Cooperative Coevolution of Artificial Neural Network Ensembles for Pattern Classification

Nicolás García-Pedrajas, *Member, IEEE*, César Hervás-Martínez, *Member, IEEE*, and Domingo Ortiz-Boyer

***Abstract*—This paper presents a cooperative coevolutive approach for designing neural network ensembles. Cooperative coevolution is a recent paradigm in evolutionary computation that allows the effective modeling of cooperative environments. Although theoretically, a single neural network with a sufficient number of neurons in the hidden layer would suffice to solve any problem, in practice many real-world problems are too hard to construct the appropriate network that solve them. In such problems, neural network ensembles are a successful alternative.**

Nevertheless, the design of neural network ensembles is a complex task. In this paper, we propose a general framework for designing neural network ensembles by means of cooperative coevolution. The proposed model has two main objectives: first, the improvement of the combination of the trained individual networks; second, the cooperative evolution of such networks, encouraging collaboration among them, instead of a separate training of each network. In order to favor the cooperation of the networks, each network is evaluated throughout the evolutionary process using a multiobjective method. For each network, different objectives are defined, considering not only its performance in the given problem, but also its cooperation with the rest of the networks.

In addition, a population of ensembles is evolved, improving the combination of networks and obtaining subsets of networks to form ensembles that perform better than the combination of all the evolved networks.

The proposed model is applied to ten real-world classification problems of a very different nature from the UCI machine learning repository and `proben1` benchmark set. In all of them the performance of the model is better than the performance of standard ensembles in terms of generalization error. Moreover, the size of the obtained ensembles is also smaller.

***Index Terms*—Classification, cooperative coevolution, multiobjective optimization, neural network ensembles.**

I. Introduction

NEURAL network ensembles [1] are receiving increasing attention in recent neural network research, due to their interesting features. They are a powerful tool especially when facing complex problems. Network ensembles are usually made up of a linear combination of several networks that have been trained using the same data, although the actual sample used by each network to learn can be different. Each network within the ensemble has a potentially different weight in the output of the ensemble. Several works have shown [1] that the network ensemble has a generalization error generally smaller than that obtained with a single network and also that the variance of the ensemble is lesser than the variance of a single network. The output y of a typical ensemble [2] with k constituent networks when an input pattern $\mathbf{x}$ is presented is

$$y(\mathbf{x}) = \sum_{i=1}^{k} w_i y_i(\mathbf{x}) \tag{1}$$

where y_i is the output of network i and w_i is the weight associated to that network. If the networks have more than one output, a different weight is usually assigned to each output. The ensembles of neural networks have some of the advantages of large networks without their problems of long training time and risk of overfitting. For more detailed descriptions of ensembles the reader is referred to [3]–[7].

Although there is no clear distinction between the different kinds of multinet networks [2], [8]–[10], we follow the distinction of [11]. In an ensemble several redundant approximations to the same function are combined by some method, and in a modular system the task is decomposed into a number of simpler components. Nevertheless, our approach incorporates an implicit decomposition that is provided by the use of cooperative coevolution [12]–[14].

This combination of several networks that cooperate in solving a given task has other important advantages such as [11], [15] the following.

- They can perform more complex tasks than any of their subcomponents [16].
- They can make an overall system easier to understand and modify.
- They are more robust than a single network.

In most cases, neural networks in an ensemble are designed independently or sequentially, so the advantages of interaction and cooperation among the individual networks are not exploited. Earlier works separate the design and learning process of the individual networks from the combination of the trained networks. In this work, we propose a framework for designing ensembles, where the training and combination of the individual networks are carried out together, in order to get more cooperative networks and more effective combinations of them.

The new framework presented in this work for designing and evolving neural network ensembles uses and benefits from two different paradigms: cooperative coevolution and multiobjective optimization. The design of neural network ensembles implies making many decisions that have a major impact on the performance of the ensembles. The most important decisions that we must face when designing an ensemble are the following.

Manuscript received April 28, 2003; revised April 27, 2004. This work was supported in part by the Spanish Comisión Interministerial de Ciencia y Tecnología under Project TIC2002-04036-C05-02 and in part by FEDER funds.

The authors are with the Department of Computing and Numerical Analysis, University of Córdoba, Córdoba E-14071, Spain (e-mail: npedrajas@uco.es; chervas@uco.es; dortiz@uco.es).

Digital Object Identifier 10.1109/TEVC.2005.844158

- The method for designing and training the individual networks.
- The method of combining the individual networks, and the mechanism for obtaining individual weights for each network if such is the case.
- The measures of performance of the individual networks.
- The methods for encouraging diversity among the members of the ensembles and how to measure such diversity.
- The method of selection of patterns that are used by each network to learn.
- Whether to include regularization terms and their form.

Techniques using multiple models usually consist of two independent phases: model generation and model combination [6]. The disadvantage of this approach is that the combination is not considered during the generation of the models. With this approach the possible interactions among the trained networks cannot be exploited until the combination stage [15], and the benefits that can be obtained from this interactions during the learning stage are lost.

However, several researchers [17], [18] have recently shown that some information about cooperation is useful for obtaining better ensembles. This new approach opens a wide field where the design and training of the different networks must be interdependent.

In this paper, we present a new model for cooperatively evolving the individual networks and their combinations. We have three main aims in the design of our model.

1) Improving the combination of networks. Some recent works have shown [17], [19] that the combination of a subset of all the trained networks can be better than the combination of all the networks.
2) Improving the introduction of correction terms for discouraging correlation, reducing mutual information, or similar ideas, as has been suggested by several authors.
3) Improving the diversity among the networks of the ensemble.

The simultaneous evolution of all the networks has been shown to be useful in some recent papers. The most common approach is the modification of the error term in the back-propagation algorithm to take into account the correlation of the networks of the ensemble, e.g., [15] and [20]–[22]. Liu *et al.* [18] evolved the population of networks minimizing the mutual information [23] among the individual networks. Liu and Yao [21] modified the standard back-propagation algorithm adding a correction term that forces the networks to be negatively correlated. Nevertheless, these works are centered only on obtaining more diverse networks in the ensemble, and some recent results have shown that it is not clear that the use of a diversity term had a beneficial effect on the ensemble [24]. So, we have opted for considering diversity as one among many other interesting objectives.

Opitz and Shavlik [25] developed a model closer to cooperative coevolution. They evolved a population of networks by means of a genetic algorithm and combined the networks in an ensemble with a linear combination. Competition among the networks is encouraged with a diversity term added to the fitness of each network. More recently, Zhou *et al.* analyzed the relationship between the ensemble and its component neural networks [17]. This study revealed that it may be better to ensemble a subset of the neural networks instead of all of them. In order to select this subset of possibly better performing networks, they applied a genetic algorithm that evolved the weight of each network in the ensemble. Their results support our approach of evolving a population of ensembles, each one being a combination of some of the evolved networks. A recent work by Bakker and Heskes [19] corroborates the results of Zhou *et al.*

Moreover, the selection and training of the individual classifiers is thought to be an issue as critical as the combination method [26], [27]. Zhou *et al.* [17] have shown that a combination of some of the networks may be better than a combination of all the networks, and that a genetic algorithm [28] can be used for obtaining that subset of networks.

We propose a model that makes use of these ideas by means of the cooperative evolution of the networks that form the ensemble. Our model relies on two central ideas: the coevolution of different subpopulations of diverse networks and the evolution of the best combinations of these networks. Cooperative coevolution [12], [29] is a recent paradigm in the field of evolutionary computation that has shown a natural tendency to evolve diverse populations.

Our cooperative model is focused on improving the following two aspects of the design and training of an ensemble: the evolution of more cooperative networks and the combination of such networks. The use of cooperative coevolution allows us to obtain more diverse networks without introducing diversity terms that can bias the learning process and the improvement of the collaborative features of the networks. Cooperative coevolution also offers a framework for the combination of networks that has been proved useful in other models of neural networks, e.g., modular neural networks [30].

The second basic idea of our model is the introduction of multiobjective optimization in the evaluation of the fitness of the networks. The performance of the network is one of its most important aspects, but not the only interesting one. The evaluation of different objectives for each network allows a more accurate estimation of the goodness of a network. Additionally, the definition of many objectives allows the inclusion of some useful measures applied to other models, such as negative correlation [21] or mutual information [18]. Multiobjective evaluation of modular networks obtained good results in a previous work [31].

The multiobjective approach improves the following features of the design of the network ensembles.

- The measures of performance of the individual networks. We can evaluate the performance of the networks from different points of view.
- The methods for encouraging diversity among the members of ensembles and how to measure such diversity. We can estimate the diversity of networks with different measures.
- Whether to include regularization terms and their form. Instead of adding a regularization term [32] to the error function that may seriously bias the learning process, we

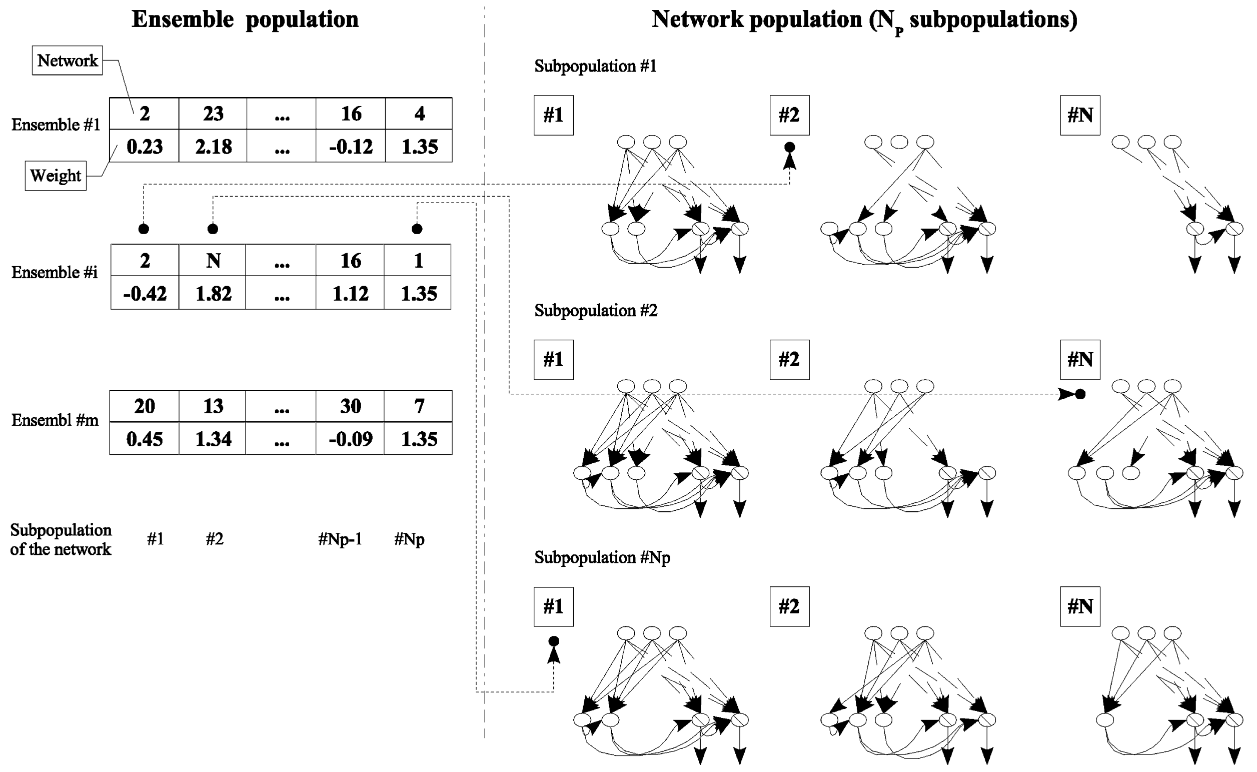

Fig. 1. Populations of ensembles and networks. Each element of the ensemble is a reference to an individual of the corresponding subpopulation of networks, together with its associated weight.

can add an objective of regularization that will encourage less complex networks without biasing the evolutionary process.

The rest of the paper is organized as follows. Section II describes the proposed model of cooperative ensembles. Sections III and IV state all the aspects of network and ensemble populations and their evolution. Section V explains the multiobjective evaluation of the individuals. Section VI describes the experimental setup and Section VII shows the results of the application of our model to ten real-world problems. A comparison with standard ensemble methods is carried out in Section VIII. Sections IX and X show a comprehensive analysis of several aspects of the evolved ensembles. Finally, Section XI states the conclusions of our work and the most important lines for future research.

II. Cooperative Ensemble of Neural Networks

Evolutionary computation [28], [33] is a set of global optimization techniques that have been widely used in the last few years for training and automatically designing neural networks. Some efforts have been made to design modular [34] neural networks with these techniques (e.g., [35]), but the design of network ensembles by means of evolutionary computation has only been focused on some of its aspects [21], [25], [36] and not on the whole process.

Cooperative coevolution [37] is a recent paradigm in the area of evolutionary computation, based on the evolution of coadapted subcomponents without external interaction. In cooperative coevolution a number of species are evolved together. Cooperation among individuals is encouraged by rewarding the individuals for their join effort to solve a target problem. The work in this paradigm has shown that cooperative coevolutionary models present many interesting features, such as specialization through genetic isolation, generalization and efficiency [29]. 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.

So, the cooperative coevolutionary model offers a very natural way for modeling the evolution of cooperative parts. This is the case of neural network ensembles, where the accuracy of the individual networks is not enough to assure a good performance. Cooperation among individual networks is also needed in order to improve the performance significantly.

Our cooperative model is based on two separate populations that evolve cooperatively. A model sharing some of these basic ideas has already been successfully applied to the evolution of modular neural networks [31]. These two populations are the following.

- *Population of networks*: This population consists of a number of independent subpopulations of networks. The independent evolution of subpopulations is an effective way of keeping the networks of different populations diverse. The absence of genetic material exchange among subpopulations also tends to produce more diverse networks whose combination is more effective. Every subpopulation is evolved using evolutionary programming.

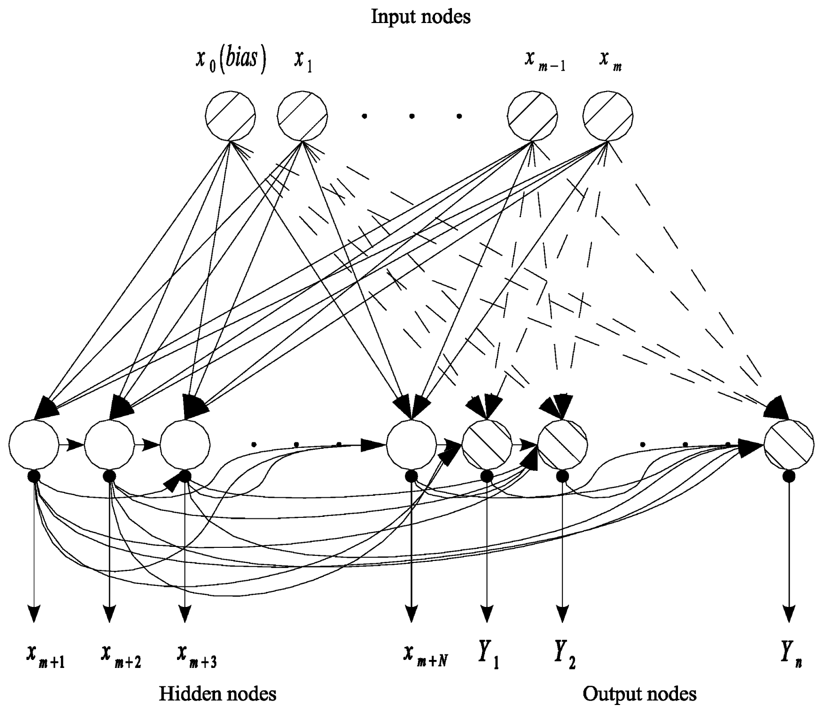

Fig. 2. Model of a GMLP.

- *Population of ensembles*: Each member of the population of ensembles is an ensemble formed by a network from every network subpopulation. Each network has an associated weight.

  The population of ensembles keeps track of the best combinations of networks, selecting the subsets of networks that are promising for the final ensemble.

The two populations evolve cooperatively. Each generation of the whole system consists of a generation of the network population followed by a generation of the ensemble population. The relationship between the two populations can be seen in Fig. 1.

The second basis of our model is the use of multiobjective optimization in the evaluation of the fitness of the individual networks. We have quoted previous works that agree that the learning process of the networks must take into account the cooperation among the networks for obtaining better ensembles. Implicit cooperation among the subpopulations in cooperative coevolution helps this learning process, but it is necessary to enforce cooperation to assure good results. The evaluation of several objectives for each network allows the model to encourage such cooperation, rewarding the networks not only for their performance in solving the given problem, but also for other aspects, such as whether they are different from other networks, whether they are useful in the ensembles or anything else considered relevant by the designer.

Additionally, every network is subject to back-propagation training throughout its evolution with a certain probability. In this way, the network is allowed to learn from the training set, but it is also prevented from being too similar to the rest of the networks by means of its evaluation using different objectives. The back-propagation algorithm is implemented as a mutation operator.

As we stated in Section I, many decisions must be made in order to design an ensemble of neural networks. In the next sections, we explain in depth all the aspects of our model and the decisions made, following the ideas we have already introduced.

III. Network Population

Our basic network is a generalized multilayer perceptron (GMLP), as defined in [38]. It consists of an input layer, an output layer, and a number of hidden nodes interconnected among them.

Given a GMLP with m inputs, N hidden nodes, and n outputs, and $\boldsymbol{X}$ and $\boldsymbol{Y}$ being the input and output vectors, respectively, it is defined by the equations [38]

$$\begin{aligned} x_i &= X_i, && 1 \le i \le m \\ h_i &= \sum_{j=1}^{i-1} w_{ij} x_j, && m < i \le m+N+n \\ x_j &= f(h_j), && m < j \le m+N+n \\ Y_i &= x_{i+m+N}, && 1 \le i \le n \end{aligned} \tag{2}$$

where w_{ij} is the weight of the connection from node j to node i. The representation of a GMLP can be seen in Fig. 2. We see that the ith node, provided it is not an input node, has connections from every jth node $j < i$.

The main advantage of using a GMLP is the parsimony of the evolved networks. Its structure allows the definition of very complex surfaces with fewer nodes than in a standard multilayer perceptron with one or two hidden layers.

The network population is formed by N_P subpopulations. Each subpopulation consists of a fixed number of networks codified directly as shown in Fig. 2. These networks are not fully connected. When a network is initialized, each connection is created with a given probability. The population is subject to operations of replication and mutation. Crossover is not used due to potential disadvantages [39] that it has for evolving neural networks. With these features the algorithm falls in the class of evolutionary programming [40].

A. Evolution of Networks

The algorithm for the evolution of the subpopulations of networks is similar to other models proposed in the literature, such as GNARL [39] or EPNet [35]. The steps for generating the new subpopulations are the following.

