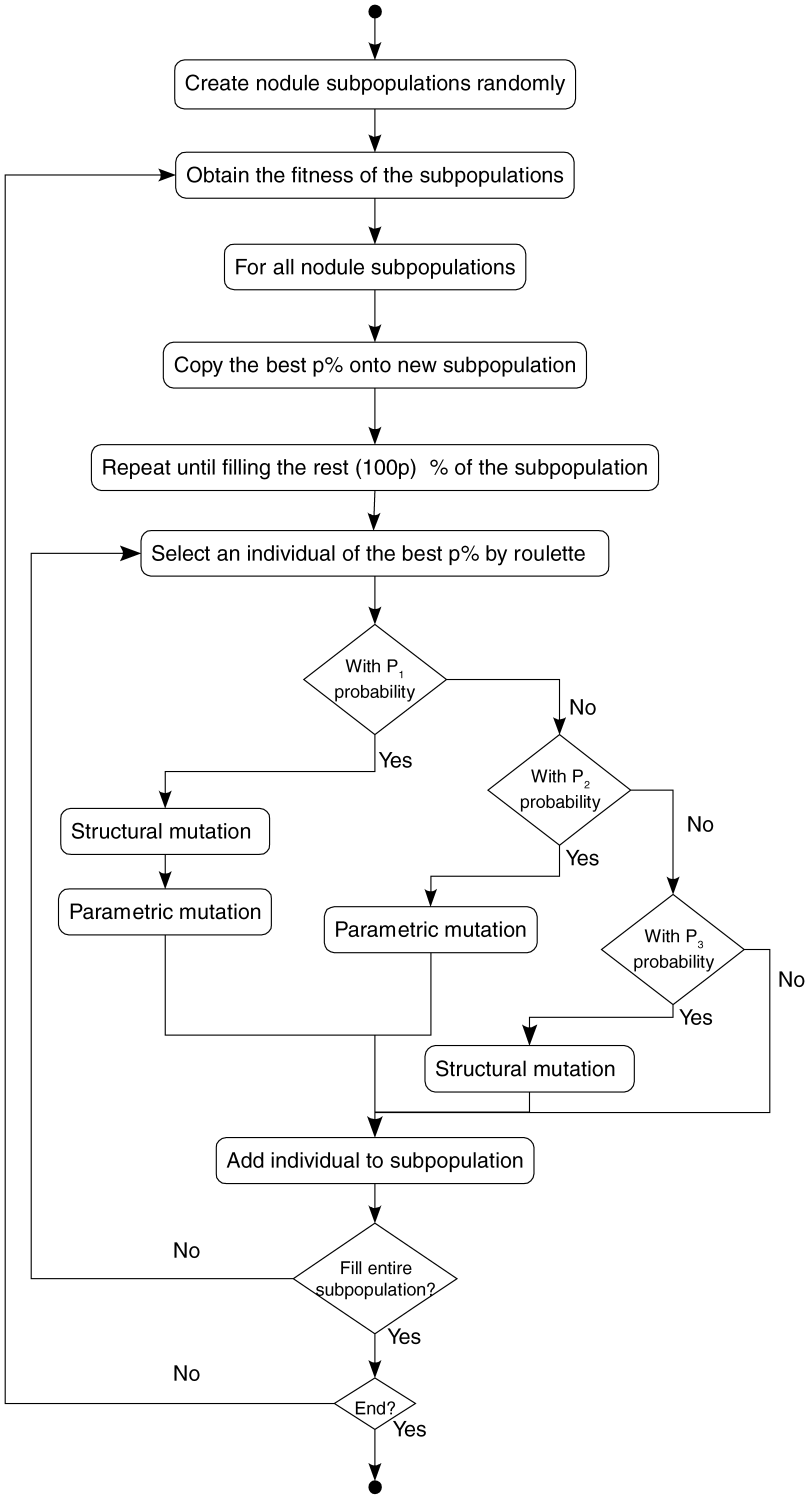

Fig. 3. Evolutionary process of nodule population. The generation of a new population of nodules is represented in detail. The fitness of the individuals of the nodule subpopulations are evaluated in parallel, allowing the model to be run in a distributed system more efficiently. The probabilities P_1, P_2 and P_3 are fixed by the user.

annealing algorithm (Kirkpatrick & Vecchi, 1983). Each step of the algorithm consists of adding a small random value to every weight of the nodule. The use of this kind of hybrid training is common in the bibliography, using simulated annealing (Yao & Liu, 1997) or different back-propagation algorithms (Bebis et al., 1997). The standard criterion for acceptance of the random step in simulated annealing is modified to take into account the concept of domination. Being x^{old} and x^{new} the vectors of objectives of the individual before and after the random step, respectively,

Table 1
Parameters of nodule structural mutations common to all the experiments

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

the criterion used for accepting a step is:

1. If x^{new} dominates x^{old}, the step is accepted.
2. If x^{old} dominates x^{new}, the step is rejected.
3. If neither x^{old} dominates x^{new} nor x^{new} dominates x^{old}, then the step is accepted with probability

$$P(x^{old}, x^{new}) = e^{-d(x^{old}, x^{new})/T(t)}, \quad (4)$$

where $d(\cdot)$ is the normalized Euclidean distance and $T(t)$ is the temperature at time t (as usual $T(t+1) = \alpha T(t)$ and $T(0) = T_0$). This criterion avoids large steps in the objectives space, in order to keep the consistence of the evolution.

As this algorithm implies a high computational cost, we perform just a few steps. However, the use of simulated annealing instead of a random step in the weight space significatively improves the performance of the model.

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 (Goldberg, 1989). We define four different structural mutations that allow the nodule to reach any topology from its starting point (Angeline et al., 1994). These mutations are:

1. *Addition of a node*. The node is added with no connections (Angeline et al., 1994) to enforce the behavioral link with its parent. As many authors have stated (Yao & Liu, 1997), maintaining the behavioral link between parents and their offsprings is very important to get a useful algorithm.
2. *Deletion of a node*. A node is selected randomly and deleted together with its connections.
3. *Addition of a connection*. A connection is added, with 0 weight (Angeline et al., 1994), to a randomly selected node.
4. *Deletion of a connection*. A randomly selected connection is removed.

When a nodule is subject to structural mutation, all the above mutations could be performed over it. For each different structural mutation type 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

$$\Delta = \Delta_m + F_r(\nu)(\Delta_M - \Delta_m). \quad (5)$$

where F_r is the relative fitness of the nodule, which is obtained from the fitness of the nodule, $F(\nu)$, $F(\nu) = e^{-\beta F(\nu)}$.

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 these structural mutation parameters used in all our experiments are shown in Table 1.

3.1. 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. It is important to note that, as the chromosome that represents the network is ordered, the permutation problem (Belew, McInerney, & Schraudolph, 1991; Hancock, 1992; Schaffer, Whitley, & Eshelman, 1992) cannot appear.

The network population is evolved using the *steady-state* genetic algorithm (Whitley, 1989; Whitley & Kauth, 1988). 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 also been shown by some works in the area (Syswerda, 1991; Whitley & Starkweather, 1990) that the steady-state genetic algorithm produces better solutions than the standard genetic algorithm.

This algorithm has some features that are different from the standard genetic algorithm. The cross-over 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. Also, the fitness is assigned to the members of the population according to their rank and not as their absolute fitness value. The algorithm allows to add 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.

Cross-over is made at nodule level. A standard two-point cross-over is used, and 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 subpopulation, some nodules of every population are removed and

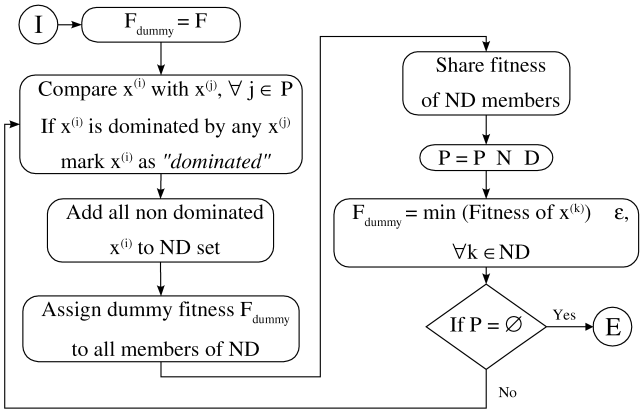

Fig. 4. Multi-objective evolutionary algorithm for obtaining the fitness of the individuals of the population.

substituted by new ones. 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; second, the new nodules have the opportunity to participate in the networks immediately after their creation (Moriarty & Miikkulainen, 1998).

4. Multi-objective evaluation of fitness

The multi-objective algorithm is common to both populations, nodules and networks. We will consider a population of individuals where individual i has a vector of objectives values $x^{(i)}$. The population has N individuals and M objectives are considered.

The proposed algorithm is based on the concept of Pareto optimality explained above. It has common points with other multi-objective evolutionary algorithms. The multi-objective algorithm for obtaining the fitness of an individual with a vector of objectives $x^{(i)}$ is outlined in Fig. 4.

This algorithm is basically an adaptation of NSGA (Srinivas & Deb, 1994) to evolutionary programming. The idea underlying NSGA is the use of a ranking selection method to emphasize current non-dominated individuals and a niching method to maintain diversity in the population.

The algorithm consists of two stages. First, the successive non-dominated fronts[4] are obtained and every individual of these fronts is assigned an equal dummy fitness, F_{dummy}. Second, the members of every front share their fitness. The sharing procedure must guarantee that none of the member of a front gets a higher fitness than any of the member of the previous front.

The following algorithm is used for obtaining the non-dominated set of solutions (Deb, 1999):

[4] A non-dominated front is a subset of individuals that are not dominated by any member of the population.

Algorithm for obtaining i-th non-dominated Pareto-front.

for $i = 1$ to N
  for $j = 1$ to N
    if $j \neq i$ and $x^{(i)}$ is dominated by $x^{(j)}$ then
      Mark $x^{(i)}$ as "dominated"
    end if
  end for
end for
All solutions not marked *dominated* are non-dominated solutions.

Once the individuals of a non-dominated front are assigned their fitness, they are not considered any more for obtaining the new non-dominated front. That is the reason why we talk about *successive non-dominated fronts*.

The problem of fitness sharing among the members of the same Pareto front is crucial, as the performance of the algorithm depends highly on it. The standard method of explicit fitness sharing (Goldberg & Richardson, 1987) (for a discussion see Deb and Goldberg (1989)) cannot be applied as the individual is not in a Euclidean space. So we must define a measure of diversity for nodules and networks in order to apply the algorithm. Our interest is focused on keeping the behavioral diversity among the individuals, so our measures are based on the functional diversity of the nodules and networks. These measures are the following:

Nodule's Functional diversity. This measure is based on comparing the discrepancy between the outputs of the two nodules. In order to obtain this measure for two nodules n_1 and n_2, $fd(n_1, n_2)$, we set in turn every input connection to 1 and the rest of the connections to 0, and get the output of the nodules. We can consider that we are applying the following training set to the nodule:

$$\begin{matrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{matrix} \tag{6}$$

The functional diversity is the average distance of the output of the two nodules over all the inputs. This measure is used for applying a standard fitness sharing algorithm.

Similar measures of diversity have been used before. In Moriarty (1997) the outputs of the networks are used as a *function vector* that measures their diversity. This function vector is represented after performing a principal component analysis (Jolliffe, 1986) to visualize the diversity of the population.

Given a set of n_k nodules of the k-th non-dominated front each having a dummy fitness of F_{dummy}, the sharing procedure is performed for each solution $i = 1, 2, \ldots, n_k$. The sharing procedure consists of the following steps for

i-th individual:

1. Compute the functional diversity with every individual *j* of the Pareto front, $d_{ij} = fd(n_i, n_j)$.
2. Compare this value with a predefined niche radius, σ_{share}, and apply the sharing function:

$$sh(d_{ij}) = \begin{cases} 1 - \left(\dfrac{d_{ij}}{\sigma_{\text{share}}}\right)^2, & \text{if } d_{ij} \leq \sigma_{\text{share}}, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

3. Calculate niche count for individual *i*:

$$m_i = \sum_{j=1}^{n_k} sh(d_{ij}). \tag{8}$$

4. Modify the fitness of the individual according to its niche count:

$$f'_i = \frac{F_{\text{dummy}}}{m_i}. \tag{9}$$

Network's functional diversity. The same idea of the above method is applied to the network population. The measure of network functional diversity is based on the functional diversity of nodules. Given two networks $n^i = (n^i_1, n^i_2, \ldots, n^i_m)$ and $n^j = (n^j_1, n^j_2, \ldots, n^j_m)$ their functional diversity, ϕ, is defined as

$$\phi(n^i, n^j) = \sum_{k=1}^{m} fd(n^i_k, n^j_k). \tag{10}$$

With this distance measure the above sharing algorithm is applied, substituting the measure $fd(n_i, n_j)$ by $\phi(n^i, n^j)$.

Another option is applying fitness sharing on objectives function values (e.g. MOGA (Fonseca & Flemming, 1993)). This method could be applied to our model, but in many of our experiments it does not maintain diversity in the population of individuals.

In the above two methods, the determination of the value of σ_{share} is a very important issue. In most cases it is very difficult to obtain the most suitable value of σ_{share} for a problem. So we have defined a third measure that is not based on explicit fitness sharing.

This method is taken from Ficici and Pollack (2001) and is similar to the *competitive fitness paradigm* (Juillé & Pollack, 1996). Having an individual *x* in a Pareto front, we compare this individual with every member of the same front, *y*, obtaining the number of objectives in which *x* is better than *y*, n_x, and the number of objectives in which *y* is better than *x*, n_y. If $n_x > n_y$ then *x* receives $n_x - n_y$ points, otherwise receives 0 points. The points for each individual are accumulated and then the individuals are ranked according to those points. A standard linear ranking is used (Bäck, 1996) for converting the rank of each individual to a selection probability.