- Networks of the initial subpopulation are created randomly. The number of nodes of the network 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 new subpopulation is generated replicating the best $P\%$ of the previous subpopulation. The remaining $(1 - P)\%$ is removed and replaced by mutated copies of networks selected by roulette selection from the best $P\%$ individuals.
- There are two types of mutation: parametric and structural. The severity of structural mutation is determined by the relative fitness, F_r, of the network. Given a network ν, its relative fitness is defined as

$$F_r(\nu) = e^{-\alpha F(\nu)} \tag{3}$$

where $F(\nu)$ is the fitness value of network ν, and α is a parameter that must be chosen by the expert. In our experiments $\alpha = 0.25$.

Parametric mutation consists of the modification of the weights of the network without modifying its topology. Many parametric mutation operators have been suggested in the specific literature: random modification of the weights [12], simulated annealing [35], and back-propagation [35], among others. In this paper, we use the back-propagation algorithm [38] as mutation operator. This algorithm is performed for a few iterations with a low value of the learning coefficient η (in our experiments $\eta = 0.25$). 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 network. The behavioral link between parents and their offspring must be enforced to avoid generational gaps that produce inconsistency in the evolution [35], [39]. 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.
- *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 connections: from an input node, from another hidden node and to an output node. The selection of the type of connection to remove is made according to the relative number of each type of nodes: input, output and hidden. Otherwise, when there is a significant difference in the number of these three types, the number of connections of each type may end up highly biased.
- *Deletion of a connection*: A connection is selected, following the same criterion of the addition of connections, and removed.

TABLE I
PARAMETERS OF NETWORK 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

All of the above mutations can be made in a single mutation operation over the network. 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:

$$\Delta = \Delta_m + F_r(\nu)(\Delta_M - \Delta_m). \tag{4}$$

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 network mutation parameters used in all of our experiments are shown in Table I.

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

For the initialization of the weights of the networks, we used the method suggested by Le Cun, [2], [41]. The weights are obtained from a uniform distribution within the interval $[-3/\sqrt{m}, 3/\sqrt{m}]$, where m is the number of inputs to the network.

The whole evolutionary process for network i in a generation is illustrated in Fig. 3(a). The figure shows the possible evolution of a network during one generation of the evolutionary process.

IV. Ensemble Population

The ensemble population is formed by a fixed number of ensembles. Each ensemble is the combination of one network from each subpopulation of networks with an associated weight. The relationship between the two populations has been shown in Fig. 1. It is important to note that, as the chromosome that represents the ensemble is ordered, the permutation problem [39], that is so important in network evolution, cannot appear.

A. Evolution of Ensembles

The ensemble population is evolved using the *steady-state* genetic algorithm [42], [43]. It has been proved that this model shows higher variance [44] and is a more aggressive and selective selection strategy [45] than the standard genetic algorithm.

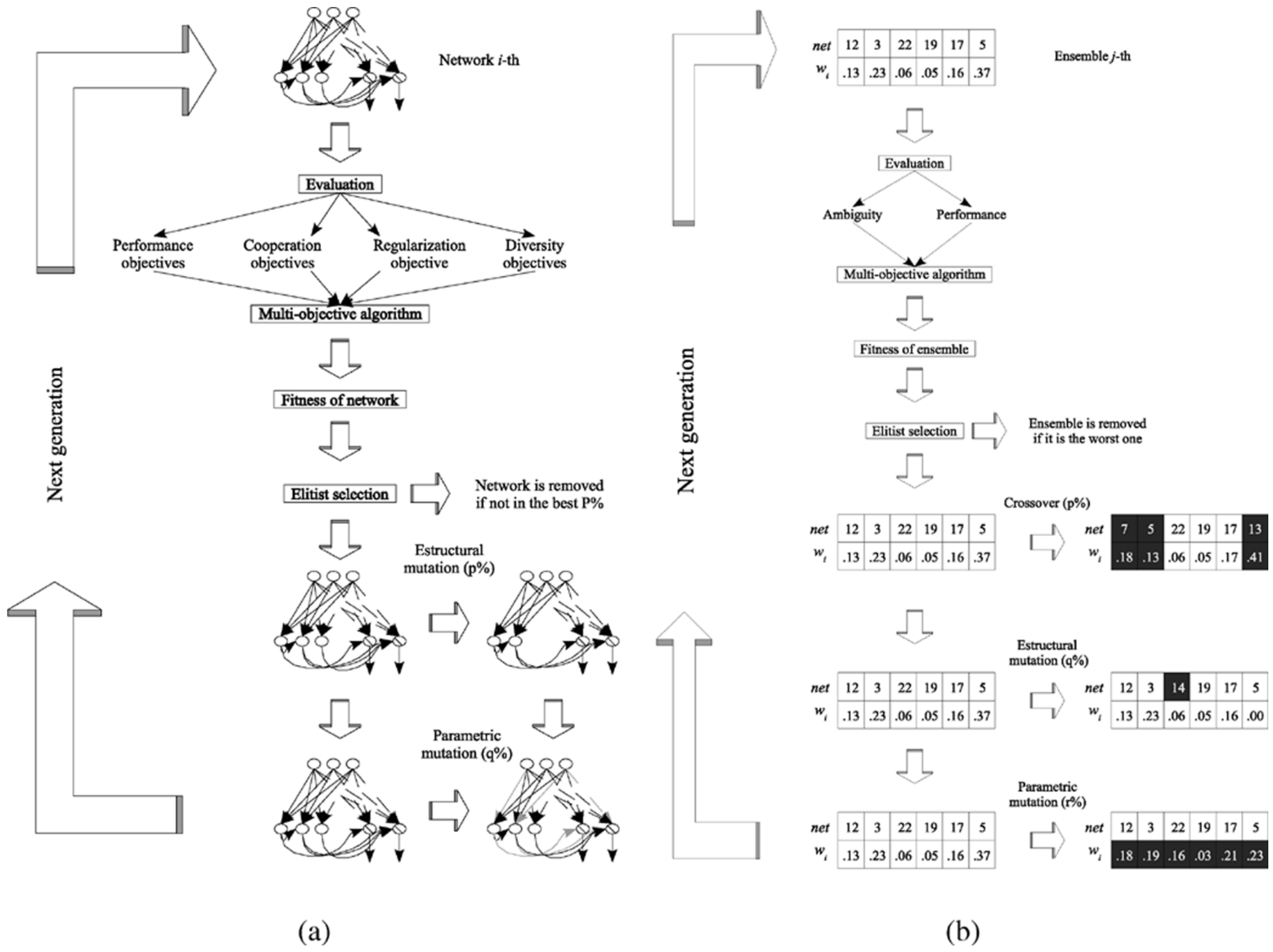

Fig. 3. (a) Evolutionary process of a network and (b) an ensemble in a generation showing the possible steps of the process.

This algorithm is selected due to the fact that we need a population of ensembles that evolves more slowly than the population of networks, as the changes in the population of ensembles have a major impact on the fitness of the networks. The steady-state genetic algorithm avoids the negative effect that this drastic modification of the population of ensembles may have over the subpopulations of networks. It has also been shown by some works in the area [46], [47] that the steady-state genetic algorithm produces better solutions than the standard genetic algorithm.

Crossover is made at network level, using a standard two-point crossover. So the parents exchange their networks to generate their offspring. Mutation is also carried out at network level.[1] When an ensemble is mutated, one of its networks is selected and is substituted by another network of the same subpopulation selected by means of a roulette algorithm.

The whole evolutionary process for ensemble jth in a generation is illustrated in Fig. 3(b). The figure shows the possible evolution of an ensemble during one generation of the evolutionary process.

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

B. Combination of Network Outputs

Basically, in a classification environment, there are three methods for combining the outputs of the networks [6]: *Majority voting*, *sum of the outputs of the networks*, and *winner takes all*.

The most commonly used methods for combining the networks are the *majority voting* and *sum of the outputs of the networks*, both of them with a weight vector that measures the confidence in the prediction of each network. Each network is assigned a weight α_i and the output of the ensemble is obtained using

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i f_i(\boldsymbol{x}) \tag{5}$$

[1]There is also a parametric mutation of the weights of the networks that is explained in Section IV-B.

where N is the number of networks that make up the ensemble, and $f_i(\boldsymbol{x})$ is the output of network i. The problem of obtaining the weight vector $\boldsymbol{\alpha}$ is not trivial. Usually, the values of the weights α_i are constrained $\sum_{i=1}^{N} \alpha_i = 1$ in order to help to produce estimators with lower prediction error [48], although justification of this constraint is just intuitive [49]. In this work, we use the sum of the outputs of the network with a weight vector.

The "basic ensemble method (BEM)," as it is called in [1], consists of weighting all the networks equally. So, having N networks, the output of the ensembles is

$$F(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{x}). \tag{6}$$

Perrone and Cooper [1] defined the *generalized ensemble method*, which is equivalent to the *mean square error – optimal linear combination* (MSE-OLC) without a constant term of Hashem [50], where the values of α_i are given by

$$\alpha_i = \frac{\sum\limits_j C_{ij}^{-1}}{\sum\limits_k \sum\limits_j C_{kj}^{-1}} \tag{7}$$

where C_{ij} is the symmetric correlation matrix $C_{ij} \equiv E[m_i(\boldsymbol{x})m_j(\boldsymbol{x})]$, where $m_k(\boldsymbol{x})$ defines the *misfit* of function k, that is the deviation from the true solution $f(\boldsymbol{x})$, $m_k(\boldsymbol{x}) \equiv f(\boldsymbol{x}) - f_k(\boldsymbol{x})$. Many other techniques have been proposed in the last few years, such as, linear regression [48], principal components analysis and least-square regression [51], correspondence analysis [6], and the use of a validation set [25].

In this work, we use a genetic algorithm for obtaining the weight of each component, and not only for selecting the most interesting subsets of networks. This approach is similar to the use of a gradient descent procedure [52], avoiding the problem of being trapped in local minima. The use of a genetic algorithm has an additional advantage over the optimal linear combination, as the former is not affected by the collinearity problem [1], [50].

Our method considers each ensemble as a chromosome and applies a standard genetic algorithm to optimize the weight of each network. The weight of each network of the ensemble is codified as a real number. The chromosome formed in this way is subject to standard two-point crossover and mutation. Mutation consists of a simulated annealing [53] algorithm.

C. Diversity

Diversity is one of the key aspects of network ensembles. The error of an ensemble can be decomposed by analysis of the ambiguity of the individual networks that make up the ensemble [54].

Let us assume that we have an ensemble of N networks, and the output of network ith on input $\boldsymbol{x}$ is $f_i(\boldsymbol{x})$. Each network has a weight α_i that measures the confidence on such network. The output of the ensemble $F(\boldsymbol{x})$ is defined by

$$F(\boldsymbol{x}) = \sum_i \alpha_i f_i(\boldsymbol{x}). \tag{8}$$

As we have stated in the description of standard ensembles, see Section IV-B, the weights usually have the restriction of being positive and sum to one.

The *ambiguity* of the network i with respect to the ensemble on input $\boldsymbol{x}$ is defined by $a_i(\boldsymbol{x}) = (f_i(\boldsymbol{x}) - F(\boldsymbol{x}))^2$. The *ensemble ambiguity* for input $\boldsymbol{x}$ is given by

$$a(\boldsymbol{x}) = \sum_i \alpha_i a_i(\boldsymbol{x}) = \sum_i \alpha_i (f_i(\boldsymbol{x}) - F(\boldsymbol{x}))^2. \tag{9}$$

The most interesting aspect of the ambiguity is that the global error of the ensemble E can be expressed in function of the error and ambiguity of each network that form the ensemble [54]

$$E = \bar{E} - \bar{A} \tag{10}$$

where $\bar{E} = \sum_i \alpha_i E_i$, and $\bar{A} = \sum_i \alpha_i A_i$, E_i being the error in the i network, and A_i the average ambiguity over the generalization set. This equation shows that the error can be decreased by improving the ambiguity of the individual networks, provided that the individual error of the networks is not increased. This can be explained as another form of bias/variance decomposition [55].

Thus, the objective of any method for developing network ensembles must be obtaining accurate networks as diverse as possible. In our model, diversity is assured by means of three different mechanisms.

1) Coevolution of genetically isolated subpopulation of networks: As there is no exchange of genetic material among the members of the different subpopulations, diversity among the subpopulations is preserved.
2) Fitness-sharing in the evaluation of the networks: When a network is evaluated, we use fitness-sharing for decreasing the fitness of the networks functionally close to each other.
3) Objectives of diversity: Additionally, each network is evaluated using one or more objectives of diversity. Again, diverse networks are rewarded. The evaluation of the ensembles also includes an objective rewarding the ensembles formed by diverse networks.

D. Pattern Sampling

One of the aspects of neural network ensemble design that has received a lot of attention in the literature is the topic of training data set sampling. Sampling methods have shown to be successful in improving the performance of different classifiers in artificial and real-world data sets [56]–[58].

These algorithms can be divided into two types: algorithms that adaptively change the distribution of the training set, based on the performance of the previous classifiers, and algorithms that do not adapt the distribution. Boosting methods are the most representative methods of the first group. The most widely used boosting methods are Ada-Boost [59] and Arcing-x4 [60]. All of them are based on adaptively increasing the probability of sampling the patterns that are not classified correctly by the previous classifiers.

Bagging [57] is the most representative algorithm of the second group. Bagging (after **B**ootstrap **agg**regat**ing**) just generates different bootstrap samples from the training set.

As we are developing a cooperative model of evolution, the methods that adaptively modify the probability of selecting a pattern are not easily incorporated to the evolution. In our model each network of every population receives a different bootstrap sample from the original training data set.

V. Multiobjective Evaluation of Network and Ensemble Fitness

If we approach the problem of evaluation of the fitness of networks and ensembles as a multiobjective optimization task, we will benefit from many advantages. First, there is no need to weight the different objectives, as would be the case using an aggregating approach. Second, the solutions based on Pareto optimality guarantee the diversity [61] of the final population. Third, there is an underlying theory applicable to our problem. In addition, it gives us a framework for adding as many objectives as may be of interest.

The next two sections describe the objectives that have been defined for networks and ensembles. These are only a subset of all the objectives that may be interesting for a given task. One of the advantages of our model is that it allows the introduction of any useful objective without modifying the general structure of the system.

A. Individual Networks Objectives

The objectives of the networks could be grouped into four sets: objectives of performance, objectives of regularization, objectives of cooperation and objectives of diversity. We have three objectives of performance, one objective of regularization, two objectives of cooperation, and four objectives of diversity. These objectives are the following.

1) Objectives of Performance: These objectives measure the performance of the network from three different complementary points of view. The objectives are the following.

- *Performance*: As we use bagging, this measure of performance is the number of patterns correctly classified by the network pondered by the weight of each pattern.
- *Shared performance*: This objective enforces the networks to classify different patterns [15]. In this way, the networks that are able to accurately classify patterns that are incorrectly classified by many ensembles are rewarded. Each pattern receives a value p_i that measures the number of ensembles that correctly classify the pattern, namely

$$p_i = \frac{m_{\text{ok}}}{M} \tag{11}$$

where m_{ok} is the number of ensembles that classify the pattern correctly, and M is the number of ensembles. The value assigned to a network for this objective is given by

$$P_{\text{sh}} = \sum_{i=1}^{P} e_i(1 - p_i) \tag{12}$$

where e_i is 1 if pattern i is correctly classified by the network, and 0, otherwise, and P is the number of patterns.

With this objective the networks that classify *difficult* patterns are rewarded, even if the total number of patterns correctly classified by the network is not high. This idea is similar to the theoretical basis of *boosting*, as boosting also encourages the learning of difficult patterns raising their probability of being sampled.

- *Ensembles*: Average performance of the ensembles where the network is present. In order to reward the best collaborating networks, this objective measures the average performance of the ensembles where the network participates. When a network does not participate in any ensemble, the objective cannot be calculated. In such a case the objective receives a value of 0.

2) Objectives of Regularization: In order to reward small networks, many measures may be included as regularization terms. These measures can be taken from network pruning [62], or regularization theory [32], [63]. Most authors use the weight decay term proposed in many papers [64], [65]

$$\sum_{i,j} |w_{ij}|. \tag{13}$$

Other authors propose a cost function of the form [62]

$$\sum_{i,j} (w_{ij})^2 \tag{14}$$

nevertheless, both measures have a strong impact on the evolution of the network due to the heavy constraint that is imposed on the weights. For that reason, we have used the following less restrictive term.

- *Regularization objective*: This term is taken from [66]. The idea is to model the weights of the network using a mixture of two Gaussians, a narrow (n) one, and a broad (b) one

$$p(w) = \pi_n \frac{1}{\sqrt{2\pi}\sigma_n} e^{-(x-\mu_n)^2/2\sigma_n^2} + \pi_b \frac{1}{\sqrt{2\pi}\sigma_b} e^{-(x-\mu_b)^2/2\sigma_b^2} \tag{15}$$

where the parameters of the distributions, π, σ, and μ, are obtained by means of an expectation-maximization (EM) algorithm [67]. The effect of this regularization term is a kind of *soft* version of weight-sharing in which the learning process decides itself which weights should be tied together.

3) Objectives of Cooperation: These two objectives explicitly promote cooperation among the networks. Instead of evaluating the performance of networks or ensembles, these objectives evaluates the relevance of the network within the ensembles where it participates and how well it cooperates with the rest of the members of those ensembles. These two objectives for a network ν in a subpopulation π, are the following.

- *Difference*: The network is removed from all the ensembles where it is present, and the performance of such ensembles with the network removed is measured. The value of this criterion is measured as the difference in performance of these ensembles with and without the network. This criterion enforces competition among subpopulations of networks 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 networks will be near 0 and the networks will be penalized.

- *Substitution*: The best k ensembles of the population are selected. In these ensembles the network of subpopulation π is substituted by the network ν. The fitness of the ensemble with the network of the subpopulation π substituted by ν is measured. The fitness assigned to the network is the average difference in the fitness of the ensembles with the original network and with the network substituted by ν. This criterion enforces competition among networks of the same subpopulation, as it tests if a network can achieve a better performance than the rest of the networks of its subpopulation.

4) Objectives of Diversity: In the development of ensembles of neural networks, we must take into account the source of the error on an ensemble. The ensemble generalization error can be expressed (see Section IV-C)

$$E = \bar{E} - \bar{A} \tag{16}$$

where $\bar{E} = \sum_i \alpha_i E_i$ is the weighted average of the individual networks' generalization errors and $\bar{A} = \sum_i \alpha_i A_i$ is the weighted average of the diversity among these networks. From this point of view, the objective of any ensemble is to obtain highly correct networks that disagree as much as possible.

Maintaining diversity needs some kind of speciation mechanism [68]. The most common techniques are: fitness sharing [22], [69], crowding [70], implicit fitness sharing [12], [68], local mating [71], and negative correlation [18]. Nevertheless most of these methods may bias the evolutionary process.

The importance of the diversity of combined networks in an ensemble have been stated by many authors [1], [18], [21], [24]. Nevertheless, some works raise some doubts about the usefulness of diversity measures in building classifier ensembles in real-world classification problems [24]. Moreover, it is very difficult to determine which measures of diversity are the most suitable for a given task. Our approach takes advantage of the cooperative and multiobjective environment we are using. We define four objectives regarding diversity, and the evolutionary process will combine networks that are good at different objectives. The number of diversity measures is enormous, so we have selected four of the most widely used, each one centered on a different idea. These objectives are the following.

- *Correlation*: Following Liu and Yao [21], we introduce an objective that measures the correlation of the error of each individual network with the ensembles where it participates. The error correlation of network i in an ensemble of networks P_i is measured using

$$P_i = \frac{1}{N}\sum_{n=1}^{P} p_i(\boldsymbol{x})$$
$$p_i(\boldsymbol{x}) = (f_i(\boldsymbol{x}) - F(\boldsymbol{x}))\sum_{j\neq i}(f_j(\boldsymbol{x}) - F(\boldsymbol{x})) \tag{17}$$

where P is the number of training patterns, $f_i(\boldsymbol{x})$ is the output of network i for pattern $\boldsymbol{x}$, and $F(\boldsymbol{x})$ is the output of the ensemble for pattern $\boldsymbol{x}$. With this measure a network must learn what all other networks have not yet learned.

The value used as objective is $-\bar{p}_i$, that is, the average error correlation of the network over all the ensembles where it is present.

From both theoretical and experimental results [72] it has been shown that, if the individual networks in an ensemble are unbiased, the most effective combination of them occurs when the errors of the individual networks are negatively correlated. As a consequence, the mutual information between each individual and the rest of the population should be minimized [18] to improve the estimation of the ensemble. This idea has been used before in other papers [18], and a very similar idea is implemented in [20].

- *Functional diversity*: Before the definition of a functional diversity measure, we defined such function axiomatically. The three axioms that a functional diversity measure for two functions f_i and f_j, $\phi(f_i, f_j)$, over a set $\mathcal{T}$, must fulfill are the following.

Axiom 1: $\phi(f_i, f_j) \geq 0$.
Axiom 2: $\phi(f_i, f_j) = 0$ if and only if $f_i(\boldsymbol{x}) = f_j(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{T}$.
Axiom 3: $\phi(f_i, f_j) = \phi(f_j, f_i)$.

Obviously, any distance measure fulfills these axioms. So, the measure we have chosen is the average Euclidean distance among the outputs of the two networks.

This measure is used to test the discrepancy among the outputs of the networks. For two networks, f_i and f_j, the functional diversity $\phi(f_i, f_j)$, is defined as

$$\phi(f_i, f_j) = \frac{1}{P}\sum_{k=1}^{P} \|f_i(\boldsymbol{x}_k) - f_j(\boldsymbol{x}_k)\|. \tag{18}$$

A similar measure, using a principal components analysis [73] of the outputs have been used in [12], [74].

- *Mutual information*. The mutual information [23] between two networks, f_i and f_j, is given by [18]

$$I(f_i, f_j) = H(f_i) + H(f_j) - H(f_i, f_j) \tag{19}$$

where $H(f_k)$ is the entropy of f_k, and $H(f_i, f_j)$ is the joint differential entropy of f_i and f_j. If we suppose that the output f_k of network k is a Gaussian random variable with variance δ_k^2, the differential entropy, $h(f_k)$ is given by

$$h(f_k) = \frac{1}{2}[1 + \log(2\pi\sigma_k^2)]. \tag{20}$$

The joint differential entropy $h(f_i, f_j)$ is given by

$$h(f_i, f_j) = 1 + \log(2\pi) + \frac{1}{2}\log|\det(\Sigma)| \tag{21}$$

where Σ is the covariance matrix of f_i and f_j. Following Liu *et al.* [18], who used this criterion for evolving a population of networks, the final form of the mutual information between the two networks is

$$I(f_i, f_j) = -\frac{1}{2}\log(1 - \rho_{ij}^2) \tag{22}$$

where ρ_{ij} is the correlation coefficient between f_i and f_j.

- *Yule's Q statistic*. This objective [24] measures whether the mistakes of the classifiers are uncorrelated, and it is one of the various statistics to assess the similarity of two classifiers outputs [75]. The Yule's Q statistic [76] for two classifiers D_i and D_k is given by

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \tag{23}$$

where N^{ab} is given by

	D_k correct (1)	D_k wrong (0)
D_i correct (1)	N^{11}	N^{10}
D_i wrong (0)	N^{01}	N^{00}

.

We have selected this measure, because it has been proved to be the one with the best results in a recent paper comparing ten different diversity measures [24]. Classifiers that recognize the same patterns will have positive values of Q, and classifiers that tend to commit mistakes in different patterns will have negative values of Q.

B. Ensemble Objectives

The same principles of multiobjective evolution can also be applied to the evolution of ensembles. So, we have defined two objectives to be considered in the evaluation of ensemble fitness. These two objectives for the ensembles are as follows.

- *Performance*: This objective is just the performance of the network measured as the number of patterns classified correctly.
- *Ambiguity*: As we have explained in Section IV-C, if the in-correlation among the networks that form the ensemble is increased, *without increasing the individual errors*, the global error of the ensemble is reduced. The ambiguity is defined over the training set.

The ensembles could also be evolved taking into account just one objective, their performance in solving the given problem, but the multiobjective approach showed better results.

C. Multiobjective Algorithm

The multiobjective algorithm is common to both populations, networks and ensembles. 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 comparison and selection of the most suitable multiobjective algorithm is not a trivial task [77]. The proposed algorithm is based on the concept of Pareto optimality [78] and has been chosen taking into account as the most important feature the computational cost. It has common points with other multiobjective evolutionary algorithms. The multiobjective algorithm for obtaining the fitness of an individual with a vector of objectives $x^{(i)}$ is outlined in Fig. 4.

The comparison and selection of the most suitable multiobjective algorithm is not a trivial task [77]. This algorithm is basically an adaptation of nondominated sorting genetic algorithm (NSGA) [79] to evolutionary programming. The idea underlying NSGA is the use of a ranking selection method to emphasize current nondominated individuals and a niching method to maintain diversity in the population. The use of second generation algorithms, such as strength Pareto evolutionary algorithm (SPEA) [80] or Pareto archive evolutionary strategy (PAES) [81], is not feasible as the concept of a separate population of nondominated individuals cannot be translated into our model.

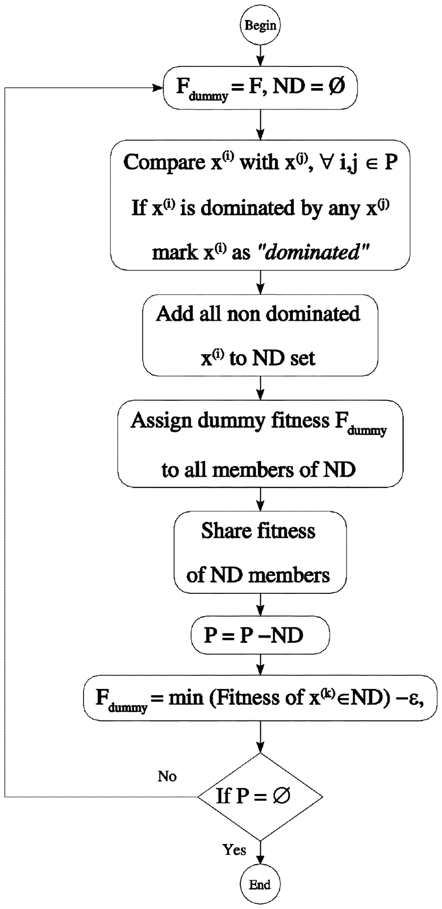
Begin
F_{dummy} = F, ND = Ø
Compare $x^{(i)}$ with $x^{(j)}$, ∀ i,j ∈ P
If $x^{(i)}$ is dominated by any $x^{(j)}$
mark $x^{(i)}$ as "dominated"
Add all non dominated $x^{(i)}$ to ND set
Assign dummy fitness F_{dummy} to all members of ND
Share fitness of ND members
P = P − ND
F_{dummy} = min (Fitness of $x^{(k)}$∈ND) −ε,
No
If P = Ø
Yes
End

Fig. 4. Multiobjective evolutionary algorithm for obtaining the fitness of individuals of the population.

The algorithm consists of two stages. First, the successive nondominated fronts[2] 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 procedure must guarantee that none of the members of a front gets a higher fitness than any of the members of the previous front.

The algorithm used for obtaining the nondominated set of solutions [78] compares the individuals pairwise and marks as *dominated* all the individuals that are dominated by at least one member of the population.

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

[2]A nondominated front is a subset of individuals that are not dominated by any member of the population.

The problem of fitness sharing among the members of the same Pareto front is crucial for the performance of the algorithm. The standard method of explicit fitness sharing [69] (for a discussion [82]) cannot be applied, as the individuals are not in an Euclidean space. So, we must define a measure of diversity for networks and ensembles in order to apply the algorithm. Our interest is focused on keeping behavioral diversity among the individuals, so our distance measure is based on the functional diversity measure defined above, $\phi(\cdot)$, for networks, and on an extension of functional diversity for ensembles.

Given a set of n_k networks of the kth nondominated 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 ith individual.

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

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{\text{share}}}\right)^2, & \text{if } d_{ij} \leq \sigma_{\text{share}} \\ 0, & \text{otherwise} \end{cases}. \tag{24}$$

3) Calculate niche count for individual i

$$m_i = \sum_{j=1}^{n_k} sh(d_{ij}). \tag{25}$$

4) Modify the fitness of the individual according to its niche count

$$f_i' = \frac{f_{\text{dummy}}}{m_i}. \tag{26}$$

For the ensemble population, we have defined an *ensemble functional diversity*. The measure of ensemble functional diversity is based on the functional diversity of networks. Given two ensembles $n^i = (n_1^i, n_2^i, \ldots, n_m^i)$ and $n^j = (n_1^j, n_2^j, \ldots, n_m^j)$, their functional diversity ϕ_{ensemble} is defined as

$$\phi_{\text{ensemble}}(n^i, n^j) = \sum_{k=1}^{m} \phi(n_k^i, n_k^j). \tag{27}$$

With this distance measure, the above sharing algorithm is applied, just substituting the measure of functional diversity $\phi(n_i, n_j)$ for the ensemble functional diversity $\phi_{\text{ensemble}}(n^i, n^j)$.

VI. Experimental Setup

The experiments were carried out with the objective of testing our cooperative model against the most used ensemble methods. We have applied our model and standard ensemble methods to several real-world problems of classification. These problems are briefly described in Table II. These ten sets cover a wide variety of problems. There are problems with different number of available patterns, from 214 to 3175, different number of classes, from 2 to 19, different kind of inputs, nominal, binary and continuous, and of different areas of application, from medical diagnosis to vowel recognition. Testing our model on this wide variety of problems can give us a clear idea of its performance.

The tests were conducted following the guidelines of Prechelt [83]. 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 error. There are two exceptions, Sonar and Vowel problems, as the patterns of these two problems are prearranged in two subsets due to their specific features. The column *network* in Table II specifies the architecture of the networks used in the standard ensembles.

The populations of cooperative ensembles were evolved without a validation set, adding the validation set to the training set. At the end of the evolution, the fifth best network, in terms of training error, was selected as the result of the evolution. The test set was then used for obtaining the generalization of this network.

The standard ensembles used for comparison are made up of 25 networks. It is known [84] that the diversity and the accuracy of the ensemble usually plateau at some size between 10 and 50 members. Moreover, Opitz and Maclin [85] have found after some exhaustive experiments that the error of the ensemble does not decrease after adding 25 networks.

For the training of the standard networks in the ensembles, we used the method of cross-validation and early stopping [86]. The networks were trained until the error over the validation set started to grow.

For Ada–Boosting, it is required that the *weak learning* algorithm, in our case, each individual network, achieves an error strictly less than 0.5. This cannot be guaranteed especially when dealing with multiclass problems. In our experiments when this error is not achieved, we generate a bootstrap sample from the original set and continue up to a limit of 25 trials. This method has been used in previous works [17], [56]. For two problems, Soybean and Vowel, the network was not able to reach this error and Ada–Boosting could not be applied.

For each data set, 30 runs of the algorithm were performed. In all the tables, we show the average error of classification over the 30 runs, the standard deviation, and the best and worst individuals. The measure of the error is the following:

$$E = \frac{1}{P} \sum_{i=1}^{P} e_i \tag{28}$$

where P is the number of patterns, and e_i is 0, if pattern i is correctly classified, and 1, otherwise.

The parameters used in our experiments are common to the ten problems. They are fairly standard as their performance is very good for the different problems we have solved. The cooperative ensembles used for all the problems are made up of ten networks, in contrast with the standard ensembles that are made up of 25 networks. We want to show how the cooperative coevolution of the networks can achieve a very good performance with a comparatively small ensemble. The population of ensembles has 100 individuals and each subpopulation of networks has 30 networks. The elitism in the population of networks is 50%.

TABLE II
SUMMARY OF DATA SETS. THE FEATURES OF EACH DATA SET CAN BE (C)ONTINUOUS, (B)INARY, OR (N)OMINAL. THE NETWORK COLUMN SHOWS THE NUMBER OF (I)NPUT, (H)IDDEN, AND (O)UTPUT NODES

	Cases		Features			Network	
Data set	Train Val Test	Class	C	B	N	(I:H:O)	Description
Cancer	360 175 174	2	9	-	-	9:10:2	There are two classes meaning if the cancer was benign (65.5% of the cases) or malignant (34.5%).
Card	346 172 172	2	6	4	5	51:8:2	There are two classes, meaning whether the application was granted (44.5% of the patterns) or denied (55.5%).
Gene	1588 794 793	3	-	-	60	120:20:3	This problem consists of two subtasks: recognizing exon/intron boundaries (referred to as EI sites), and recognizing intron/exon boundaries (IE sites).
Glass	107 54 53	6	9	-	-	9:12:6	This data set is from the UCI Machine Learning Repository. The set contains data from 6 different types of glass.
Heart	134 68 68	2	6	1	6	13:10:2	This data set comes from the Cleveland Clinic The goal is the prediction of the presence or absence of heart disease in those patients.
Horse	182 91 91	3	13	2	5	58:20:3	This data set is from the UCI Machine Learning Repository. The aim is to predict the fate of a horse that has a colic: to survive, to die, of to be euthanized.
Pima	384 192 192	2	8	-	-	8:8:2	The patterns are divided into two classes that show whether the patient shows signs of diabetes.
Sonar	104 - 104	2	60	-	-	60:10:2	The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock.
Soybean	342 171 170	19	6	13	16	82:10:19	The task is to recognize 19 different diseases of soybeans.
Vowel	528 - 462	11	10	-	-	10:20:11	Speaker independent recognition of the eleven steady state vowels of British English.

The parametric mutation rate in the population of ensembles is 100% and the structural mutation rate is 5%.

VII. EXPERIMENTAL RESULTS

The first step of our experiments was to obtain a lower limit for the performance of the model. In order to obtain such a limit, we trained a single network using the same model of the cooperative ensembles, the GMLP. It is obvious that an ensemble must be, at least, better than a single network. The results are shown in Table III. The network was not able to achieve useful results for the Soybean and Vowel problems.

Once we have established a lower bound for the performance of our model, we carried out an experiment in order to test the viability of the model. From the ten test problems we have described, we chose a subset of five problems: Cancer, Glass, Heart, Horse, and Pima. We made 30 runs for each problem, using all the objectives we have defined. The results are shown in Table IV.

The performance of the model is clearly better than the results of the single network. Nevertheless, the use of the ten objectives is not very advisable, because some of the objectives share the same principles. So, we did a second experiment in order to test whether all the objectives were useful in the evolution. For two problems, Heart and Glass, we repeated the 30 runs removing each objective in turn. The results are shown in Table V.

Table V shows that removing an objective only has a significant effect on the generalization error (with a 5% confidence level) in four cases in the Heart problem, and in none of them in the Glass problem. Moreover, removing the functional diversity objective has a beneficial effect over the error in the Glass

TABLE III
ERROR RATES FOR A SINGLE NETWORK (SNG) AND STANDARD ENSEMBLES. THE t-TEST COMPARES THE AVERAGE ERROR OF THE EXPERIMENT WITH COOPERATIVE ENSEMBLES USING SIX OBJECTIVES AND THE EXPERIMENTS WITH A SINGLE NETWORK AND EACH ENSEMBLE METHOD