4.1. Nodule's objectives

As we have stated, the introduction of multi-objective optimization gives us the freedom to define many objectives to be fulfilled by the nodules. The diversity that guarantees the Pareto-optimal solutions allows the combination of nodules that are good in different objectives.

In this work we present the set of objectives that have achieved a good performance in the classification problems of the experimental stage. Many different objectives could be defined, specially when the available a priori knowledge of a problem allows the definition of some of the features of the networks. The objectives used in our work are the following,[5] for a nodule ν in a subpopulation π:

Difference (δ). The nodule is removed from all the networks where it is present, and the performance of such networks with the nodule removed is measured. The value of this criterion is measured as the difference in performance of these networks with and without the nodule. 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 0 and the subpopulations will be penalized.

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 subpopulation 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 can achieve a better performance than the rest of the nodules of its subpopulation.

Average performance. This objective is the average of the fitness values of the networks where the nodule ν is present. This criterion is the only one commonly used in cooperative coevolution (Moriarty, 1997; Potter, 2000).

Complexity (*three objectives*). Three objectives play the role of a regularization term. These objectives penalize big networks with respect to smaller ones. These objectives are the number of nodes, the number of connections and the sum of the weights of the nodules. All of them are taken in negative value, as we want to minimize them. In most problems selecting just the number of nodes is enough to control the size of the networks. If smaller networks are needed, the other two objective could be used.

Regularization theory (Girosi, Jones, & Poggio, 1995) is

[5] A subset of these objectives could be selected for each problem, as not all the objectives are suitable for all the problems.

based on adding a *smoothness functional* to the function to be optimized, which takes into account the smoothness of the functions implemented by the neural networks. The effect of the regularization term that we used is also the smoothing of the function of the network, by reducing the number of free parameters of the network, as the case of pruning (Goutte & Hansen, 1997; Hansen & Pedersen, 1994).

The use of regularization terms is common use in the evolution of neural networks. In most cases the fitness of the networks are penalized according to their size (Yao & Shi, 1995) or introducing some kind of pruning (Bebis et al., 1997). Nevertheless this has a negative effect over the networks, as the penalty term has a great impact on the fitness of the network. Introducing the regularization as another objective, allows the regularization of the networks without having a dramatic impact on their performance.

Count of networks. Number of networks where the nodule ν participates. It encourages the nodule to be part of the networks. Moreover, when a nodule participates in more networks, its fitness is more accurately estimated and must be rewarded.

This objective also avoids, in part, the negative effect after the elimination of a nodule that participates in many networks during the evolutionary process.

All the members of all the subpopulations of nodules evolve with the same subset of objectives.[6]

4.2. Network's objectives

Each individual of the population of networks has also several objectives. Initially only the performance of the network was considered, as this performance can be measured for networks. But, as the nodule population evolves better considering more objectives that can guide the evolution towards the most interesting subcomponents, this same principle could be applied to the evolution of networks.

So we also applied to networks the multi-objective evolutionary algorithm developed for nodules. The objectives defined are the following:

Performance. Performance of the network in solving the given task. As we have applied the model only to classification tasks, the performance of the network is just the number of patterns from the training set correctly classified.

Fitness of each nodule. As many objectives as nodule subpopulations. With this objective we intend to encourage the combination of the best individuals of the nodule subpopulations.

This objective has also a very interesting *side effect*, as the nodules with the highest fitness values will survive during the evolution, the combination of nodules with high fitness values are more durable and so there is more time for evolving to useful networks.

4.3. Stop criterion and selection of the best individual

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.

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

5. Experimental results

The tests were conducted following the guidelines of Prechelt (1994). Each set of available data was divided into three subsets: 50% of the patterns were used for learning, 25% of them for validation and the remaining 25% for testing the generalization of the individuals.

In none of the problems a patent over-training effect was observed in MOBNET.[7] So, the experiments were always carried out without using a separate validation set, and the patterns of this set were added to the training set. In the experiments with multi-layer perceptrons (MLP) and modular neural networks, the validation set was actually used as the over-training effect was evident.

For each data set three different random permutations of the patterns were made. For each permutation and for every model the experiments were repeated 10 times, except for the linear classifier and the C4.5 algorithm that are deterministic. The fitness of the individuals of all the models 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 in learning and generalization over 10 repetitions on each permutation of the data set, the standard deviation, and the best and worst individuals. The measure of the error is

$$E = \frac{1}{P}\sum_{i=1}^{P} e_i, \tag{11}$$

where P is the number of patterns and e_i is 0 if pattern i is correctly classified, and 1 otherwise. Each permutation of the data set are labelled 'I', 'II' and 'III' in the tables. The averaged results over the three permutations are labelled 'All'.

[6] We added another objective that measured the performance of the nodule taken isolated as a network. This objective was intended as a control for testing the cooperation among the nodules. In all the experiments the introduction of this objective yielded to worse results.

[7] This aspect is discussed in Section 5.5.

Table 2
MOBNET's parameters for the four data set

Parameter	Problem			
	Pima	Heart	Card	Glass
Number of networks	500	100	100	100
Nodule subpopulations	3	5	10	10
Number of nodules	40	40	40	40
Inputs scaling	[−2.5, 2.5]	[−2.5, 2.5]	[−2.5, 2.5]	[−2.5, 2.5]
Min. improvement (%)	1	1	10	10
Evolution of networks				
Replacement rate (%)	2	2	2	2
Mutation rate (%)	5	5	5	5
Objectives	Nodule fitness	Nodule fitness	Nodule fitness	Nodule fitness
	Performance	Performance	Performance	Performance
	Network diversity	–	–	–
Distance measure	Net funct. diver.	Obj. comparison	Obj. comparison	Obj. comparison
δ_{share}	0.1	–	–	–
Evolution of nodules				
Elitism (%)	60	50	50	50
Mutation rates (P_1, P_2, P_3) (%)	25, 25, 25	0, 60, 100	10, 25, 25	0, 60, 100
Simulated annealing	$T_0 = 1.0$, $\alpha = 0.95$, $n = 10$	$T_0 = 1.0$, $\alpha = 0.95$, $n = 100$	$T_0 = 1.0$, $\alpha = 0.95$, $n = 10$	$T_0 = 1.0$, $\alpha = 0.95$, $n = 100$
Objectives	Substitution	Substitution	Substitution	Substitution
	Difference	Difference	Difference	Difference
	All networks	All networks	All networks	All networks
	Count	Count	Count	Count
	Nodes	Nodes	Nodes	Nodes
	Connections	–	Connections	–
Distance measure	Funct. diversity	Euclidean distance	Funct. diversity	Euclidean distance
δ_{share}	1.0	1.3	1.3	1.3
Elitist mutation rate (%)	10	0	10	0

In order to test the performance of MOBNET in solving the proposed problems, we have compared it with the following classification algorithms:

1. A linear classifier using the variables selected by a stepwise discriminant analysis based on Wilk's lambda criterion (Dillon & Goldstein, 1984) with a tolerance level of 0.001. This model is the one that obtained the best results of all the classical statistical models applied as it is the optimal Bayesian classifier (Anderson, 1984) if the classes are normally distributed with common covariance matrixes. The experiments were made using the statistical package SPSS (SPSS, 1999).
2. The C4.5 (Quinlan, 1993) algorithm, using the code implemented by the author of the algorithm, which uses a decision tree.
3. A two-layer perceptron trained with the `quickprop` (Sejnowski, Hinton, & Touretzky, 1988) algorithm, using cross-validation and early-stopping (Finnoff, Hergert, & Zimmermann, 1993). The experiments were made using the simulator Nevada Backpropagation (Carlson, 1996).
4. A modular neural network using the adaptive mixture of local experts model. This is interesting as our model can be considered modular in many of its features. For this model we also used cross-validation and early-stopping. The experiments were made using the simulator NeuralWorks Professional II Plus (NeuralWare, 1993).
5. MOBNET using just one nodule in order to test that the cooperation among the nodules actually improves the performance of the isolated nodules.

The parameters of all of the above algorithms were optimized for each problem by a trial process. The source code of MOBNET in C is available by anonymous ftp from ftp://ftp.ayrna.org/pub/Research/Source

The errors achieved solving these problems using COVNET and the most important results reported in the bibliography can be found in (García-Pedrajas et al., submitted).

5.1. Pima 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.

Table 3
Error rates for Pima data set. For all the models and every permutation of the data sets, the table shows the training and generalization averaged error and standard deviation, and the best and the worst individuals

Model	Set	Training				Generalization			
		Mean	SD	Best	Worst	Mean	SD	Best	Worst
MOBNET	I	0.2220	0.0053	0.2170	0.2326	0.2042	0.0164	0.1771	0.2344
	II	0.2198	0.0075	0.2101	0.2326	0.1979	0.0174	0.1719	0.2292
	III	0.2226	0.0046	0.2135	0.2326	0.1932	0.0189	0.1615	0.2188
	All	0.2215	0.0058			*0.1984*	0.0176		
Linear classifier	I	0.2400	–	–	–	0.2190	–	–	–
	II	0.2380	–	–	–	0.2030	–	–	–
	III	0.2480	–	–	–	0.1980	–	–	–
	All	0.2420	–			*0.2067*	–		
C4.5	I	0.1718	–	–	–	0.2552	–	–	–
	II	0.1545	–	–	–	0.2552	–	–	–
	III	0.1441	–	–	–	0.2396	–	–	–
	All	0.1568	–			*0.2500*	–		
`quickprop`	I	0.2313	0.0069	0.2222	0.2413	0.2219	0.0189	0.1979	0.2604
	II	0.2174	0.0117	0.1979	0.2413	0.2313	0.0133	0.2153	0.2604
	III	0.2319	0.0025	0.2274	0.2361	0.1938	0.0084	0.1875	0.2135
	All	0.2269	0.0103			*0.2156*	0.0212		
Adaptive mixture of experts	I	0.2528	0.0135	0.2413	0.2847	0.2485	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		
MOBNET (1 nodule)	I	0.2422	0.0273	0.2205	0.3160	0.2422	0.0438	0.2031	0.3542
	II	0.2328	0.0112	0.2118	0.2500	0.2047	0.0168	0.1823	0.2448
	III	0.2462	0.0298	0.2240	0.3212	0.2135	0.0364	0.1771	0.3021
	All	0.2404	0.0240			*0.2201*	0.0369		

There are eight attributes for each pattern, all of them are continuous. Former results can be found in Smith, Everhart, Dickson, Knowler, and Johannes (1988). Following this previous work and the recommendations of Prechelt (1994, 1996) we have divided the data set into 576 patterns for training, and 192 patterns for generalization. The data set contains 500 instances of class 1 and 268 instances of class 2.

The parameters of MOBNET for this data set are shown in Table 2.

Error rates for all models are shown in Table 3. The results show that the generalization error of MOBNET is better than the other neural network models. This difference is statistically significant with a significance level of $\alpha = 0.05$, as the t-test shows (Table 4). MOBNET performs better in average than the two classical methods, the linear classifier and the C4.5 algorithm. It is also interesting the fact that the standard deviation of the errors of MOBNET is smaller than the deviation of the other models.

The networks evolved with MOBNET are quite compact as Table 5 shows. The networks are clearly smaller than the obtained using a MLP. It is specially interesting the fact that the connections are sparse, as this feature makes the hardware implementation easier.

Table 4
p-values of statistical tests over Pima results. First a Kolmogorov–Smirnov test is carried out, assuring that the errors are normally distributed. Then, an F test is made to test if the variances of the errors of the two experiments are equal. Finally, a t test tests if there are significant differences in the means of the errors from the two experiments. We use a level of significance of $\alpha = 0.05$ for accepting the alternative hypothesis

Test	MOBNET	`quickprop`	Mixture	MOBNET (1 nodule)
K–S test	0.8370	0.6767	0.7440	0.0959
F test	–	0.3170	0.0196	0.0001
t test	–	0.0012	0.0000	0.0058

Table 5
Network sizes for the four problems. The table shows the averaged number of nodes and connections of the networks

Problem	Model					
	MOBNET		`quickprop`		Mixture	
	Nodes	Conns.	Nodes	Conns.	Nodes	Conns.
Pima	6.17	29.43	24	256	17	180
Heart	11.40	64.63	16	266	25	360
Card	20.17	220.30	16	950	14	597
Glass	14.87	132.10	24	360	–	–

Table 6
Error rates for Heart data set. For all the models and every permutation of the data sets, the table shows the training and generalization averaged error and standard deviation, and the best and the worst individuals

Model	Set	Training				Generalization			
		Mean	SD	Best	Worst	Mean	SD	Best	Worst
MOBNET	I	0.0728	0.0119	0.0545	0.0990	0.1471	0.0208	0.1176	0.1912
	II	0.1020	0.0097	0.0891	0.1188	0.1088	0.0232	0.0735	0.1471
	III	0.0856	0.0114	0.0693	0.1040	0.1529	0.0158	0.1324	0.1765
	All	0.0868	0.0162			*0.1363*	0.0278		
Linear classifier	I	0.1490	–	–	–	0.1470	–	–	–
	II	0.1530	–	–	–	0.0880	–	–	–
	III	0.1460	–	–	–	0.1620	–	–	–
	All	0.1493	–	–	–	*0.1323*	–	–	–
C4.5	I	0.0644	–	–	–	0.2206	–	–	–
	II	0.0644	–	–	–	0.2059	–	–	–
	III	0.0594	–	–	–	0.2059	–	–	–
	All	0.0627	–			*0.2108*	–		
`quickprop`	I	0.0975	0.0226	0.0495	0.1188	0.1471	0.0196	0.1176	0.1765
	II	0.1025	0.0140	0.0842	0.1238	0.1544	0.0199	0.1324	0.1912
	III	0.0896	0.0129	0.0644	0.0990	0.1662	0.0241	0.1176	0.1912
	All	0.0965	0.0173			*0.1559*	0.0221		
Adaptive mixture of experts	I	0.0500	0.0135	0.0347	0.0743	0.1853	0.0210	0.1618	0.2206
	II	0.0490	0.0137	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		
MOBNET (1 nodule)	I	0.1554	0.0266	0.1337	0.2030	0.1853	0.0319	0.1324	0.2206
	II	0.1733	0.0175	0.1436	0.2030	0.1456	0.0401	0.0735	0.2206
	III	0.1168	0.0322	0.1287	0.2178	0.2015	0.0679	0.1471	0.3676
	All	0.1734	0.0263			*0.1775*	0.0531		

5.2. Heart data set

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 extracted from a larger set of 75, which correspond to the results of various medical tests carried out on many patients. 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 Detrano et al. (1989)).