Method	Training				Validation				Generalization				t-test
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	Mean	StD	Best	Worst	
Cancer													
Sng	0.0742	0.0117	0.0639	0.1222	0.0312	0.0112	0.0171	0.0800	*0.0278*	0.0118	0.0172	0.0632	0.0000
Std	0.0695	0.0051	0.0611	0.0833	0.0284	0.0044	0.0171	0.0343	*0.0238*	0.0060	0.0172	0.0402	0.0000
Bag	0.0678	0.0037	0.0611	0.0750	0.0280	0.0046	0.0171	0.0343	*0.0215*	0.0052	0.0115	0.0287	0.0000
Arc	0.0651	0.0027	0.0583	0.0722	0.0257	0.0044	0.0171	0.0343	*0.0195*	0.0044	0.0115	0.0287	0.0000
Ada	0.0719	0.0037	0.0639	0.0778	0.0269	0.0048	0.0171	0.0343	*0.0249*	0.0046	0.0172	0.0287	0.0000
Card													
Sng	0.1303	0.0190	0.0954	0.1705	0.2097	0.0187	0.1744	0.2500	*0.1607*	0.0271	0.1163	0.2151	0.0000
Std	0.0694	0.0054	0.0549	0.0780	0.1888	0.0091	0.1744	0.2093	*0.1279*	0.0095	0.1105	0.1453	0.0190
Bag	0.0698	0.0064	0.0578	0.0809	0.1884	0.0087	0.1744	0.2093	*0.1271*	0.0099	0.1105	0.1453	0.0420
Arc	0.0949	0.0090	0.0809	0.1214	0.1936	0.0141	0.1628	0.2267	*0.1318*	0.0122	0.1163	0.1628	0.0010
Ada	0.1007	0.0102	0.0838	0.1243	0.2010	0.0119	0.1802	0.2267	*0.1390*	0.0144	0.1105	0.1744	0.0000
Gene													
Sng	0.2303	0.0247	0.1940	0.2966	0.2537	0.0252	0.1877	0.3199	*0.2545*	0.0267	0.2081	0.3153	0.0000
Std	0.0916	0.0039	0.0844	0.0995	0.1362	0.0085	0.1222	0.1511	*0.1408*	0.0069	0.1236	0.1538	0.0000
Bag	0.0930	0.0034	0.0882	0.1008	0.1361	0.0084	0.1159	0.1511	*0.1396*	0.0064	0.1248	0.1538	0.0000
Arc	0.1011	0.0055	0.0926	0.1134	0.1480	0.0071	0.1348	0.1688	*0.1519*	0.0082	0.1324	0.1639	0.0000
Ada	0.0893	0.0056	0.0787	0.1020	0.1375	0.0083	0.1159	0.1524	*0.1429*	0.0092	0.1261	0.1614	0.0000
Glass													
Sng	0.4442	0.0598	0.3084	0.6075	0.4932	0.0525	0.3889	0.6296	*0.4862*	0.0909	0.2830	0.6226	0.0000
Std	0.1611	0.0300	0.1121	0.2430	0.3951	0.0365	0.3148	0.4630	*0.3566*	0.0388	0.2830	0.4340	0.0000
Bag	0.1526	0.0265	0.1028	0.1963	0.3889	0.0512	0.3148	0.5000	*0.3453*	0.0432	0.2830	0.4340	0.0000
Arc	0.1495	0.0214	0.1121	0.1963	0.3994	0.0509	0.2778	0.4815	*0.3560*	0.0490	0.2264	0.4340	0.0000
Ada	0.2794	0.0236	0.2243	0.3458	0.3667	0.0317	0.3148	0.4444	*0.3145*	0.0259	0.2642	0.3585	0.0000
Heart													
Sng	0.1704	0.0267	0.1269	0.2463	0.1917	0.0404	0.1324	0.3088	*0.1897*	0.0575	0.1029	0.3529	0.0000
Std	0.0993	0.0148	0.0672	0.1269	0.1941	0.0233	0.1618	0.2500	*0.1569*	0.0216	0.1176	0.2059	0.0000
Bag	0.0940	0.0137	0.0597	0.1194	0.1966	0.0199	0.1618	0.2353	*0.1618*	0.0250	0.1176	0.2206	0.0000
Arc	0.1072	0.0159	0.0896	0.1418	0.1907	0.0313	0.1471	0.2794	*0.1569*	0.0346	0.1029	0.2500	0.0000
Ada	0.1294	0.0146	0.0970	0.1493	0.1730	0.0199	0.1324	0.2206	*0.1569*	0.0260	0.1176	0.2353	0.0000
Horse													
Sng	0.2810	0.0455	0.2143	0.4011	0.4190	0.0553	0.3187	0.5495	*0.3462*	0.0413	0.2857	0.4615	0.0000
Std	0.1156	0.0123	0.0934	0.1374	0.3725	0.0286	0.3297	0.4286	*0.2879*	0.0246	0.2308	0.3297	0.0100
Bag	0.1141	0.0139	0.0824	0.1484	0.3740	0.0275	0.3297	0.4176	*0.2949*	0.0294	0.2418	0.3626	0.0015
Arc	0.1328	0.0177	0.0989	0.1648	0.3890	0.0335	0.3297	0.4615	*0.3099*	0.0314	0.2418	0.3846	0.0000
Ada	0.1595	0.0142	0.1319	0.1868	0.3879	0.0304	0.3297	0.4396	*0.2985*	0.0233	0.2527	0.3516	0.0001
Pima													
Sng	0.2757	0.0398	0.2396	0.4557	0.2854	0.0495	0.2292	0.5000	*0.2622*	0.0517	0.1875	0.4375	0.0000
Std	0.2295	0.0089	0.2083	0.2422	0.2295	0.0162	0.2031	0.2708	*0.2174*	0.0144	0.1875	0.2500	0.0000
Bag	0.2316	0.0070	0.2161	0.2474	0.2339	0.0104	0.2135	0.2552	*0.2179*	0.0138	0.1875	0.2344	0.0000
Arc	0.2226	0.0133	0.1875	0.2422	0.2424	0.0192	0.2031	0.2865	*0.2297*	0.0145	0.1979	0.2552	0.0000
Ada	0.2419	0.0095	0.2188	0.2630	0.2354	0.0151	0.1979	0.2656	*0.2076*	0.0213	0.1771	0.2448	0.0347
Sonar													
Sng	0.1532	0.0402	0.0865	0.2308	–	–	–	–	*0.2224*	0.0307	0.1538	0.3269	0.0000
Std	0.0372	0.0140	0.0000	0.0577	–	–	–	–	*0.2035*	0.0204	0.1635	0.2404	0.0000
Bag	0.0269	0.0108	0.0096	0.0481	–	–	–	–	*0.1910*	0.0218	0.1538	0.2404	0.0000
Arc	0.0321	0.0154	0.0000	0.0577	–	–	–	–	*0.2205*	0.0211	0.1827	0.2596	0.0000
Ada	0.0147	0.0135	0.0000	0.0481	–	–	–	–	*0.2253*	0.0173	0.2019	0.2596	0.0000
Soybean													
Sng	–	–	–	–	–	–	–	–	–	–	–	–	–
Std	0.2882	0.0289	0.2047	0.3450	0.3396	0.0344	0.2632	0.3918	*0.3782*	0.0334	0.3235	0.4353	0.0000
Bag	0.2867	0.0291	0.2251	0.3304	0.3419	0.0375	0.2573	0.4152	*0.3812*	0.0399	0.3176	0.4706	0.0000
Arc	0.2910	0.0303	0.2456	0.3538	0.3468	0.0347	0.2924	0.4269	*0.3786*	0.0347	0.3235	0.4529	0.0000
Ada	–	–	–	–	–	–	–	–	–	–	–	–	–
Vowel													
Sng	–	–	–	–	–	–	–	–	–	–	–	–	–
Std	0.5383	0.0411	0.4375	0.6042	–	–	–	–	*0.7582*	0.0309	0.6753	0.8074	0.0000
Bag	0.5492	0.0455	0.4508	0.6383	–	–	–	–	*0.7563*	0.0323	0.6948	0.8030	0.0000
Arc	0.5521	0.0429	0.4337	0.6250	–	–	–	–	*0.7598*	0.0244	0.7165	0.8352	0.0000
Ada	–	–	–	–	–	–	–	–	–	–	–	–	–

problem. These results showed us that the use of all the defined objectives was not the best option. That idea is also reinforced by two additional reasons.

1) The literature on multiobjective optimization shows that these algorithms do not perform successfully with so many objectives as ten [87].
2) Many of the objectives are closely related and it would be more advisable to choose a few or just one of them from each group.

In order to test the relevance of each objective, we carried out another experiment over Heart and Glass data sets using every objective alone in turn. The results are shown in Table VI.

TABLE IV
ERROR RATES USING ALL OF THE TEN OBJECTIVES

Problem	Training				Generalization			
	Mean	StD	Best	Worst	Mean	StD	Best	Worst
Cancer	0.0230	0.0011	0.0210	0.0248	*0.0142*	0.0058	0.0057	0.0287
Glass	0.1497	0.0234	0.1056	0.2050	*0.2824*	0.0471	0.2075	0.3774
Heart	0.0886	0.0051	0.0792	0.0941	*0.1289*	0.0192	0.1029	0.1765
Horse	0.1167	0.0115	0.0989	0.1429	*0.2956*	0.0319	0.2418	0.3956
Pima	0.2171	0.0018	0.2101	0.2188	*0.1991*	0.0135	0.1719	0.2240

TABLE V
ERROR RATES USING NINE OBJECTIVES FOR HEART AND GLASS DATA SETS. THE t-TEST COMPARES THE AVERAGE ERROR OF THE EXPERIMENT WITH TEN OBJECTIVES AND EACH EXPERIMENT USING NINE OBJECTIVES

Removed objective	Training				Generalization				t-test
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	
					Glass				
Difference	0.1356	0.0192	0.0994	0.1677	*0.2792*	0.0395	0.2075	0.3585	0.6564
Substitution	0.1323	0.0226	0.0870	0.1739	*0.2862*	0.0470	0.1698	0.3585	0.7982
Shared performance	0.1400	0.0255	0.0870	0.1988	*0.2893*	0.0382	0.2264	0.3585	0.5347
Performance	0.1391	0.0305	0.0745	0.1925	*0.2818*	0.0423	0.2075	0.3774	0.9568
Ensembles	0.1329	0.0229	0.0870	0.1739	*0.2818*	0.0432	0.1698	0.3585	0.8276
Regularization	0.1188	0.0238	0.0807	0.1739	*0.2899*	0.0395	0.2264	0.3774	0.5041
Func. diversity	0.1180	0.0203	0.0683	0.1677	*0.2522*	0.0410	0.1887	0.3396	0.0104
Yule's Q	0.1321	0.0224	0.0932	0.1739	*0.2805*	0.0370	0.1887	0.3396	0.8180
Correlation	0.1282	0.0240	0.0870	0.1677	*0.2761*	0.0504	0.1698	0.3585	0.6195
Mutual info	0.1362	0.0200	0.1056	0.1801	*0.2893*	0.0413	0.2075	0.3774	0.5476
					Heart				
Difference	0.0809	0.0064	0.0644	0.0941	*0.1402*	0.0264	0.0882	0.2059	0.1057
Substitution	0.0794	0.0091	0.0594	0.0941	*0.1333*	0.0253	0.0882	0.2059	0.5065
Shared performance	0.0799	0.0061	0.0693	0.0891	*0.1446*	0.0262	0.1029	0.1912	0.0107
Performance	0.0789	0.0075	0.0644	0.0941	*0.1382*	0.0188	0.1029	0.1765	0.0622
Ensembles	0.0797	0.0070	0.0644	0.0941	*0.1426*	0.0232	0.0882	0.1912	0.0155
Regularization	0.0751	0.0084	0.0545	0.0891	*0.1475*	0.0373	0.1029	0.2500	0.0193
Func. diversity	0.0766	0.0079	0.0594	0.0891	*0.1415*	0.0230	0.1029	0.1912	0.0249
Yule's Q	0.0785	0.0065	0.0693	0.0941	*0.1363*	0.0185	0.1029	0.1765	0.1362
Correlation	0.0795	0.0073	0.0644	0.0941	*0.1304*	0.0231	0.0882	0.1765	0.7894
Mutual info	0.0790	0.0066	0.0693	0.0891	*0.1368*	0.0242	0.0735	0.1912	0.1999

Due to the surprisingly good performance on the Glass problem of the objectives *substitution* and *difference* alone, we performed an additional test with these objectives on the Pima data set (also, shown in Table VI). Further experimentation has shown that *substitution* and *difference* alone are very efficient in producing low learning errors. In problems where learning and generalization errors are highly correlated, the results of these objectives alone are excellent.

TABLE VI
ERROR RATES USING ONE OBJECTIVE FOR HEART, GLASS, AND PIMA DATA SETS. THE t-TEST COMPARES THE AVERAGE ERROR OF THE EXPERIMENTS WITH TEN OBJECTIVES AND EACH EXPERIMENT USING ONE OBJECTIVE

Objective	Training				Generalization				t-test
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	
Glass									
Difference	0.0422	0.0090	0.0248	0.0621	*0.2358*	0.0353	0.1887	0.3208	0.0001
Substitution	0.0395	0.0112	0.0186	0.0621	*0.2270*	0.0482	0.1321	0.3585	0.0000
Shared performance	0.0383	0.0099	0.0186	0.0621	*0.2346*	0.0236	0.2075	0.2830	0.0000
Performance	0.0644	0.0143	0.0435	0.0932	*0.2553*	0.0335	0.1887	0.3208	0.0046
Ensembles	0.0536	0.0124	0.0248	0.0807	*0.2453*	0.0367	0.1698	0.3019	0.0010
Regularization	0.1942	0.0102	0.1739	0.2174	*0.3031*	0.0317	0.2642	0.3585	0.0612
Func. diversity	0.0986	0.0190	0.0683	0.1304	*0.2711*	0.0471	0.1698	0.3396	0.3052
Yule's Q	0.2636	0.0860	0.0745	0.3789	*0.3453*	0.0506	0.2264	0.4528	0.0000
Correlation	0.0814	0.0178	0.0559	0.1366	*0.2698*	0.0333	0.2075	0.3585	0.1951
Mutual info	0.4737	0.0667	0.2609	0.5652	*0.4547*	0.0798	0.2830	0.6038	0.0000
Heart									
Difference	0.0566	0.0048	0.0495	0.0644	*0.1270*	0.0206	0.1029	0.1765	0.5510
Substitution	0.0578	0.0100	0.0396	0.0743	*0.1361*	0.0244	0.0882	0.1912	0.2191
Shared performance	0.0901	0.0122	0.0693	0.1139	*0.1549*	0.0252	0.1176	0.2206	0.0000
Performance	0.0678	0.0074	0.0545	0.0842	*0.1426*	0.0239	0.1029	0.1765	0.0171
Ensembles	0.0601	0.0083	0.0396	0.0743	*0.1348*	0.0193	0.1029	0.1765	0.2859
Regularization	0.1033	0.0025	0.0990	0.1089	*0.1304*	0.0228	0.0882	0.1765	0.7877
Func. diversity	0.0749	0.0089	0.0545	0.0891	*0.1392*	0.0200	0.0882	0.1765	0.0620
Yule's Q	0.1266	0.0143	0.0941	0.1683	*0.1627*	0.0267	0.1176	0.2353	0.0000
Correlation	0.0629	0.0052	0.0545	0.0743	*0.1377*	0.0227	0.1029	0.1912	0.1092
Mutual info	0.2078	0.0349	0.1238	0.2921	*0.2034*	0.0330	0.1324	0.2647	0.0000
Pima									
Difference	0.2098	0.0013	0.2066	0.2118	*0.2118*	0.0138	0.1823	0.2396	0.0059
Substitution	0.2109	0.0016	0.2083	0.2135	*0.2118*	0.0120	0.1927	0.2396	0.0037

From these experiments, we obtain a subset of six objectives to be used for all the problems. This subset is made up of *difference, substitution, ensembles, shared performance, regularization,*and *Yule's Q*. These objectives were selected with the criterion of selecting at least one objective from each group and within each group selecting the best performing one in the two previous experiments.

The results for the ten problems with this subset of six objectives are shown in Table VII. The results are excellent and are among the best in the literature [4], [16], [35], [56], [85].

Table VII also shows the computational effort needed for obtaining the given results. The computational effort of an evolutionary process where all the evolutions end in success, as it is our case, can be defined [88] as the number of evaluations of the fitness function. We have a population of fixed size $M = 100$, so the number of evaluations of the fitness function in G generations is $M \times G$. In the table, we show the average number of generations of the 30 runs of each experiment.

The results obtained are very good when they are compared with other works using these data sets. Table VIII shows a summary of the results reported in papers devoted to ensemble or similar classification methods. Comparisons must be made cautiously, as the experimental setup is different in many papers. There are differences also in the methods used for estimating the generalization error. Some of the papers use tenfold cross-validation that for some of the problems obtains a more optimistic

TABLE VII
ERROR RATES USING THE SUBSET OF SIX OBJECTIVES FOR ALL THE PROBLEMS. THE COMPUTATIONAL EFFORT IS SHOWN AS THE AVERAGE NUMBER OF GENERATIONS

Problem	Training				Generalization				Generations
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	
Cancer	0.0230	0.0009	0.0210	0.0248	*0.0123*	0.0047	0.0057	0.0230	20
Card	0.0873	0.0034	0.0792	0.0927	*0.1217*	0.0103	0.1047	0.1395	20
Gene	0.0727	0.0074	0.0626	0.0936	*0.1238*	0.0105	0.1021	0.1425	20
Glass	0.0886	0.0103	0.0745	0.1180	*0.2289*	0.0482	0.1321	0.3019	30
Heart	0.0784	0.0057	0.0644	0.0842	*0.1196*	0.0207	0.0735	0.1618	20
Horse	0.1521	0.0086	0.1355	0.1758	*0.2674*	0.0343	0.2088	0.3407	20
Pima	0.2131	0.0016	0.2083	0.2153	*0.1969*	0.0170	0.1615	0.2292	20
Sonar	0.0003	0.0018	0.0000	0.0096	*0.1436*	0.0153	0.1154	0.1731	20
Soybean	0.0200	0.0027	0.0156	0.0273	*0.0761*	0.0083	0.0588	0.1000	20
Vowel	0.0340	0.0118	0.0152	0.0606	*0.4587*	0.0269	0.4156	0.5152	30

TABLE VIII
RESULTS OF PREVIOUS WORKS USING THE SAME DATA SETS. WE RECORD THE RESULTS OF THE BEST METHOD AMONG THE ALGORITHMS TESTED IN EACH PAPER

Data set	Reference													
	Coop	[16][1]	[4][2]	[84][3]	[6][1]	[95][2]	[15][2]	[18][2]	[96][1]	[97][2]	[5][2]	[98][1]	[99][2]	[100][1]
Cancer	0.0123	–	0.035	–	–	0.038	–	–	0.0120	0.0310	0.0272	0.034	0.0263	0.033
Card	0.1217	0.093	–	–	0.1398	–	0.135	0.130	0.0910	0.1300	0.1432	–	0.1433	–
Gene	0.1238	–	–	–	–	–	–	–	–	0.0503	–	0.051	–	–
Glass	0.2289	–	0.249	–	0.3144	0.238	–	–	0.2518	0.2277	0.2519	0.226	0.3154	0.329
Heart	0.1196	0.151	0.197	0.166	0.1751	–	–	–	0.1384	0.2045	0.1604	–	0.1617	–
Horse	0.2674	–	0.169	–	–	–	–	–	–	0.1825	–	–	–	–
Pima	0.1969	0.226	0.244	0.234	–	–	0.221	0.223	0.1960	–	0.2402	–	0.2372	0.260
Sonar	0.1436	–	–	–	0.2278	0.154	–	–	–	0.1651	0.1529	0.163	–	–
Soybean	0.0761	–	0.070	–	–	–	–	–	0.0781	0.0757	0.0633	0.056	0.0568	–
Vowel	0.4587	–	–	–	–	0.517[1]	–	–	–	–	–	–	–	–

[1] Hold out.

[2] k-fold cross-validation.

[3] Best classifier.

estimation of the error. With these cautions, we can say that for Cancer, Glass, Heart, Pima, Sonar, Soybean, and Vowel data sets our methods achieve a performance that is better or similar to all the results reported in the cited papers. Gene and Horse results are poorer than those obtained by other papers, and card results are improved by two of the papers. As in our experiments, most of these papers use an experimental setup and a set of parameters common to all the problems.

VIII. COMPARISON WITH STANDARD ENSEMBLE MODELS

In order to assure the level of performance of the model, we made a comparison with standard ensembles of neural networks. We trained four different ensembles of neural networks, using the same individual neural network and the same back-propagation algorithm that we used for the cooperative ensemble. The four ensembles are the following.

TABLE IX
AVERAGE SIZE OF THE EVOLUTIONARY AND NONEVOLUTIONARY ENSEMBLES. THE TABLE SHOWS NUMBER OF NETWORKS IN THE ENSEMBLE, AND THE AVERAGE SIZE OF THE ENSEMBLE AND OF EACH NETWORK. THE t-TEST COMPARES THE AVERAGE NUMBER OF NODES AND CONNECTIONS FOR ENSEMBLES AND CONSTITUENT NETWORKS

	Cooperative ensemble					Classical ensembles					t-tests			
	Ensemble size			Network size		Ensemble size			Network size		Ensemble		Network	
	Nets	Nodes	Conns	Nodes	Conns	Nets	Nodes	Conns	Nodes	Conns	Nodes	Conns	Nodes	Conns
Cancer	10	58.9	583.0	5.89	58.30	25	250	3000	10	120	0.0000	0.0000	0.0000	0.0000
Card	10	68.9	1881.3	6.89	188.13	25	200	10800	8	432	0.0000	0.0000	0.0000	0.0000
Gene	10	97.5	6421.9	9.75	642.19	25	500	62000	20	2480	0.0000	0.0000	0.0000	0.0000
Glass	10	67.3	839.7	6.73	83.97	25	250	4800	12	192	0.0000	0.0000	0.0000	0.0000
Heart	10	72.8	903.1	7.28	90.31	25	240	4000	10	160	0.0000	0.0000	0.0000	0.0000
Horse	10	203.0	7796.7	20.30	779.67	25	500	31000	20	1240	0.0000	0.0000	0.3558	0.0000
Pima	10	79.0	763.2	7.90	76.32	25	200	2200	8	88	0.0000	0.0000	0.6207	0.0000
Sonar	10	64.3	2567.9	6.43	256.79	25	250	15750	10	630	0.0000	0.0000	0.0000	0.0000
Soybean	10	194.2	12822.0	19.42	1282.20	25	250	25500	10	1020	0.0000	0.0000	0.0000	0.0000
Vowel	10	173.1	3442.9	17.31	344.29	25	500	11000	20	440	0.0000	0.0000	0.0000	0.0000

Standard (Std):	An ensemble of networks without sampling. The combination of networks is made using the *generalized ensemble method* (GEM) [1]. If two networks are linearly correlated, one of them is removed.
Bagging (Bag):	An ensemble of networks using Bagging [56]. As in *standard*, the combination of networks is made using GEM.
Arcing (Arc):	An ensemble of networks using one of the Boosting [89] methods, Arcing-x4 [56], [60]. As in previous methods, the combination of networks is made using GEM.
Ada (Ada):	An ensemble of networks using one of the Boosting methods, Ada-Boosting [59], [56]. The combination of networks is made following the Ada algorithm itself.

For each problem, we did 30 runs with every type of ensemble. All the experimental setup was done as in the experiments with the cooperative ensemble in order to make a fair comparison.

The results of these four ensemble methods are shown in Table III. Considering the overall performance, the best performing method is Ada-Boosting together with Bagging, the latter with a slightly worse performance than the former.

From Table III, we can see how the cooperative ensemble is better than the four ensemble models for all the problems with a confidence level of 5%. This result is more important if we consider that the cooperative ensemble uses 10 networks against the 25 networks that form the other ensembles. It is also interesting to note that the size of the networks in the evolved ensembles is also less than the size of the corresponding networks in the standard ensembles. Table IX shows the average size of the ensembles and networks. We can see that not only the cooperative ensembles have fewer networks, but also that the constituent networks are significantly smaller (with the exception of the soybean problem).

IX. ANALYSIS OF THE COOPERATIVE ENSEMBLES

In this section, we will study the behavior of the cooperative ensemble. In the previous section, we have assured that the model shows a dramatic reduction of generalization error when compared with standard ensembles. Here, we want to test the sensitivity of the model to the number of networks in the ensemble, the relevance of each objective and how it behaves in a bias/variance decomposition test.

A. Analysis of the Effect of Ensemble Size

In order to test the influence of the number of network subpopulations, that is, the size of the ensembles, we carried out experiments for Cancer, Glass, Heart, Horse, and Pima problems with 5, 10, 15, 25, and 30 subpopulations of networks. For each size we performed ten runs of the algorithm. The results are shown in Table X.[3]

The table shows that in some of the problems, namely, Heart and Horse, the addition of new networks to the ensemble produces an improvement in the performance of the model, but this increment is not significant and could not pay for the increased complexity of the model. We have performed and ANOVA I test in order to verify whether there are significant differences among the results obtained with different numbers of subpopulations. With a confidence level of 5% there are significant differences just in a few cases. We can assure that the generalization error is not significantly improved when more networks are added. Not surprisingly, the learning error is improved as new networks are being added to the ensemble.

[3]There are minor differences between Table X and Table VII, due to the fact that in this table we have only considered the first ten runs of the algorithm for the case of ten network subpopulations.

TABLE X
ERROR RATES USING 5, 10, 15, 25, AND 30 SUBPOPULATIONS OF NETWORKS FOR CANCER, GLASS, HEART, HORSE, AND PIMA DATA SETS.

Subpopulations	Training				Generalization			
	Mean	StD	Best	Worst	Mean	StD	Best	Worst
					Cancer			
5	0.0240	0.0022	0.0210	0.0267	*0.0138*	0.0073	0.0057	0.0230
10	0.0230	0.0006	0.0229	0.0248	*0.0115*	0.0027	0.0057	0.0172
15	0.0229	0.0000	0.0229	0.0229	*0.0126*	0.0045	0.0057	0.0172
20	0.0229	0.0000	0.0229	0.0229	*0.0144*	0.0030	0.0115	0.0172
25	0.0225	0.0008	0.0210	0.0229	*0.0126*	0.0065	0.0000	0.0230
30	0.0229	0.0000	0.0229	0.0229	*0.0115*	0.0061	0.0000	0.0230
					Glass			
5	0.1112	0.0186	0.0870	0.1366	*0.2604*	0.0532	0.1509	0.3396
10	0.0888	0.0093	0.0745	0.1056	*0.2151*	0.0464	0.1321	0.2830
15	0.0758	0.0064	0.0683	0.0870	*0.2245*	0.0372	0.1887	0.2830
20	0.0714	0.0089	0.0559	0.0807	*0.2566*	0.0428	0.2075	0.3208
25	0.0547	0.0087	0.0435	0.0683	*0.2226*	0.0396	0.1698	0.3019
30	0.0553	0.0046	0.0497	0.0612	*0.2189*	0.0347	0.1887	0.2830
					Heart			
5	0.0891	0.0081	0.0792	0.1040	*0.1426*	0.0310	0.1176	0.2059
10	0.0757	0.0078	0.0644	0.0842	*0.1221*	0.0197	0.1029	0.1471
15	0.0767	0.0053	0.0693	0.0891	*0.1265*	0.0232	0.1029	0.1765
20	0.0767	0.0053	0.0693	0.0842	*0.1265*	0.0221	0.0882	0.1618
25	0.0733	0.0039	0.0693	0.0792	*0.1221*	0.0250	0.0882	0.1618
30	0.0792	0.0033	0.0743	0.0842	*0.1147*	0.0152	0.0882	0.1324
					Horse			
5	0.1799	0.0050	0.1722	0.1868	*0.2780*	0.0156	0.2527	0.2967
10	0.1575	0.0073	0.1502	0.1758	*0.2703*	0.0324	0.2308	0.3187
15	0.1447	0.0046	0.1392	0.1538	*0.2681*	0.0399	0.1978	0.3187
20	0.1414	0.0046	0.1355	0.1465	*0.2516*	0.0271	0.2088	0.2967
25	0.1344	0.0055	0.1245	0.1429	*0.2582*	0.0239	0.2308	0.2967
30	0.1370	0.0070	0.1245	0.1465	*0.2462*	0.0195	0.2088	0.2747
					Pima			
5	0.2172	0.0017	0.2135	0.2188	*0.1948*	0.0254	0.1615	0.2344
10	0.2132	0.0014	0.2101	0.2153	*0.1974*	0.0164	0.1771	0.2292
15	0.2141	0.0016	0.2118	0.2170	*0.2036*	0.0231	0.1667	0.2344
20	0.2127	0.0020	0.2101	0.2153	*0.2083*	0.0196	0.1667	0.2344
25	0.2142	0.0009	0.2135	0.2153	*0.2010*	0.0174	0.1719	0.2292
30	0.2142	0.0009	0.2135	0.2153	*0.2016*	0.0179	0.1667	0.2240

B. Objectives Study

In this section, we evaluate the relevance of each objective in the overall performance of the ensemble. We carry out two studies. First, we test the individual contribution of each objective by removing it from the evolution and evaluating the performance of the model without that objective. Second, we evaluate the capability of every objective, evolving the ensembles with every objective alone in turn.

1) Necessity Analysis: The necessity analysis evaluates how relevant each objective is for the overall performance of the model. In order to test the importance of the objectives, we remove every objective in turn and evolve the model using the other five objectives. The results of the evolution with five objectives for ten runs are shown in Table XI. There are some interesting effects that can be observed in these results.

- As a general rule, all the objectives are useful. The error rate is in most cases worse when any of the objectives are removed. Nevertheless, the performance of the model considering five objectives is still acceptable.
- The learning error decreases when the regularization term is removed, but the generalization error usually increases without this objective. So, we can conclude that the regularization term is playing its role, encouraging small net-

TABLE XI
NECESSITY RESULTS. ERROR RATES REMOVING EVERY ONE OF THE SIX OBJECTIVES IN TURN FOR CANCER, GLASS, HEART, HORSE, AND PIMA DATA SETS. THE t-TEST COMPARES THE AVERAGE ERROR OF THE EXPERIMENT WITH SIX OBJECTIVES, ROW LABELED *ALL OBJECTIVES*, AND EACH EXPERIMENT USING FIVE OBJECTIVES

Removed objective	Training				Generalization				t-test
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	
Cancer									
All objectives	*0.0230*	*0.0009*	*0.0210*	*0.0248*	*0.0123*	*0.0047*	*0.0057*	*0.0230*	–
Difference	0.0225	0.0008	0.0210	0.0229	*0.0167*	0.0042	0.0115	0.0230	0.0125
Substitution	0.0232	0.0012	0.0210	0.0248	*0.0149*	0.0062	0.0057	0.0287	0.1576
Ensembles	0.0225	0.0012	0.0210	0.0248	*0.0149*	0.0062	0.0057	0.0230	0.1576
Shared performance	0.0227	0.0011	0.0210	0.0248	*0.0161*	0.0053	0.0057	0.0230	0.0369
Regularization	0.0190	0.0018	0.0171	0.0229	*0.0195*	0.0062	0.0057	0.0230	0.0004
Yule's Q	0.0229	0.0000	0.0229	0.0229	*0.0132*	0.0055	0.0057	0.0230	0.5952
Glass									
All objectives	*0.0886*	*0.0103*	*0.0745*	*0.1180*	*0.2289*	*0.0482*	*0.1321*	*0.3019*	–
Difference	0.0683	0.0171	0.0373	0.0932	*0.2547*	0.0528	0.1887	0.3208	0.1606
Substitution	0.0826	0.0117	0.0621	0.1056	*0.2623*	0.0483	0.1887	0.3396	0.0389
Ensembles	0.0683	0.0097	0.0559	0.0870	*0.2453*	0.0503	0.1509	0.3019	0.1800
Shared performance	0.0752	0.0122	0.0559	0.0994	*0.2434*	0.0301	0.1887	0.2830	0.1229
Regularization	0.0590	0.0098	0.0435	0.0745	*0.2358*	0.0481	0.1509	0.3019	0.3391
Yule's Q	0.1012	0.0134	0.0745	0.1242	*0.2755*	0.0311	0.2264	0.3396	0.0031
Heart									
All objectives	*0.0784*	*0.0057*	*0.0644*	*0.0842*	*0.1196*	*0.0207*	*0.0735*	*0.1618*	–
Difference	0.0777	0.0074	0.0693	0.0891	*0.1397*	0.0222	0.1029	0.1765	0.0128
Substitution	0.0817	0.0053	0.0743	0.0891	*0.1500*	0.0258	0.1176	0.2059	0.0005
Ensembles	0.0767	0.0067	0.0644	0.0842	*0.1426*	0.0241	0.1029	0.1912	0.0057
Shared performance	0.0777	0.0078	0.0594	0.0891	*0.1397*	0.0187	0.1176	0.1618	0.0098
Regularization	0.0673	0.0042	0.0594	0.0743	*0.1485*	0.0264	0.1176	0.2059	0.0010
Yule's Q	0.0782	0.0056	0.0693	0.0842	*0.1294*	0.0135	0.1029	0.1471	0.1710
Horse									
All objectives	*0.1521*	*0.0086*	*0.1355*	*0.1758*	*0.2674*	*0.0343*	*0.2088*	*0.3407*	–
Difference	0.1509	0.0132	0.1319	0.1685	*0.2912*	0.0356	0.2088	0.3297	0.0673
Substitution	0.1487	0.0083	0.1392	0.1648	*0.2725*	0.0314	0.2088	0.3187	0.6787
Ensembles	0.1520	0.0053	0.1429	0.1612	*0.2692*	0.0316	0.2088	0.3077	0.8824
Shared performance	0.1568	0.0071	0.1465	0.1648	*0.2835*	0.0206	0.2527	0.3187	0.1706
Regularization	0.1011	0.0173	0.0769	0.1209	*0.2637*	0.0298	0.2198	0.3297	0.7648
Yule's Q	0.1601	0.0109	0.1429	0.1795	*0.2846*	0.0271	0.2308	0.3187	0.1582
Pima									
All objectives	*0.2131*	*0.0016*	*0.2083*	*0.2153*	*0.1969*	*0.0170*	*0.1615*	*0.2292*	–
Difference	0.2141	0.0008	0.2135	0.2153	*0.2120*	0.0161	0.1875	0.2396	0.0184
Substitution	0.2132	0.0033	0.2049	0.2170	*0.2167*	0.0137	0.1979	0.2396	0.0019
Ensembles	0.2116	0.0031	0.2049	0.2153	*0.2161*	0.0171	0.1823	0.2396	0.0036
Shared performance	0.2130	0.0018	0.2101	0.2153	*0.2089*	0.0167	0.1875	0.2396	0.0602
Regularization	0.2120	0.0024	0.2066	0.2153	*0.2010*	0.0141	0.1771	0.2292	0.4899
Yule's Q	0.2149	0.0007	0.2135	0.2153	*0.2057*	0.0139	0.1823	0.2240	0.1457

works with worse learning errors but in most cases better generalization errors.

- The usefulness of the diversity objective is not clear. The deletion of this objective significantly increases the generalization error only in the Glass problem. This result agrees with the work of Kuncheva and Whitaker [24] that has raised some doubts on the use of diversity terms in the learning process.

2) Capability Analysis: The aim of capability analysis is to study the performance of every objective when it is used as the

TABLE XII
CAPABILITY RESULTS. ERROR RRATES USING ONE OBJECTIVE FOR CANCER, GLASS, HEART, HORSE, AND PIMA DATA SETS. THE t-TEST COMPARES THE AVERAGE ERROR OF THE EXPERIMENT WITH SIX OBJECTIVES, ROW LABELED *ALL OBJECTIVES*, AND EACH EXPERIMENT USING ONE OBJECTIVE

Objective	Training				Generalization				t-test
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	
					Cancer				
All objectives	*0.0230*	*0.0009*	*0.0210*	*0.0248*	*0.0123*	*0.0047*	*0.0057*	*0.0230*	–
Difference	0.0213	0.0020	0.0171	0.0229	*0.0195*	0.0082	0.0057	0.0287	0.0013
Substitution	0.0190	0.0034	0.0133	0.0229	*0.0172*	0.0061	0.0115	0.0287	0.0104
Ensembles	0.0202	0.0016	0.0171	0.0229	*0.0178*	0.0050	0.0115	0.0287	0.0029
Shared performance	0.0242	0.0031	0.0190	0.0286	*0.0218*	0.0053	0.0172	0.0287	0.0000
Regularization	0.0288	0.0011	0.0267	0.0305	*0.0138*	0.0040	0.0115	0.0230	0.3626
Yule's Q	0.0227	0.0014	0.0210	0.0248	*0.0172*	0.0054	0.0115	0.0230	0.0082
					Glass				
All objectives	*0.0886*	*0.0103*	*0.0745*	*0.1180*	*0.2289*	*0.0482*	*0.1321*	*0.3019*	–
Difference	0.0429	0.0085	0.0311	0.0559	*0.2396*	0.0179	0.2075	0.2642	0.5002
Substitution	0.0429	0.0115	0.0248	0.0621	*0.2396*	0.0427	0.1887	0.3208	0.5368
Ensembles	0.0590	0.0053	0.0497	0.0683	*0.2509*	0.0309	0.1887	0.2830	0.1857
Shared performance	0.0354	0.0093	0.0186	0.0497	*0.2358*	0.0204	0.2075	0.2642	0.6641
Regularization	0.1894	0.0107	0.1739	0.2112	*0.2906*	0.0335	0.2642	0.3585	0.0006
Yule's Q	0.2410	0.0653	0.0994	0.3292	*0.3264*	0.0388	0.2642	0.3962	0.0000
					Heart				
All objectives	*0.0784*	*0.0057*	*0.0644*	*0.0842*	*0.1196*	*0.0207*	*0.0735*	*0.1618*	–
Difference	0.0584	0.0039	0.0495	0.0644	*0.1279*	0.0209	0.1029	0.1618	0.2780
Substitution	0.0594	0.0099	0.0446	0.0743	*0.1294*	0.0276	0.0882	0.1765	0.2404
Ensembles	0.0589	0.0054	0.0495	0.0644	*0.1309*	0.0176	0.1176	0.1618	0.1311
Shared performance	0.0926	0.0138	0.0743	0.1139	*0.1559*	0.0210	0.1176	0.1765	0.0000
Regularization	0.1045	0.0028	0.0990	0.1089	*0.1265*	0.0199	0.0882	0.1471	0.3651
Yule's Q	0.1302	0.0107	0.1139	0.1535	*0.1603*	0.0176	0.1324	0.1912	0.0000
					Horse				
All objectives	*0.1521*	*0.0086*	*0.1355*	*0.1758*	*0.2674*	*0.0343*	*0.2088*	*0.3407*	–
Difference	0.0839	0.0085	0.0733	0.0989	*0.2714*	0.0220	0.2418	0.3077	0.7307
Substitution	0.0630	0.0057	0.0549	0.0696	*0.2780*	0.0249	0.2198	0.3187	0.3738
Ensembles	0.0919	0.0063	0.0806	0.0989	*0.2857*	0.0279	0.2527	0.3187	0.1357
Shared performance	0.0795	0.0083	0.0659	0.0916	*0.2681*	0.0188	0.2418	0.3077	0.9493
Regularization	0.2179	0.0072	0.2051	0.2271	*0.2890*	0.0194	0.2527	0.3187	0.0673
Yule's Q	0.2586	0.0202	0.2308	0.2894	*0.3176*	0.0409	0.2527	0.3846	0.0005
					Pima				
All objectives	*0.2131*	*0.0016*	*0.2083*	*0.2153*	*0.1969*	*0.0170*	*0.1615*	*0.2292*	–
Difference	0.2097	0.0011	0.2083	0.2118	*0.2115*	0.0070	0.2031	0.2240	0.0125
Substitution	0.2108	0.0019	0.2083	0.2135	*0.2135*	0.0092	0.1979	0.2292	0.0055
Ensembles	0.2092	0.0025	0.2031	0.2118	*0.2068*	0.0242	0.1615	0.2448	0.1608
Shared performance	0.2278	0.0050	0.2240	0.2413	*0.2130*	0.0164	0.1927	0.2500	0.0124
Regularization	0.2153	0.0008	0.2135	0.2170	*0.2146*	0.0175	0.1875	0.2448	0.0073
Yule's Q	0.2302	0.0096	0.2222	0.2500	*0.2208*	0.0251	0.1927	0.2708	0.0015

only objective to be evaluated along the evolution. So, we evolve the populations considering just one of the six objectives used for the evolution of networks. The results for Cancer, Glass, Heart, Horse, and Pima data sets are shown in Table XII. The performance of the isolated objectives shows two reasonable results.