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 data set has 270 patterns: 202 patterns were selected for training, and the remaining 68 were reserved for testing the generalization ability of the networks. The parameters of MOBNET for this data set are shown in Table 2.

Error rates for the four models are shown in Table 6. The results show that the generalization error of MOBNET is better than the errors obtained with MLP and the adaptive mixture of experts. MOBNET also outperforms the C4.5 algorithm. However, the linear classifier obtains the bests results in average over the three partitions, nevertheless, the results of the linear classifier and MOBNET are almost the same.

As in the Pima data set, the sizes of the networks evolved by MOBNET (see Table 5) are considerably smaller than the sizes of the MLP and modular networks.

Statistical tests in Table 7 show that Mobnet performs better than `quickprop` and the mixture of experts with a significance level of $\alpha = 0.05$. It shows also that the combination of several nodules produces smaller errors than an isolated nodule.

Table 7
p-values of statistical tests over Heart results

Test	MOBNET	`quickprop`	Mixture	MOBNET (1 nodule)
K-S test	0.5025	0.4418	0.3502	0.6792
F test	–	0.2162	0.5661	0.0008
t test	–	0.0037	0.0000	0.0005

Table 8
Error rates for Card data set. For all the models and every permutation of the data sets, the table shows the training and generalization averaged error and standard deviation, and the best and the worst individuals

Model	Set	Training				Generalization			
		Mean	SD	Best	Worst	Mean	SD	Best	Worst
MOBNET	I	0.1270	0.0043	0.1197	0.1332	0.1192	0.0110	0.1037	0.1453
	II	0.1344	0.0070	0.1236	0.1467	0.1029	0.0073	0.0930	0.1163
	III	0.1290	0.0060	0.1216	0.1409	0.1192	0.0074	0.1047	0.1279
	All	0.1301	0.0065			*0.1138*	0.0115		
Linear classifier	I	0.1370	–	–	–	0.1160	–	–	–
	II	0.1310	–	–	–	0.1100	–	–	–
	III	0.1410	–	–	–	0.1160	–	–	–
	All	0.1363	–	–	–	*0.1140*	–	–	–
C4.5	I	0.0830	–	–	–	0.1337	–	–	–
	II	0.1139	–	–	–	0.1279	–	–	–
	III	0.0869	–	–	–	0.1569	–	–	–
	All	0.0946	–			*0.1395*	–		
`quickprop`	I	0.0861	0.0112	0.0618	0.1004	0.1407	0.0094	0.1221	0.1512
	II	0.0770	0.0166	0.0541	0.1100	0.1093	0.0145	0.0872	0.1337
	III	0.0836	0.0146	0.0637	0.1100	0.1320	0.0110	0.1163	0.1512
	All	0.0822	0.0143			*0.1273*	0.0176		
Adaptive mixture of experts	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		
MOBNET (1 nodule)	I	0.1575	0.0331	0.1409	0.2510	0.1372	0.0453	0.1105	0.2626
	II	0.1681	0.0361	0.1448	0.2471	0.1390	0.0513	0.1047	0.2384
	III	0.1793	0.0492	0.1429	0.2703	0.1715	0.0710	0.1105	0.2907
	All	0.1683	0.0397			*0.1492*	0.0572		

5.3. Credit card data set

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 data set used for training the networks had 51 inputs and 2 outputs. As usual we divided the set into two parts: 518 patterns for training, and 172 for testing the generalization.

The parameters of MOBNET for this data set are shown in Table 2.

Error rates for both models are shown in Table 8. The results show that the generalization error of MOBNET is better than the error obtained with MLP and modular networks. Nevertheless, the performance of Mobnet and the linear classifier are very similar, being the performance of the C4.5 algorithm worse.

As in previous problems the networks evolved with MOBNET are far smaller than the MLP and modular networks.

Statistical tests shows the significance of the differences in the means of the errors (see Table 9).

Table 9
p-values of statistical tests over Card results

Test	MOBNET	`quickprop`	Mixture	MOBNET (1 nodule)
K–S test	0.4763	0.7638	0.6407	0.0087
F test	–	0.0246	0.6764	–
t test	–	0.0009	0.0000	–

5.4. Glass data set

This data set is also from the UCI Machine Learning Repository. The set contains data from six different types of glass. The study of classification of types of glass was motivated by criminal investigation, the glass left at the scene of the crime can be used as evidence if it is correctly classified. There are six classes corresponding to six types of glasses: glasses from building windows float and non-float

Table 10
Error rates for Glass data set. For all the models and every permutation of the data sets, the table shows the training and generalization averaged error and standard deviation, and the best and the worst individuals

Model	Set	Training				Generalization			
		Mean	SD	Best	Worst	Mean	SD	Best	Worst
MOBNET	I	0.2553	0.0218	0.2236	0.2981	0.2962	0.0309	0.2453	0.3396
	II	0.2571	0.0183	0.2236	0.2795	0.3528	0.0236	0.3019	0.3962
	III	0.2371	0.0264	0.1925	0.2733	0.4057	0.0438	0.3585	0.4717
	All	0.2480	0.0246			*0.3516*	0.0559		
Linear classifier	I	0.3580	–	–	–	0.3076	–	–	–
	II	0.3665	–	–	–	0.3396	–	–	–
	III	0.3975	–	–	–	0.4905	–	–	–
	All	0.3740	–			*0.3792*	–		
C4.5	I	0.1056	–	–	–	0.3208	–	–	–
	II	0.1056	–	–	–	0.4151	–	–	–
	III	0.0559	–	–	–	0.3774	–	–	–
	All	0.0890	–			*0.3711*	–		
`quickprop`	I	0.1579	0.0149	0.1402	0.1869	0.3642	0.0427	0.3019	0.4528
	II	0.1131	0.0676	0.0654	0.2617	0.4264	0.0472	0.3585	0.5283
	III	0.1140	0.0548	0.0374	0.1682	0.4170	0.0301	0.3774	0.4528
	All	0.1283	0.0536			*0.4025*	0.0481		
MOBNET (1 nodule)	I	0.3714	0.0495	0.3230	0.4596	0.4151	0.0881	0.3208	0.5849
	II	0.4112	0.0723	0.3478	0.5404	0.4396	0.0999	0.3208	0.5849
	III	0.3981	0.0593	0.3416	0.4907	0.4830	0.0740	0.3962	0.6038
	All	0.3936	0.0613			*0.4459*	0.0895		

processed, glasses from vehicle windows float processed, glasses from containers, glasses from tableware, and glasses from headlamps. Each record has nine continuous attributes meaning the weight percentage of different chemical components.

This data set is very difficult to classify due to two important features. First, the number of available patterns is low (214) for six different classes. Second, the number of patterns in each class is very unbalanced, ranging from 76 (building windows non-float processed) to 9 (tableware).

The parameters of MOBNET for this data set are shown in Table 2.

Error rates for both models are shown in Table 10. For this problem the modular network was not able to achieve useful results and for that reason it is not included in the table. For this problem, MOBNET considerably outperforms the linear classifier. This result is very interesting, as this problem is more complex, classical statistical methods find it more difficult to solve. The linear classifier assumes that the classes are normally distributed and have common covariance matrixes, these two hypotheses are more restrictive when there are more classes and the number of prototypes on each class are more unbalanced. In this kind of problems is where the application of a more complex model as MOBNET is more interesting.

Table 11
p-values of statistical tests over Glass results

Test	MOBNET	`quickprop`	MOBNET (1 nodule)
K–S test	0.5026	0.8250	0.5629
F test	–	0.4221	0.0136
t test	–	0.0000	0.0004

The same considerations about networks sizes (Table 5) and statistical significance (Table 11) of the difference of the errors of the previous problems can be applied to this set.

5.5. Over-training analysis

In this section we will briefly study the effect of the over-training on our model. In a first stage we made the experiments using a separate validation set and early-stopping. The results were worse than the obtained adding this validation set to the training set for all the four problems.

In order to measure this effect when the validation set was not used, we represent in Fig. 5 the behavior along the evolutionary process of three representative errors: (i) the averaged training error of the population (this value is the one used for stopping the evolution), (ii) the averaged generalization error of the population, and (iii) the generalization error of the best network.[8] The over-training effect must be considered in the third value represented, as it is the generalization error of the network that would be

[8] This network is selected following the defined criterion and would be the network selected as the result of the evolution.

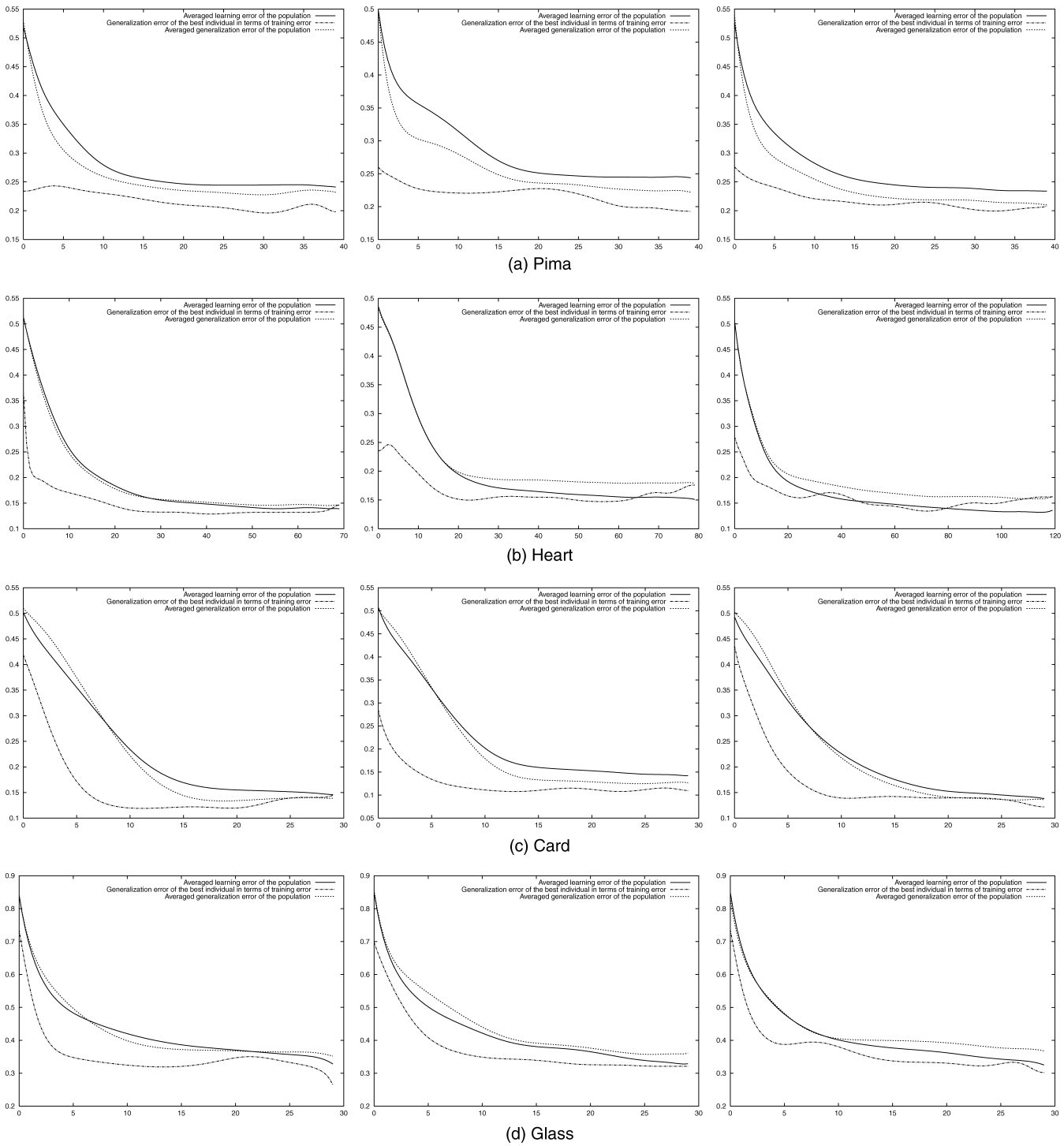

Fig. 5. Training and testing error of the population of networks along the evolutionary process in the classification of the four data set. For each problem the first three experiments of the first partition are shown.

selected as the result of the evolution. The Figure shows that the over-training effect did not appear in almost any of the experiments and it is almost negligible when it occurred.

Our explanation of this absence of over-training is based on the training errors achieved by MOBNET. As it is shown in all the error tables, the learning error is always worse in MOBNET than in the rest of the models. As its fitness is not centered only on performance, the model learns the overall features of the patterns and is less sensitive to the noise of the data.

6. Conclusions

In this work we have shown how adding several

objectives to fitness estimation of the subcomponents that form an individual in cooperative coevolutionary computation could improve the credit assignment to those subcomponents, and subsequently improve the evolutionary process. Once many objectives have been defined, multi-objective optimization techniques are the most natural way to obtain such fitness estimation.