- The objectives that are focused on performance, *difference*, *substitution*, *ensembles*, and *shared performance*,

have a better performance when they are used as the only objective, in some cases they can achieve the same performance as the six objectives.
- Objectives focused on other aspects of the ensembles, such as regularization or diversity, have a worse performance.

These two effects are more important when we consider the learning error. For instance, for Glass, Heart, and Horse data sets, the learning error is clearly improved when using performance based objectives. On the other hand, if we consider regularization and diversity objectives the learning error is more than twice the learning error using the six objectives.

C. Bias and Variance Decomposition

Many of the recent papers studying neural network ensembles analyze error performance in terms of two factors: *bias* and *variance*. The bias of a learning algorithm is the contribution to the error of the central tendency when it is trained using different data, while the variance is the contribution of the error of the deviations from the central tendency. These two terms are evaluated with respect to a distribution of training sets $\mathcal{T}$ usually obtained by different permutations of the available data.

In addition, there is an *irreducible error* that is given by the degree to which the correct answer for a pattern can differ from that for other patterns with the same description. As this error cannot be estimated in most real-world problems, the measures of bias and variance usually include this error.

Several authors have suggested different proposals for estimating the decomposition of the classification error, the bias and variance terms [49], [90], and [91]. All of them have different advantages and drawbacks, so we have used three measures in order to study the behavior of our cooperative model comprehensively (an excellent discussion of this topic can be found in [4]).

Let us assume that the training pairs, (x, y) are drawn from a test instance distribution X, Y, and that the classification of pattern x by means of classifier $\mathcal{L}$ for a distribution of training data sets $\mathcal{T}$ is $\mathcal{L}(T)(x)$. Kohavi and Wolpert [91] proposed a method where the expected zero-one error of classifier C can be expressed by

$$E(C) = \sum_x P(x)(\sigma_x^2 + \text{bias}_x^2 + \text{variance}_x) \tag{29}$$

where

$$\begin{aligned} \text{bias}_x^2 &\equiv \frac{1}{2}\sum_{y\in Y}[P_{Y,X}(Y=y \mid X=x) - P_{\mathcal{T}}(\mathcal{L}(T)(x)=y)]^2 \\ \text{variance}_x &\equiv \frac{1}{2}\left(1-\sum_{y\in Y}P(\mathcal{L}(T)(x)=y)^2\right) \\ \sigma_x^2 &\equiv \frac{1}{2}\left(1-\sum_{y\in Y}P(Y=y \mid X=x)^2\right). \end{aligned} \tag{30}$$

However, these measures of bias and variance do not measure the extent to which each of these underlying quantities contribute to error [4]. The irreducible error is included in the bias term. This estimation has the valuable property that the sum of bias and variance terms equals the total error.

The bias estimation of Kong and Dietterich [90] measures the probability of error of the central tendency of the learning algorithm. This measure is useful when comparing the quality of the central tendency of different classifiers, regarding other considerations as the frequency or strength of such central tendency. The formulation of this measure bias_{KD} is given by

$$\text{bias}_{\text{KD}} = P_{(Y,X),\mathcal{T}}(C^o_{\mathcal{L},\mathcal{T}}(X) \neq Y) \tag{31}$$

where the central tendency $C^o_{\mathcal{L},\mathcal{T}}(X)$, for learner $\mathcal{L}$ over the distribution of training data sets $\mathcal{T}$ is the class with the greatest probability of selection for pattern x by classifiers learned by $\mathcal{L}$ from training sets drawn from $\mathcal{T}$, and is defined

$$C^o_{\mathcal{L},\mathcal{T}}(X) = \max_y P_{\mathcal{T}}(\mathcal{L}(\mathcal{T})(x) = y). \tag{32}$$

Nevertheless, the estimation of the variance of Kong and Dietterich does not adequately measure the error due to the deviations from the central tendency. So, we have also used the decomposition defined by Breiman [92]

$$\text{bias}_B = P(Y,X),\mathcal{T}(\mathcal{L}(\mathcal{T}) \neq Y \wedge \mathcal{L}(\mathcal{T})(X) = C^o_{\mathcal{L},\mathcal{T}}(X)) \tag{33}$$

$$\text{variance}_B = P(Y,X),\mathcal{T}(\mathcal{L}(\mathcal{T}) \neq Y \wedge \mathcal{L}(\mathcal{T})(X) \neq C^o_{\mathcal{L},\mathcal{T}}(X)). \tag{34}$$

These definitions have the advantage that the bias term is a direct measure of the contribution of the central tendency to the total error, and variance is a measure of the contribution of the deviations from the central tendency to the total error.

For estimating these five measures, we have basically followed the experimental setup used in [4]. We divided our data set into four randomly selected partitions. We selected each partition in turn to be used as the test set and trained the learner with the other three partitions. This method was repeated ten times with different random partitions, making a total of 40 runs of the learning algorithm.

The central tendency was evaluated as the most frequent classification for a pattern. The error was measured as the proportion of incorrectly classified patterns. This experimental setup guarantees that each pattern is selected for the test set the same number of times, and alleviates the effect that the random selection of patterns can have over the estimations.

Fig. 5 shows the estimation of the bias of Kong and Dietterich for the learning algorithms used. Instead of representing the value of the bias, we have chosen a value of 1 for the estimation obtained for the cooperative ensemble and we represent the relative bias of the rest of the ensemble methods to the cooperative ensemble.

Fig. 5 shows how the cooperative ensemble central tendency is always more accurate than the central tendency of the rest of the ensemble methods. This difference is more drastic in complex problems, such as Gene, Soybean, and Vowel.

Fig. 6 shows the estimation of the relative bias and variance of Kohavi and Wolpert. As we have stated, the contribution of bias to error is the portion of the total error that is made by the central tendency of the algorithm. The contribution of variance is the portion of the error that is due to deviations from the central

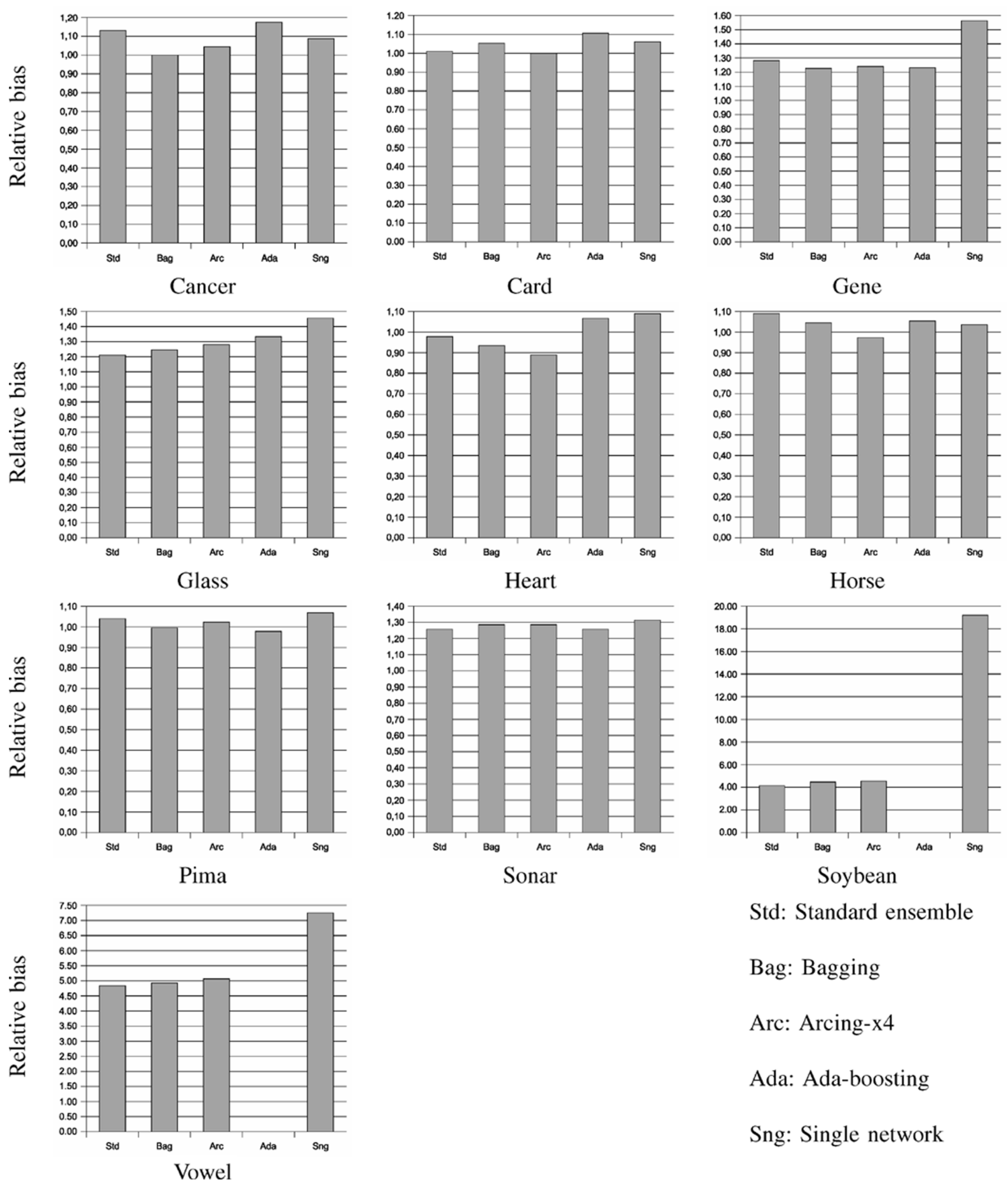

Fig. 5. Comparison of relative bias as defined by Kong and Dietterich [90] for the ten data sets. A value of 1 has been chosen for the estimation obtained for the cooperative ensemble bias, and we represent the relative bias of the rest of the ensemble methods to the cooperative ensemble. The y axis represents this relative bias.

tendency. More informally, bias is the portion of classifications that are incorrect and equal to the central tendency, and variance is the portion of classifications that are incorrect and differ from the central tendency.

Fig. 6 shows that the cooperative ensemble improves the bias error in some problems. Nevertheless, there are no significant differences in bias error between the cooperative ensemble and the other methods in Cancer, Card, Heart, Horse, and Pima prob-

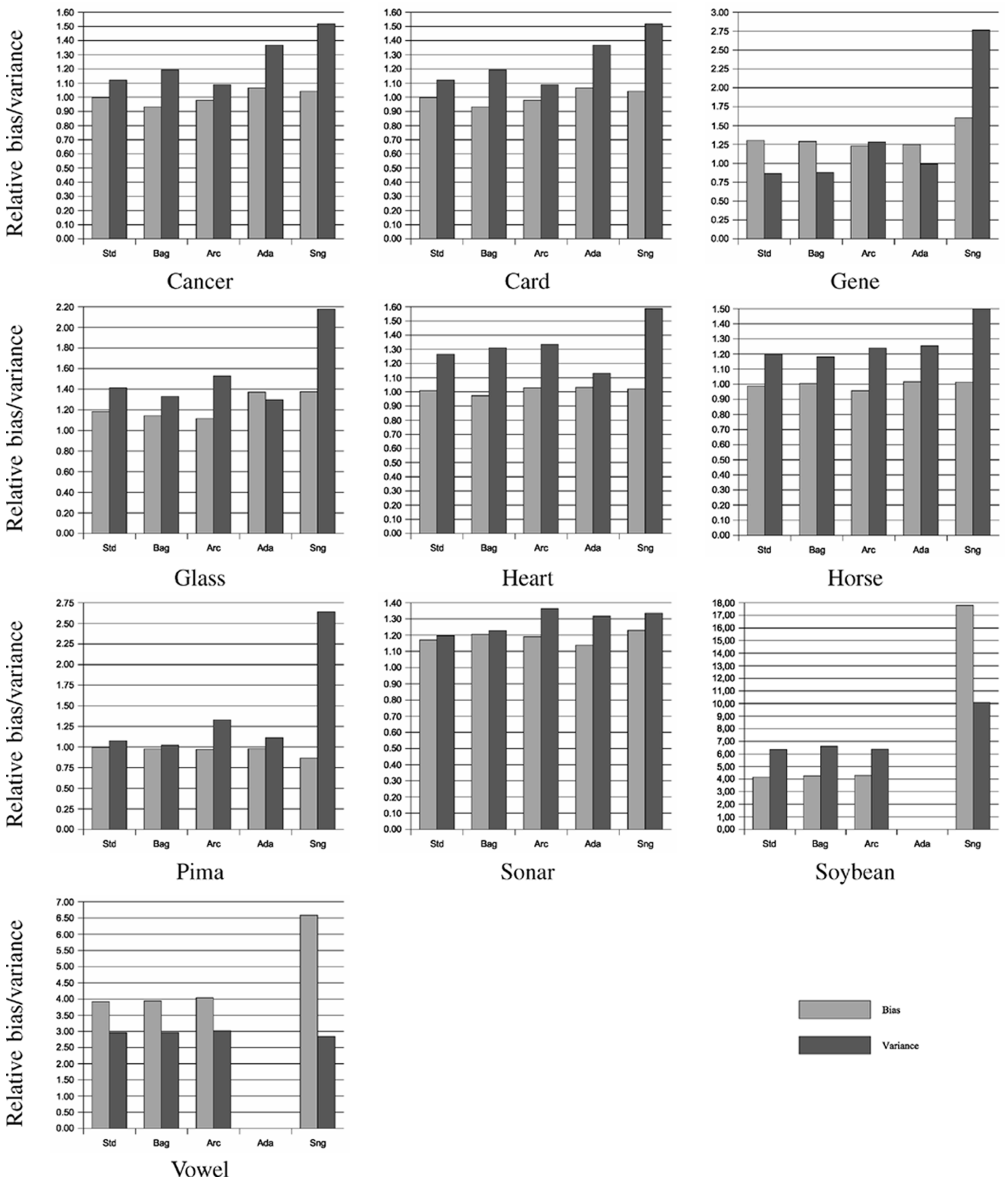

Fig. 6. Comparison of relative bias and variance as defined by Kohavi and Wolpert [91] for the ten data sets. A value of 1 has been chosen for the estimation obtained for the cooperative ensemble, and we represent the relative bias and variance of the rest of the ensemble methods to the cooperative ensemble. The y axis represents this relative bias and variance.

lems. We must also take into account that the *irreducible error* is included in the bias estimation in the model of Kohavi and Wolpert. On the other hand, the variance is reduced in almost all the problems. This is important, as it means that the algorithm is less sensitive to variations in the training set, assuring a more robust learning process. We also notice, as can be expected, that Bagging is more successful in reducing variance and Ada-Boost reduces bias more frequently.

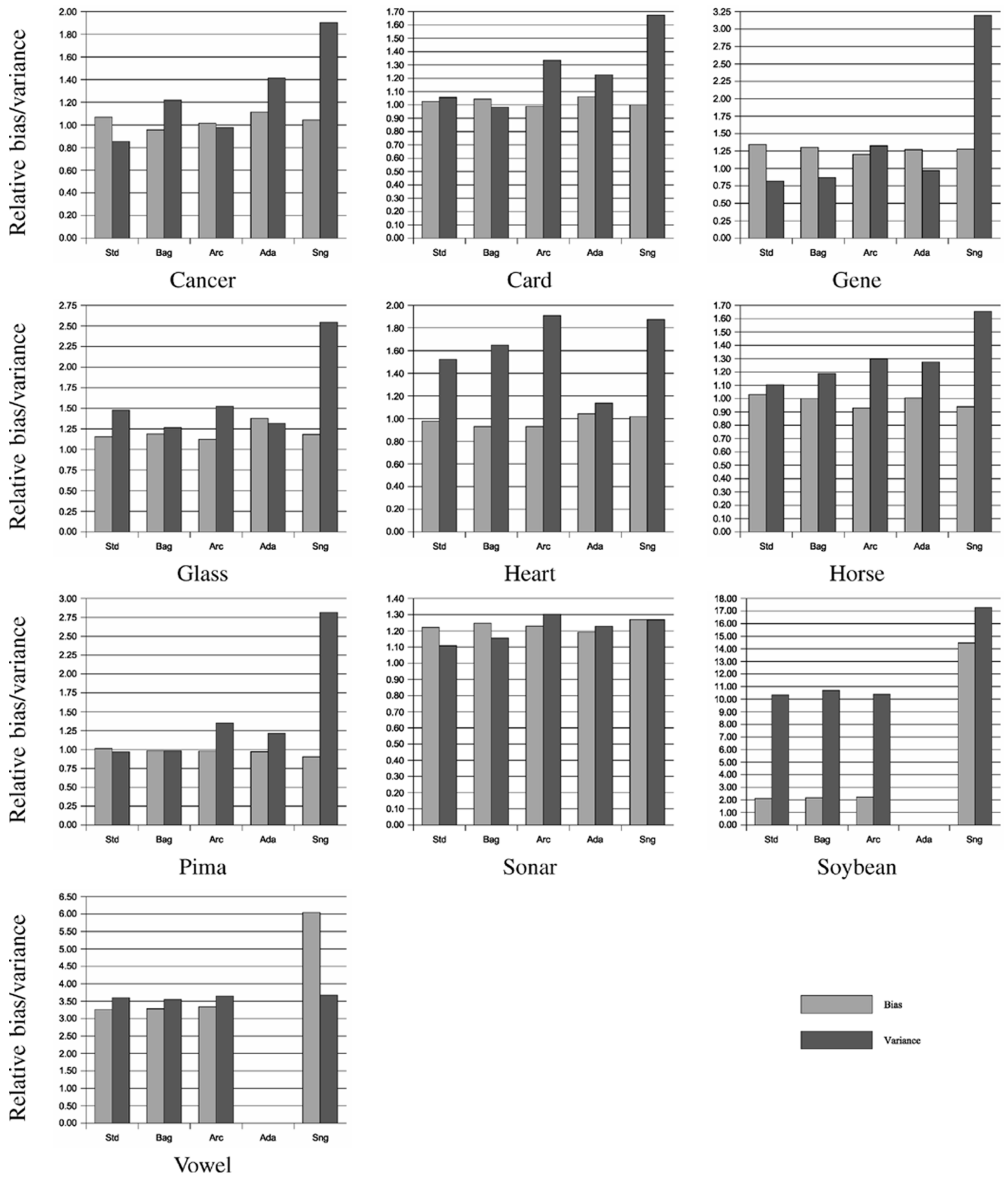

Fig. 7. Comparison of relative bias and variance as defined by Breiman [92] for the ten data sets. A value of 1 has been chosen for the estimation obtained for the cooperative ensemble, and we represent the relative bias and variance of the rest of the ensemble methods to the cooperative ensemble. The y axis represents this relative bias and variance.

Fig. 7 shows the relative bias and variance of Breiman. Breiman estimation is more closely related with the intuitive idea of bias and variance. In this estimation, irreducible error is shared by the two terms. Considering this estimation, the figure shows how the cooperative ensemble reduces both bias and variance terms of error. This accomplished reduction allows us to say that the cooperative ensemble is both accurate in its central tendency and little responsive to the variations of the training set.

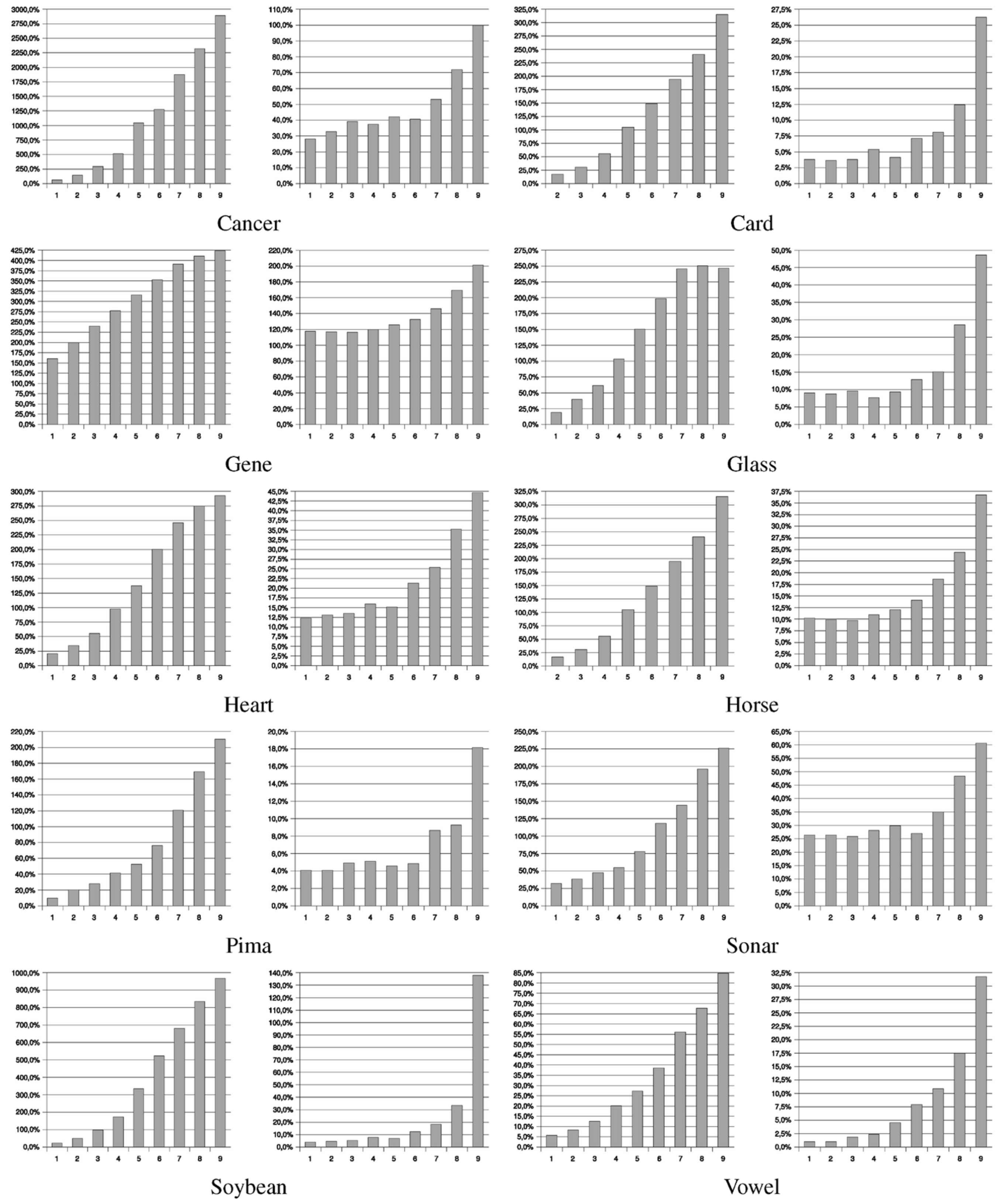

Fig. 8. Lesion study for the ten problems. It shows the average percentage of increment in generalization error when the best network is removed, on the left, or the worst network is removed, on the right. The x axis represents the number of networks removed from the ensemble, the y axis represents the increment of the generalization error in percentage when the corresponding networks are removed.

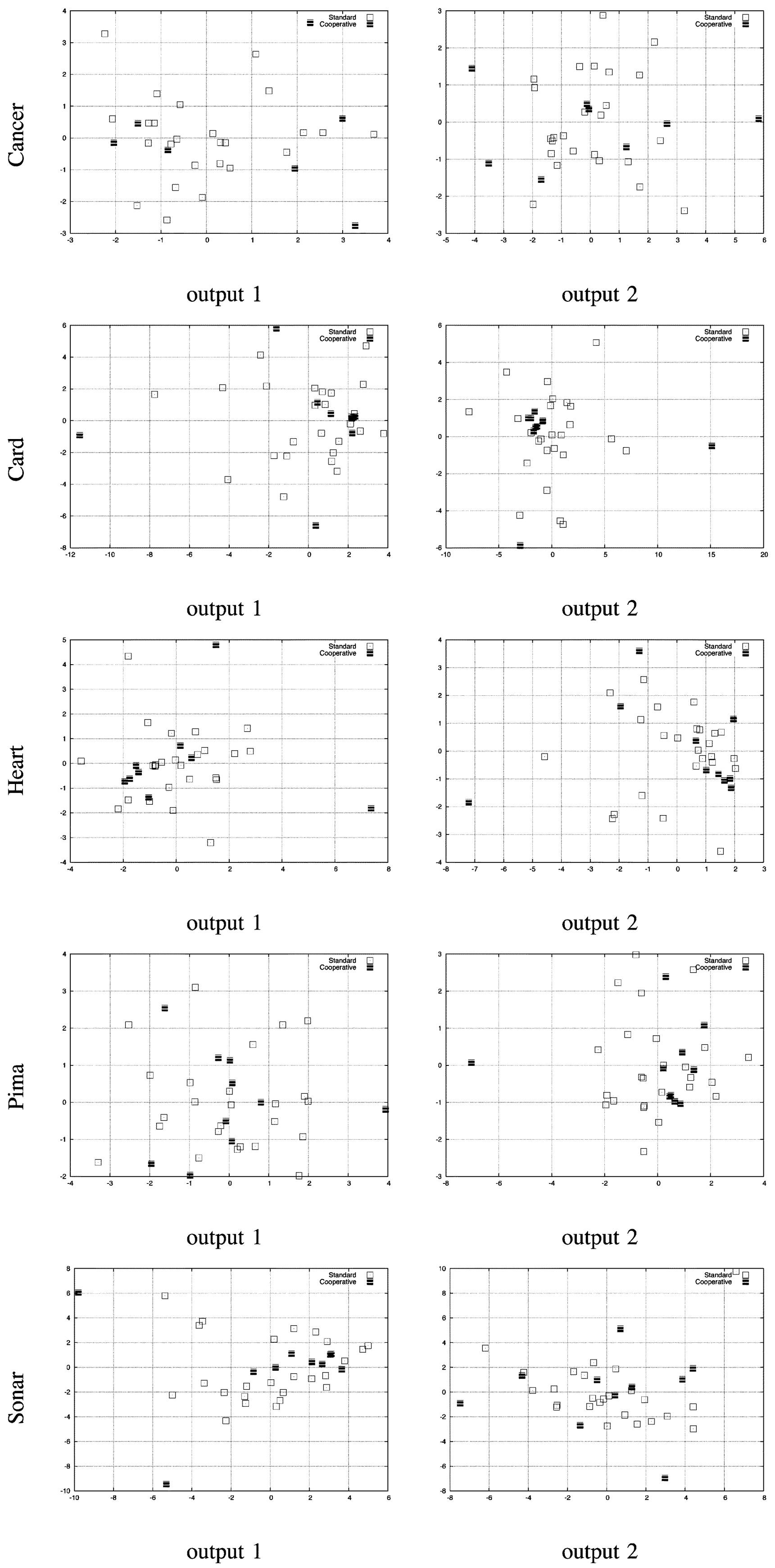

Fig. 9. First two principal components for every network of the best ensemble of the first run for cancer, card, heart, pima, and sonar problems.

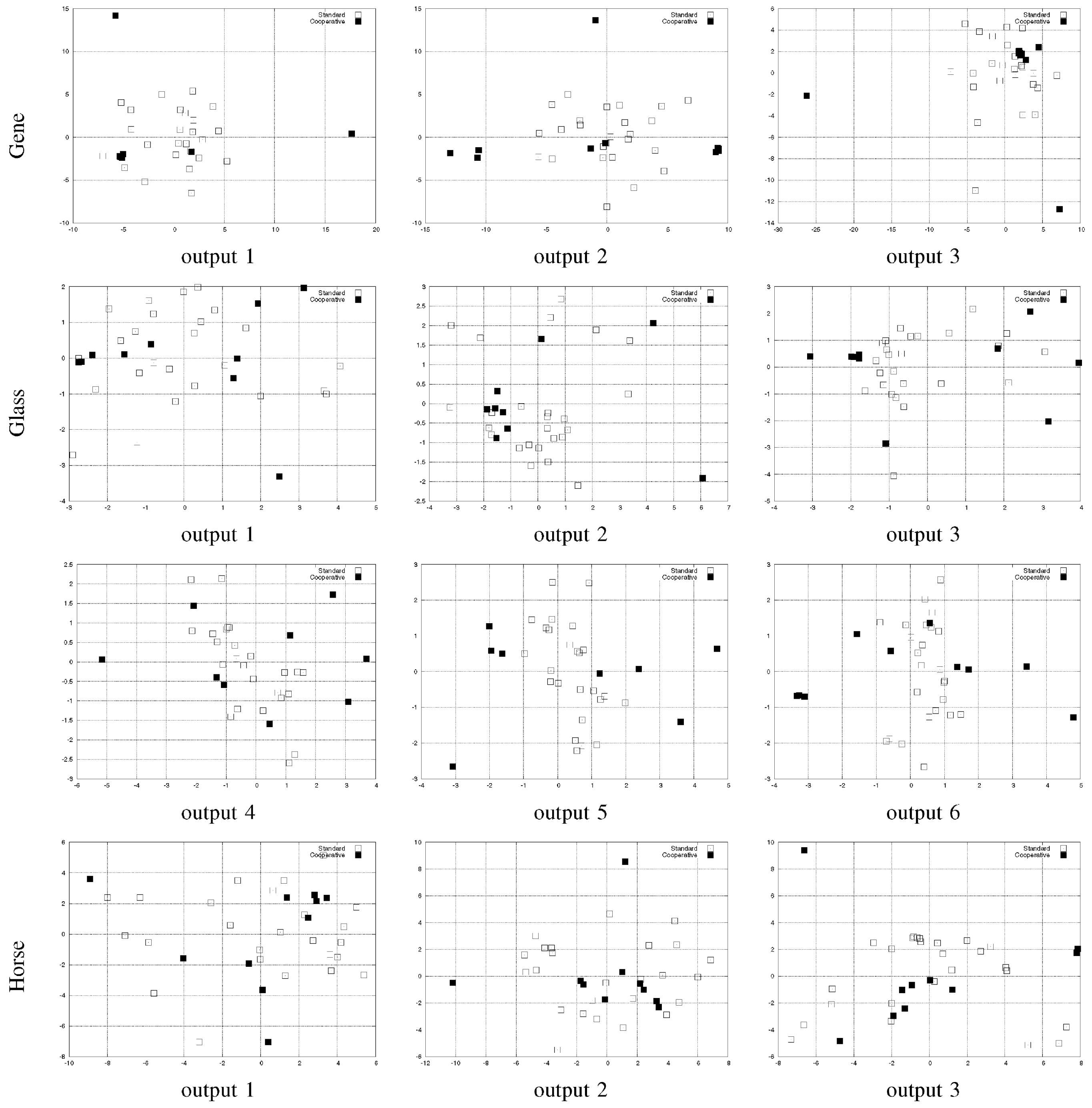

Fig. 10. Two first principal components for every network of the best ensemble of the first run for gene, glass, and horse problems.

D. Lesion Study

Lesions are used in biological networks to identify functions or functional areas of the brain. Previous works [74] have used lesions to study the functionality of different parts of an evolved network. Here, we use lesions for testing the robustness of the ensembles. Once all the networks of the ensemble are evolved, we test the performance penalty of removing each network. In this way, we can evaluate the robustness of the ensemble if any of its constituent networks ceases to work.

We have carried out two different experiments. In a first experiment, we have in turn the network that most contributes to the performance of the ensemble, until the ensemble is just one network. In a second experiment, we repeated the previous steps removing the network that contributes the last to network performance. Fig. 8 shows the average generalization error over the 30 runs for the two experiments for the ten data sets.

Fig. 8 shows a smooth degradation on the performance of the ensemble as the best networks are removed. Moreover, the effect of removing the worst network causes less damage, and this assures quite a robust ensemble. As could be expected, the effect of removing a network is more important in complex problems, such as Gene, Soybean, or Horse.

X. Why Does the Cooperative Method Outperform Standard Ensembles?

The experiments have shown that the cooperative ensemble is able to perform better than the standard ensemble methods. In the previous sections, we have gained some insights into the way the cooperative ensemble works. In this section, we want to study the features of the cooperative ensemble that would help to explain the reason why the performance of the cooperative ensemble is above the performance of the standard one.

The explanation for this performance is not an easy task, so our objective is to make explicit some of the features of cooperation rather than to carry out an exhaustive study that is outside the scope of this paper.

A. Principal Components Analysis

Our first objective in this study is the visualization of the functionality of the networks. It is a well-known issue that as the networks are more *different*, the performance of the ensemble is increased. In order to show the functionality of a network, we follow the functional representation of a network (or node) of Moriarty and Miikkulainen [12]. In order to obtain the functional representation of a network, we calculate for each network its *function vector*. The steps to compute this vector are the following.

1) Initialize the function vector to nil.
2) For each input unit i of the network do:
   a) Set input i to 1 and all the others to 0.
   b) Propagate the activation through the network.
   c) Append the output of the network to the function vector.

In this way, we produce a function vector for each output of the network. In order to represent the functionality of the network, we perform a principal component analysis of the function vectors, retaining the first two components. Following this method, we can represent each network by a point in a two-dimensional (2-D) space. The position of each network depends on its functionality, and we can consider that if two networks are closed in the 2-D space their functionality is somewhat similar. On the other hand, if two networks are separated, their functionality must be very different. Figs. 9 and 10 show the representation of the networks of the best ensemble of the first run of the algorithms for all the problems, except for soybean and vowel that are not represented due to their large number of outputs. For the standard ensemble, we have chosen the model that best performs for each problem.

Figs. 9 and 10 show how the networks in the standard ensembles are more clustered, with many networks sharing similar functionality. On the other hand, the networks of the cooperative ensemble tend to be more spread, so their collaboration is more effective.

In order to corroborate the previous affirmation, we have measured the dispersion of the networks represented on the figures. Table XIII shows the average distance of each network from the centroid of the cluster made up of all the networks of the ensemble. This value can be considered a measure of the dispersion of the networks in the ensemble [93]. In the table, we show the average value of all the outputs. With n networks, each one represented by a two dimensional vector $\mathbf{x}_i$ the average distance is given by

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_{i1}-m_1)^2+(x_{i2}-m_2)^2} \tag{35}$$

where $\mathbf{m}$ is the mean of the vectors that represent the networks in the ensemble. The table shows that the average distance of the networks is greater among the cooperative ensemble than the standard ensemble. This means that the networks have more varied behavior, and the efficiency of their combination should be better. Table XIII also shows the p-values of a t-test that compares whether the differences between the means are significant. With a confidence level of 5%, the differences are significant for Cancer, Heart, Gene, and Glass problems.

TABLE XIII
Average Distance From the Centroid of the Networks of the Ensemble

	Cooperative ensemble	Standard ensemble	t-test
Cancer	2.3313	1.7189	0.0223
Card	3.6876	3.1214	0.2048
Heart	2.5383	1.8328	0.0311
Pima	1.7762	1.7390	0.4497
Sonar	3.9832	3.2953	0.1418
Gene	8.0168	4.0682	0.0000
Glass	2.5015	1.6256	0.0000
Horse	4.2360	4.2757	0.4685

B. Measures of the Diversity of the Individual Networks

In order to complete the previous study of the diversity of the networks in the ensembles, we obtain the error correlation among the networks that make up the standard and cooperative ensembles. The values are shown in Table XIV. We have also obtained the value of Yule's Q statistic (Table XIV) in order to test the similarity among the classifications performed by each network. As in the previous study, for the standard ensemble we have chosen the model that best performs for each problem. The t-tests show that all the differences are significant.

Table XIV shows that, as a general rule, the correlation among the errors of the individuals networks in the cooperative ensemble is below the correlation among the networks that form the standard ensemble. There are two exceptions, Cancer and Pima problems. For these two problems, the standard ensemble shows a smaller correlation. The value of Yule's Q has the same property. This result, together with the results from principal component analysis, suggests that one of the sources of the excellent behavior of the cooperative ensemble comes from the

TABLE XIV
MEASURES OF THE DIVERSITY OF THE INDIVIDUAL NETWORKS OF THE ENSEMBLES

	Cooperative ensemble		Standard ensemble		t-test	
	Error correlation	Yule's Q	Error correlation	Yule's Q	Error correlation	Yule's Q
Cancer	0.2395	0.4936	0.0866	0.2101	0.0000	0.0000
Card	0.2360	0.4849	0.5424	0.8948	0.0000	0.0000
Gene	0.1069	0.2385	0.2577	0.5472	0.0000	0.0000
Glass	0.1214	0.2028	0.2611	0.4398	0.0000	0.0000
Heart	0.2551	0.4670	0.4716	0.8148	0.0000	0.0000
Horse	0.2879	0.5314	0.3998	0.6929	0.0000	0.0000
Pima	0.2622	0.4671	0.1188	0.2236	0.0000	0.0000
Sonar	0.1886	0.3804	0.4043	0.7341	0.0000	0.0000
Soybean	0.1716	0.3701	-	-	-	-
Vowel	0.1329	0.2646	0.2291	0.5152	0.0000	0.0000

TABLE XV
CLASSIFICATION OF PATTERNS BY THE NETWORKS THAT MADE UP THE COOPERATIVE AND STANDARD ENSEMBLES

	Cooperative ensemble		Standard ensemble		t-test	
	Average class.	No classiÞed	Average class.	No classiÞed	Average class.	No classified
Cancer	0.8744	0.0002	0.7453	0.0000	0.0000	0.3215
Card	0.7582	0.0078	0.8363	0.0277	0.0000	0.0000
Gene	0.6418	0.0042	0.7420	0.0039	0.0000	0.4880
Glass	0.4916	0.0390	0.5235	0.0533	0.0025	0.0016
Heart	0.7472	0.0098	0.7994	0.0324	0.0000	0.0000
Horse	0.6206	0.0271	0.6507	0.0326	0.0000	0.1086
Pima	0.6991	0.0108	0.6389	0.0000	0.0000	0.0000
Sonar	0.7060	0.0029	0.7690	0.0080	0.0000	0.0077
Soybean	0.6100	0.0059	0.0125	0.9153	0.0000	0.0000
Vowel	0.3309	0.0836	0.0950	0.4094	0.0000	0.0000

fact that the cooperation is able to obtain more diverse networks than the standard methods for constructing ensembles.