The definition of as many objectives as could be interesting for a given problem allows the introduction of any a priori knowledge in the evolution process. This a priori knowledge can improve the performance of the networks evolved in many problems. For instance, in our experiments we have reinforced the regularization objectives in such problems where previous papers have shown that smaller networks performed better than bigger ones.

This model has been tested against several widely used methods of classification. MOBNET has shown to perform better than all these methods in three out four test problems. In the other one, it has outperformed all the methods except the linear classifier. It is also important to note the great improvement of MOBNET over the results obtained with other methods of neural networks, as MLP and modular networks, obtained with smaller networks.

Another important aspect of our model is that it could be applied to other environments of coevolution as well. Any tasks where several objectives could be defined to be fulfilled by the subcomponents is a potential field for introducing our multi-objective model for evaluating the fitness. At present, we are working on extending this model to a more general environment for cooperative coevolution for any given problem.

The absence of any significant over-training effect allows the addition of the pattern of the validation set to the training set, greatly improving the learning capabilities of the model. We think that the absence of this effect comes from the fact that the cooperative structure of our model drives every population to the most important features of the data; any nodule that learns the noise of the data will receive a poor evaluation and will be eventually lost during the evolutionary process.

MOBNET produces networks that are a little bigger than the one obtained by methods that use a regularization term (as COVNET García-Pedrajas et al. (submitted)). In MOBNET the complexity restrictions are introduced as another objective, and the effect on the evolution is a lower penalty on big networks. Nevertheless, that avoids the negative effect of the regularization term over the evolution when small networks have a poor performance in solving a given problem.

Acknowledgements

The authors would like to acknowledge R. Moya-Sánchez for her helping in the final version of this paper. This work has been supported in part by the Project ALI98-0676-CO2-02 of the Spanish Comisión Interministerial de Ciencia y Tecnología.

References

Anderson, T. W. (1984). *An introduction to multivariate statistical analysis* (2nd ed). New York: Wiley.

Angeline, P. J., Saunders, G. M., & Pollack, J. B. (1994). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, *5*(1), 54–65.

Baum, E. B. (1991). Book review on neural network design and the complexity of learning. *IEEE Transactions on Neural Networks*, *2*, 181–182.

Baum, E. B., & Haussler, D. (1989). What net size gives valid generalization? *Neural Computation*, *1*, 151–160.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. New York, NY: Oxford University Press.

Bebis, G., Georgiopoulos, M., & Kasparis, T. (1997). Coupling weight elimination with genetic algorithms to reduce network size and preserve generalization. *Neurocomputing*, *17*, 167–194.

Belew, R. K., McInerney, J., & Schraudolph, N. N. (1991). *Evolving networks: Using genetic algorithms with connectionist learning* (Technical Report No. CS90-174). San Diego: Department of Computer Science Engineering, University of California.

Blum, A. L., & Rivest, R. L. (1992). Training a 3-node neural network is NP-complete. *Neural Networks*, *5*, 117–127.

Caelli, T., Guan, L., & Wen, W. (1999). Modularity in neural computing. *Proceedings of the IEEE*, *87*(9), 1497–1518.

Caelli, T., Squire, D., & Wild, T. (1993). Model-based neural network. *Neural Networks*, *6*(2), 613–625.

Carlson, D. (1996). *Nevprop 3 ©user manual*. Reno, NE.

Cho, S.-B., & Shimohara, K. (1998). Evolutionary learning of modular neural networks with genetic programming. *Applied Intelligence*, *9*, 191–200.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *2*, 303–314.

Deb, K. (1999). Evolutionary algorithms for multi-criterion optimization in engineering design. *Proceedings of Evolutionary Algorithms in Engineering and Computer Science (EUROGEN'99)* (pp. 135–161). Jyväskylä, Finland.

Deb, K., & Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer (Ed.), (pp. 42–50). *Proceedings of the Third International Conference on Genetic Algorithms*, Los Altos, CA: Morgan Kaufmann.

Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S., & Froelicher, V. (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. *American Journal of Cardiology*, *64*, 304–310.

Dillon, W. R., & Goldstein, M. (1984). *Multivariate analysis: Methods and applications*. New York: Wiley.

Ficici, S. G., & Pollack, J. B. (2001). Pareto optimality in coevolutionary learning. In J. Kelemen, & P. Sosík (Eds.), (pp. 316–325). *Sixth European Conference on Artificial Life, ECAL 2001*, Prague, Czech Republic.

Finnoff, W., Hergert, F., & Zimmermann, H. G. (1993). Improving model selection by nonconvergent methods. *Neural Networks*, *6*, 771–783.

Fonseca, C. M., & Flemming, P. J. (1993). Genetic algorithms for multi-objective optimization: Formulation, discussion, and generalization. In S. Forrest (Ed.), (pp. 416–423). *Proceedings of the Fifth International Conference on Genetic Algorithms*, Urbana, IL: Morgan Kauffman.

Fonseca, C. M., & Flemming, P. J. (1995). An overview of evolutionary

algorithms in multi-objective optimization. *Evolutionary Computation*, *3*(1), 1–16.

García-Pedrajas, N., Hervás-Martínez, C., & Muñoz-Pérez, J. (2001a). Symbiont: A cooperative evolutionary model for evolving artificial neural networks for classification. In B. Bouchon-Meunier, J. Gutiérrez-Ríos, L. Magdalena, & R. R. Yager (Eds.), (*Vol. 2*) (pp. 341–354). *Technologies for constructing intelligent systems*, Berlin: Springer.

García-Pedrajas, N., Hervás-Martínez, C., & Muzñoz-Pérez, J. (submitted for publication). COVNET: a cooperative coevolutionary model for evolving artificial neural networks. *IEEE Transactions on Neural Networks*.

Giordana, A., & Neri, F. (1996). Search-intensive concept induction. *Evolutionary Computation*, *3*(4), 375–416.

Girosi, F., Jones, M., & Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural Computation*, *7*, 219–269.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. *Proceedings of the Second International Conference on Genetic Algorithms* (pp. 148–154). San Mateo, CA: Morgan Kaufmann.

Goutte, C., & Hansen, L. K. (1997). Regularization with a pruning prior. *Neural Networks*, *10*(6), 1053–1059.

Guan, L., Anderson, J. A., & Sutton, J. P. (1997). A network of networks processing model for image regularization. *IEEE Transactions on Neural Networks*, *8*(1), 169–174.

Hancock, P. J. B. (1992). Genetic algorithms and permutation problems: A comparison of recombination operators for neural net structure specification. In D. Whitley, & J. D. Schaffer (Eds.), (pp. 108–122). *Proceedings of the International Workshop of Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*, Los Alamitos, CA: IEEE Computer Society Press.

Hansen, L. K., & Pedersen, M. W. (1994). Controlled growth of cascade-correlation nets. *Proceedings of the International Conference on Artificial Neural Networks* (Vol. 1, pp. 797–800). Sorrento, Italy.

Haykin, S. (1994). *Neural networks—A comprehensive foundation*. New York, NY: Macmilan.