C. *Classification of Individual Patterns*

The last study of the networks of the ensembles concerns the patterns that each network classifies accurately. For all the 30 runs of the cooperative and standard models, we took the best ensemble and obtained the following two values.

1) For each ensemble, we obtained the average number of patterns correctly classified by its constituent networks considered alone. This value was average for the best ensemble of all the 30 runs.
2) For each ensemble, we obtained the percentage of patterns that were not correctly classified by any of its constituent networks. This is a measure of how well the networks *cover* the training set.

These two values for all the problems are shown in Table XV. The results are very interesting as they show some differences between the behavior of the standard and cooperative ensemble's networks. The following differences can be noted.

- As a general rule, the networks of the cooperative ensemble perform worse than the networks from the standard ensemble. This means that the former are more *local* in their behavior, instead of trying to classify every pattern, they specialized in some subsets of the patterns.
- In most cases, the networks of the cooperative ensemble left fewer patterns inaccurately classified by all the networks. The number of patterns that are incorrectly clas-

sified by all networks is low, even for complex problems. In this way, the combination of the classifiers can be more effective.

The t-tests show that the differences are significant for all comparisons in the case of average classification, and for all comparisons in no classification, except Cancer, Gene, and Horse problems.

XI. Conclusion

In this paper, we have proposed a novel approach to ensemble design based on cooperative coevolution. The proposed model is a framework for all the steps of the design and training of network ensembles. The simultaneous evolution of all the networks that form the ensemble allows us to obtain more cooperative networks with a performance significantly above the performance of classic ensemble methods.

The performance has been thoroughly tested over a set of ten real-world problems with different features. Our model has proved an excellent performance in the solution of these real-world problems. The performance obtained is among the best present in the literature. We have also performed an extensive analysis of the behavior of the model in different scenarios.

Additional experiments have shown that the cooperative ensemble reduces the variance term of the error dramatically. This feature assures a learning algorithm less dependent on the training data set. The experiments have also shown a smooth degradation of the performance of the ensemble, when its networks are removed.

The multiobjective evaluation of the fitness of the networks introduces the possibility of enforcing several aspects of the networks that are interesting for a better performance of the ensembles. In this work, we have proposed a set of general objectives that can be applied for any problem. However, the definition of other sets of objectives that may be adequate for a given problem might improve the performance of the model.

Recent works [24] have stated that it is not clear that the use of diversity terms has a beneficial effect over the ensemble. Our results partially agree with this statement, as the performance of the model is not clearly improved, when the defined diversity objective is considered.

A. Future Work

The results of our model in classification greatly encourages a continuation of our research in cooperative coevolution of ensembles. One of the most natural continuations of out work is the application of the ideas of this paper to consensual networks [94].

It is also possible that the evolution of different kinds of networks on each subpopulation may generate more diverse populations with a potentiality for forming better ensembles.

Arcing and Ada-Boosting methods also suggest the possibility of developing an incremental cooperative environment where new subpopulations are added when the evolution stagnates. The new subpopulations would be added following the Arcing or Ada models, that is, focusing their attention on the patterns that are less easily classified by the previous subpopulations.

Acknowledgment

The authors would like to thank R. Moya–Sánchez for her helping in the final version of this paper.

References

[1] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," in *Neural Networks for Speech and Image Processing*, R. J. Mammone, Ed. London, U.K.: Chapman & Hall, 1993, pp. 126–142.

[2] S. Haykin, *Neural Networks – A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice–Hall, 1999.

[3] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. 1st Int. Workshop Multiple Classifier Syst.*, J. Kittler and F. Roli, Eds., 2000, pp. 1–15.

[4] G. I. Webb, "Multiboosting: A technique for combining boosting and wagging," *Mach. Learn.*, vol. 40, no. 2, pp. 159–196, Aug. 2000.

[5] S. Dzeroski and B. Zenko, "Is combining classifiers with stacking better than selecting the best one?," *Mach. Learn.*, vol. 54, pp. 255–273, 2004.

[6] C. J. Merz, "Using correspondence analysis to combine classifiers," *Mach. Learn.*, vol. 36, no. 1, pp. 33–58, Jul. 1999.

[7] A. Fern and R. Givan, "Online ensemble learning: An empirical study," *Mach. Learn.*, vol. 53, pp. 71–109, 2003.

[8] J. Hansen, "Combining predictors: Comparison of five meta machine learning methods," *Inf. Sci.*, vol. 119, no. 1–2, pp. 91–105, 1999.

[9] R. Avnimelech and N. Intrator, "Booested mixture of experts: An ensemble learning scheme," *Neural Comput.*, vol. 11, no. 2, pp. 483–497, 1999.

[10] G. Giacinto and F. Roli, "Dynamic classifier selection," in *Lecture Notes in Computer Science*, vol. 1857, Proc. Multiple Classifier Syst. 2000, 2000, pp. 177–189.

[11] A. J. C. Sharkey, "On combining artificial neural nets," *Connection Sci.*, vol. 8, pp. 299–313, 1996.

[12] D. E. Moriarty and R. Miikkulainen, "Forming neural networks through efficient and adaptive coevolution," *Evol. Comput.*, vol. 4, no. 5, pp. 373–399, 1997.

[13] P. J. Darwen and X. Yao, "Speciation as automatic categorical modularization," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 101–108, Jul. 1997.

[14] V. R. Khare, "Automatic Problem Decomposition using Co-Evolution and Ensembles," M.S. thesis, Univ. Birmingham, Birmingham, U.K., 200.

[15] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Trans. Evol. Comput.*, vol. 4, pp. 380–387, Nov. 2000.

[16] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 28, no. 3, pp. 417–425, Jun. 1998.

[17] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artif. Intell.*, vol. 137, no. 1–2, pp. 239–253, May 2002.

[18] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Evolving a cooperative population of neural networks by minimizing mutual information," in *Proc. 2001 IEEE Congr. Evol. Comput.*, Seoul, Korea, May 2001, pp. 384–389.

[19] B. Bakker and T. Heskes, "Clustering ensembles of neural network models," *Neural Netw.*, vol. 16, no. 2, pp. 261–269, Mar. 2003.

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

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

[22] ——, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 6, pp. 716–726, 1999.

[23] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, ser. Wiley series in telecommunication. New York: Wiley, 1991.

[24] L. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learn.*, vol. 51, no. 2, pp. 181–207, May 2003.

[25] 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.

[26] K. Tumer and J. Ghosh, "Analysis of decision boundaries in linearly combined classifiers," *Pattern Recognit.*, vol. 29, no. 2, pp. 341–348, 1996.

[27] ——, "Error correlation and error reduction in ensemble learning," Univ. Texas, Austin, TX, Tech. Rep. TX 78712–1084, 1996.

[28] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison–Wesley, 1989.
[29] 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.
[30] N. García-Pedrajas, C. Hervás-Martínez, and J. Muñoz-Pérez, "COVNET: A cooperative coevolutionary model for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 14, pp. 575–596, May 2003.
[31] ——, "Multiobjective cooperative coevolution of artificial neural networks," *Neural Netw.*, vol. 15, no. 10, pp. 1255–1274, Nov. 2002.
[32] P. M. Williams, "Bayesian regularization and pruning using a Laplace prior," *Neural Comput.*, vol. 7, pp. 117–143, 1995.
[33] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer–Verlag, 1994.
[34] T. Caelli, L. Guan, and W. Wen, "Modularity in neural computing," *Proc. IEEE*, vol. 87, pp. 1497–1518, Sep. 1999.
[35] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 8, pp. 694–713, May 1997.
[36] Y. Liu and X. Yao, "Negatively correlated neural networks for classification," in *Proc. 3rd Int. Symp. Artificial Life Robotics (AROBIII'98)*, vol. 2, M. Sugisaka, Ed., Beppu, Japan, 1998, pp. 736–739.
[37] M. A. Potter, "The design and analysis of a computational model of cooperative coevolution," Ph.D. dissertation, Goerge Mason Univ., Fairfax, VA, 1997.
[38] P. J. Werbos, *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. New York: Wiley, 1994.
[39] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 5, pp. 54–65, Jan. 1994.
[40] D. B. Fogel, "Evolving artificial intelligence," Ph.D. dissertation, Univ. California, San Diego, CA, 1992.
[41] Y. LeCun, L. Bottou, G. B. Orr, and K.-R Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, G. B. Orr and K.-R. Müller, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 9–50.
[42] D. Whitley and J. Kauth, "GENITOR: A different genetic algorithm," in *Proc. Rocky Mountain Conf. Artif. Intell.*, Denver, CO, 1988, pp. 118–130.
[43] D. Whitley, "The GENITOR algorithm and selective pressure," in *Proc 3rd Int. Conf. Genetic Algorithms*, M. K. Publishers, Ed., Los Altos, CA, 1989, pp. 116–121.
[44] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 2–9.
[45] D. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*, G. Rawlins, Ed. Sasn Mateo, CA: Morgan Kaufmann, 1991, pp. 94–101.
[46] D. Whitley and T. Starkweather, "GENITOR II: A distributed genetic algorithm," *J. Experimental Theoretical Artif. Intell.*, pp. 189–214, 1990.
[47] G. Syswerda, "A study of reproduction in generational and steady-state genetic algorithms," in *Foundations of Genetic Algorithms*, G. Rawlins, Ed. Sasn Mateo, CA: Morgan Kaufmann, 1991, pp. 94–101.
[48] M. Leblanc and R. Tibshirani, "Combining estimates in regression and classification," Dept. Statistics, Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 1993.
[49] L. Breiman, "Stacked regressions," *Mach. Learn.*, vol. 24, no. 1, pp. 49–64, 1996.
[50] S. Hashem, "Optimal linear combinations of neural networks," *Neural Netw.*, vol. 10, no. 4, pp. 599–614, 1997.
[51] C. J. Merz, "A principal components approach to combining regression estimates," *Mach. Learn.*, vol. 36, no. 1, pp. 9–32, Jul. 1999.
[52] J. Kivinen and M. Warmuth, "Exponential gradient descent versus gradient descent for linear predictors," *Inf. Comput.*, vol. 132, no. 1, pp. 1–63, Jan. 1997.
[53] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.
[54] A. Krogh and J. Vedelsby, "Neural networks ensembles, cross validation, and active learning," in *Advanced in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: MIT Press, 1995, pp. 231–238.
[55] R. Meir, "Bias, variance and the combination of estimators; The case of linear least squares," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds. Cambridge, MA: MIT Press, 1995, vol. 7.
[56] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Mach. Learn.*, vol. 36, no. 1/2, pp. 105–142, Jul./Aug. 1999.
[57] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[58] J. R. Quinlan, "Bagging, boosting, and c4.5," in *Proc. 13th Nat. Conf. Artif. Intell.*, 1996, pp. 725–730.
[59] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, Bari, Italy, 1996, pp. 148–156.
[60] L. Breiman, "Arcing classifiers," *Ann. Statist.*, vol. 26, pp. 801–824, 1998.
[61] P. A. N. Bosmand and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 174–188, Apr. 2003.
[62] R. Reed, "Pruning algorithms – A survey," *IEEE Trans. Neural Netw.*, vol. 4, pp. 740–747, 1993.
[63] C. Goutte and L. K. Hansen, "Regularization with a pruning prior," *Neural Netw.*, vol. 10, no. 6, pp. 1053–1059, 1997.
[64] PAD. C. Plaut, S. J. Nowlan, and G. E. Hinton, "Experiments in learning by back propagation," Carnegie-Mellon Univ., Pittsburgh, Tech. Rep. CMU-CS-86-126, 1986.
[65] M. Ishikawa, "A structural learning algorithm with forgetting of link weights," Electrotechnical Lab., Tsukuba-City, Japan, Tech. Rep. TR-90-7, 1990.
[66] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural Comput.*, vol. 4, no. 4, pp. 473–493, 1992.
[67] A. Dempster, N. Laird, and D. Rubin, "Maximum-likelihood from incomplete data via the EM algorithm," *J. Royal Statist. Soc. Ser. B*, vol. 39, no. 1–38, 1977.
[68] J. Horn, D. E. Goldberg, and K. Deb, "Implicit niching in a learning classifier system: Natures's way," *Evol. Comput.*, vol. 2, no. 1, pp. 37–66, 1994.
[69] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd Int. Conf. Genetic Algorithms*, San Mateo, CA, 1987, pp. 148–154.
[70] K. A. de Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, MI, 1975.
[71] R. J. Collins and D. R. Jefferson, "Selection in massively parallel genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, San Mateo, CA, 1991, pp. 249–256.
[72] R. T. Clemen and R. L. Winkler, "Limits for the precision and value of information from dependent sources," *Oper. Res.*, vol. 33, pp. 427–442, 1985.
[73] I. T. Jolliffe, *Principal Components Analysis*. New York: Springer–Verlag, 1986.
[74] D. E. Moriarty, "Symbiotic evolution of neural networks in sequential decision tasks," Ph.D. dissertation, Univ. Texas, Austin, TX, 1997.
[75] A. Afifi and S. Azen, *Statistical Analysis. A Computer Oriented Approach*. New York: Academic, 1979.
[76] G. Yule, "On the association of attributes in statistics," *Phil. Trans.*, vol. 194, pp. 257–319, 1900.
[77] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 117–132, Apr. 2003.
[78] K. Deb, "Evolutionary algorithms for multicriterion optimization in engineering design," in *Proc. Evol. Algorithms Eng. Comput. Sci. (EUROGEN'99)*, Jyväskylä, Finland, May/Jun. 30/3, 1999, pp. 135–161.
[79] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.
[80] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 257–271, Nov. 1999.
[81] K. D. Knowles and D. W. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evol. Comput.*, vol. 8, no. 2, pp. 149–172, 2000.
[82] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed., Los Altos, CA, 1989, pp. 42–50.
[83] L. Prechelt, "Proben1–A set of neural network benchmark problems and benchmarking rules," Universität Karlsruhe, Fakultät für Informatik, Karlsruhe, Germany, Tech. Rep. 21/94, 1994.
[84] G. Zenobi and P. Cunningham, "Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error," in *Lecture Notes in Artificial Intelligence*, vol. 2167, Proc. 12th Eur. Conf. Mach. Learn. (ECML 2001), L. de Raedt and P. Flach, Eds., 2001, pp. 576–587.
[85] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artif. Intell. Res.*, vol. 11, pp. 169–198, 1999.

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

[87] C. A. Coello-Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowl. Inf. Syst.*, vol. 1, no. 3, pp. 269–308, Aug. 1999.

[88] J. R. Koza, *Genetic Programming II. Automatic Discovery of Reusable Programs*. Cambridge, MA: MIT Press, 1994.

[89] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, 1990.

[90] E. B. Kong and T. G. Dietterich, "Error-correction output coning corrects bias and variance," in *Proc. 12th Int. Conf. Mach. Learn.*, A. Prieditis and J. F. Lemmer, Eds., 1995, pp. 275–283.

[91] R. Kohavi and D. H. Wolpert, "Bias plus variance decomposition for zero–one loss functions," in *Proc. 13th Int. Conf. Mach. Learn.*, L. Saitta, Ed., 1996, pp. 275–283.

[92] L. Breiman, "Bias, variance, and arcing classifiers," Dept. Statistics, Univ. California, , Berkeley, CA, Tech. Rep. 460, 1996.

[93] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988, Advanced Reference Series.

[94] J. A. Benediktsson, J. R. Sveinsson, O. K. Ersoy, and P. H. Swain, "Parallel consensual networks," *IEEE Trans. Neural Netw.*, vol. 8, pp. 54–64, Jan. 1997.

[95] J. Friedman, T. Hastie, and R. Tibshirani, "Additice logistic regression: A statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.

[96] M. M. Islam, X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Trans. Neural Netw.*, vol. 14, pp. 820–834, Jul. 2003.

[97] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learn.*, vol. 40, pp. 139–157, 2000.

[98] L. Breiman, "Randomizing outputs to increase prediction accuracy," *Mach. Learn.*, vol. 40, pp. 229–242, 2000.

[99] L. Todorovski and S. Dzeroski, "Combining classifiers with meta decision trees," *Mach. Learn.*, vol. 50, pp. 223–249, 2003.

[100] E. Cantú-Paz and C. Kamath, "Inducing oblique decision trees with evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 54–68, Feb. 2003.

Nicolás García-Pedrajas (M'01) was born in Córdoba, Spain, in 1970. He received the B.S. degree in computing and the Ph.D. degree from the University of Málaga, Spain, in 1993 and 2001, respectively. His dissertation research involves the automatic design of artificial neural networks using both genetic algorithms and evolutionary programming.

He is a Professor of Computer Science and Artificial Intelligence in the Department of Computing and Numerical Analysis, University of Córdoba. His current research interests include neural networks, evolutionary computation, and game playing.

Dr. García-Pedrajas is a member of several technical societies including the Association for Computing Machinery (ACM), the INNS, and the IEEE Computer Society.

César Hervás-Martínez (M'98) was born in Cuenca, Spain. He received the B.S. degree in statistics and operating research from the Universidad Complutense of 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 of Computer Science and Artificial Intelligence, Department of Computing and Numerical Analysis, University of Córdoba, Córdoba, Spain, and an Associate Professor in the Department of Quantitative Methods, School of Economics. He is a member of other technical societies. His current research interests include neural networks, evolutionary computation, and modeling of natural systems.

Domingo Ortiz-Boyer was born in Córdoba, Spain, in 1970. He received the B.S. degree in computing and the Ph.D. degree from the University of Málaga, Spain, in 1995 and 2001, respectively.

He is a Professor of Computer Science and Artificial Intelligence, Department of Computing and Numerical Analysis, University of Córdoba. His current research interests include evolutionary computation, function optimization, and nonlinear regression.