Haynes, T. D., & Sen, S. (1997). Co-adaptation in a team. *International Journal of Computational Intelligence and Organizations (IJCIO)*, *1*(4).

Hillis, W. D. (1991). Co-evolving parasites improves simulated evolution as an optimization technique. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), (*Vol. ii*) (pp. 313–384). *Artificial life*, Reading, MA: Addison-Wesley.

Horn, J., Goldberg, D. E., & Deb, K. (1994). Implicit niching in a learning classifier system: natures's way. *Evolutionary Computation*, *2*(1), 37–66.

Hornik, K., & Stinchombe, M. (1992). Multilayer feed-forward networks are universal approximators. In White H. et al. (Eds.), *Artificial neural networks: Approximation and learning theory*. Oxford: Blackwell Science.

Jolliffe, I. T. (1986). *Principal components analysis*. New York, NY: Springer.

Judd, J. S. (1988). On the complexity of loading shallow neural networks. *Journal of Complexity*, *4*, 177–192.

Judd, J. S. (1990). *Neural network design and the complexity of learning*. Cambridge, MA: MIT Press.

Juillé, H., & Pollack, J. B. (1996). Coevolving intertwined spirals. *Proceedings of the Fifth Annual Conference on Evolutionary Programming* (pp. 461–468). Cambridge, MA: MIT Press.

Kirkpatrick, S., Jr., C, D. G., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Koza, J. R. (1994). *Genetic programming II. Automatic discovery of reusable programs*. Cambridge, MA: MIT Press.

Lin, S.-H., Kung, S. Y., & Lin, L. J. (1997). Face recognition/detection by probabilistic decision-based neural networks. *IEEE Transactions on Neural Networks*, *8*(1), 114–132.

Liu, Y., & Yao, X. (1999). Ensemble learning via negative correlation. *Neural Networks*, *12*(10), 1399–1404.

Liu, Y., Yao, X., & Higuchi, T. (2000). Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, *4*(4), 380–387.

Liu, Y., Yao, X., Zhao, Q., & Higuchi, T. (2001). Evolving a cooperative population of neural networks by minimizing mutual information. *Proceedings of the 2001 IEEE Congress on Evolutionary Computation* (pp. 384–389). Seoul, Korea.

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs*. New York: Springer.

Moriarty, D. E. (1997). *Symbiotic evolution of neural networks in sequential decision tasks*. Unpublished doctoral dissertation, University of Texas at Austin (Report AI97-257).

Moriarty, D. E., & Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, *22*, 11–32.

Moriarty, D. E., & Miikkulainen, R. (1998). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, *4*(5).

NeuralWare (1993). *Neural computing: A technology handbook for professional II/Plus*. Pittsburgh, PA.

Paredis, J. (1995). Coevolutionary computation. *Artificial Life*, *2*, 355–375.

Potter, M. A., & Jong, K. A. de. (2000). Cooperative coevolution: an architecture for evolving coadapted subcomponents. *Evolutionary Computation*, *8*(1), 1–29.

Prechelt, L. (1994). *Proben1—A set of neural network benchmark problems and benchmarking rules* (Technical Report No. 21/94). Karlsruhe, Germany: Fakultät für Informatik, Universität Karlsruhe.

Prechelt, L. (1996). A quantitative study of experimental evaluation of neural network learning algorithms. *Neural Networks*, *9*, 457–462.

Puppala, N., Sen, S., & Gordin, M. (1998). Shared memory based cooperative coevolution. *Proceedings of the Fifth IEEE International Conference on Evolutionary Computation* (pp. 570–574). IEEE Publishing.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo: Morgan Kaufmann.

Rosenberg, R. S. (1967). *Simulation of genetic populations with biochemical properties*. Unpublished doctoral dissertation, University of Michigan.

Rosin, C. D., & Belew, R. K. (1997). New methods for competitive coevolution. *Evolutionary Computation*, *5*(1), 1–29.

Schaffer, J. D. (1984). *Some experiments in machine learning using vector evaluated genetic algorithm*. Unpublished doctoral dissertation, Nashville, TN: Vanderbilt University.

Schaffer, J. D., Whitley, L. D., & Eshelman, L. J. (1992). Combinations of genetic algorithms and neural networks: A survey of the state of the art. In L. D. Whitley, & J. D. Schaffer (Eds.), (pp. 1–37). *Proceedings of Cogann-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks*, Los Alamitos, CA: IEEE Computer Society Press.

Sejnowski, T. J., Hinton, G. E., & Touretzky, D. S. (Eds.), (1988). *Faster-learning variations on back-propagation: An empirical study*. San Mateo, CA: Morgan Kaufmann.

Shang, Y., & Wah, B. W. (1996). Global optimization for neural networks training. *IEEE Computer*, *29*(3), 45–54.

Sima, J. (1994). Loading deep networks is hard. *Neural Computation*, *6*(5), 842–850.

Smalz, R., & Conrad, M. (1994). Combining evolution with credit apportionment: A new learning algorithm for neural nets. *Neural Networks*, *7*(2), 341–351.

Smith, J. M. (1989). *Evolutionary genetics*. New York, NY: Oxford University Press.

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., & Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261–265). IEEE Computer Society Press.

SPSS, I. (1999). *SPSS ©9.0 advanced models*. Chicago, IL: SPSS Inc.

Srinivas, N., & Deb, K. (1994). Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation*, *2*(3), 221–248.

Steuer, R. E. (1986). *Multiple criteria optimization: Theory, computation, and application*. New York, NY: Wiley.

Syswerda, G. (1991). A study of reproduction in generational and steady-state genetic algorithms. In G. Rawlins (Ed.), *Foundations of genetic algorithms* (pp. 94–101). Los Altos, CA: Morgan Kaufmann.

Whitehead, B. A., & Choate, T. D. (1996). Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Transactions on Neural Networks*, *7*(4).

Whitley, D. (1989). The GENITOR algorithm and selective pressure. *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 116–121). Los Altos, CA: Morgan Kaufmann.

Whitley, D., & Kauth, J. (1988). GENITOR: a different genetic algorithm. *Proceedings of the Rocky Mountain Conference on Artificial Intelligence* (pp. 118–130). Denver, CO.

Whitley, D., & Starkweather, T. (1990). GENITOR II: a distributed genetic algorithm. *Journal of the Experimental Theoretical Artificial Intelligence*, 189–214.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, *9*(87), 1423–1447.

Yao, X., & Liu, Y. (1997). A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, *8*(3), 694–713.

Yao, X., & Shi, Y. (1995). A preliminary study on designing artificial neural networks using co-evolution. *Proceedings of the IEEE Singapore International Conference on Intelligent Control and Instrumentation* (pp. 149–154). Singapore.

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: methods and applications*. Unpublished doctoral dissertation, Eidgenössische Technische Hochschule Zürich.

Zitzler, E., & Thiele, L. (1998). Multi-objective optimization using evolutionary algorithms—a comparative case study. *Problem solving from nature—PPSN V* (pp. 292–301). Amsterdam, The Netherlands: Springer.