Learning Ensembles of Neural Networks by Means of a Bayesian Artificial Immune System

Pablo A. Dalbem Castro and Fernando José Von Zuben, *Senior Member, IEEE*

***Abstract*— In this paper, we apply an immune-inspired approach to design ensembles of heterogeneous neural networks for classification problems. Our proposal, called Bayesian artificial immune system, is an estimation of distribution algorithm that replaces the traditional mutation and cloning operators with a probabilistic model, more specifically a Bayesian network, representing the joint distribution of promising solutions. Among the additional attributes provided by the Bayesian framework inserted into an immune-inspired search algorithm are the automatic control of the population size along the search and the inherent ability to promote and preserve diversity among the candidate solutions. Both are attributes generally absent from alternative estimation of distribution algorithms, and both were shown to be useful attributes when implementing the generation and selection of components of the ensemble, thus leading to high-performance classifiers. Several aspects of the design are illustrated in practical applications, including a comparative analysis with other attempts to synthesize ensembles.**

***Index Terms*— Artificial immune system, Bayesian networks, classification problems, combinatorial optimization, ensemble of neural networks.**

I. Introduction

ENSEMBLE is a learning paradigm where alternative proposals, called components, combine their individual outputs to produce a unique solution to a given problem [1], [2]. Ensembles of neural networks have been widely explored in the last decade for both classification and regression problems. The main motivation is the possibility of improving the generalization capability and, consequently, the overall performance of the system. Several theoretical and empirical results have indicated that the success of the ensemble paradigm is supported by two prior conditions: their individual components must be accurate enough and must present distinct error profiles [3].

Basically, the design of ensembles can be viewed as a three-stage process: generation of a pool of candidates, selection of some of them from this pool, and combination of the individual outputs to produce a single output. Given that there is no systematic procedure to properly implement these stages in a broad range of applications, several meta-heuristics have been proposed for helping the design of ensembles.

In this context, and aiming at promoting distinct behavior for the components of the ensembles, some proposals have been reported in the literature using artificial immune systems (AISs) to generate ensembles of neural networks [4]–[10], motivated by their remarkable characteristics [11], [12]: 1) they are inherently able to maintain population diversity; 2) the size of the population at each generation is automatically adjusted according to the demands of the problem; and 3) local optimal solutions are simultaneously preserved once located.

However, the available AISs suffer from a drawback that produces a negative impact on complex optimization problems, such as generation of neural network ensembles, they are not able to deal effectively with building blocks. Given that each candidate solution may be taken as a vector of attributes, building blocks are high-quality partial components of the whole solution [13], [14]. These partial solutions are composed of specific elements of the vector of attributes. If the mutation operator does not take into account the relationship among the attributes of the problem, building blocks are not supposed to survive, being disrupted by mutation. With the preservation of building blocks, you may concentrate the search on reduced portions of the search space, and then the chances of finding the global optimum and local optima increases over time.

Recently, we have proposed a general framework to implement high-performance AISs, denoted Bayesian AIS [15], which replaces the traditional mutation operator with a probabilistic model representing the probability distribution of the promising solutions found so far. Then the obtained probabilistic model is used to generate new individuals. A Bayesian network is adopted as the probabilistic model because of its capability to properly capture the most expressive interactions among the variables of the problem. Besides the capability to deal with building blocks, BAIS still preserves the aforementioned advantages of AISs. Founded on the BAIS framework, some algorithms were developed and applied to different optimization problems [16]–[19]. In response to the particularities of each application domain, these algorithms differ in the codification scheme (responsible for the characterization of the search space), in the objective function (e.g., mono-objective or multiobjective), in the suppression mechanism, and also in every operator that depends on the kind of codification adopted. As a consequence, the BAIS of this paper is not the same as the BAIS of previous papers. It employs the same framework (core structure), but is composed of dedicated modules to design ensembles of neural networks.

Firstly, BAIS evolves a population of single-hidden-layer multilayer perceptron (MLP) networks, with alternative activation functions for each neuron, aiming at promoting diversity in performance under high accuracy of classification. The degree of diversity between the classifiers is estimated on the

Manuscript received November 20, 2009; revised November 1, 2010 and November 16, 2010; accepted November 16, 2010. Date of publication December 23, 2010; date of current version February 9, 2011.

The authors are with the Laboratory of Bioinformatics and Bioinspired Computing, Department of Electrical and Computer Engineering, University of Campinas, Campinas, São Paulo 13083-970, Brazil (e-mail: pablo@dca.fee.unicamp.br; vonzuben@dca.fee.unicamp.br).

Digital Object Identifier 10.1109/TNN.2010.2096823

1045–9227/$26.00 © 2010 IEEE

basis of their individual outputs. If two or more classifiers classify correctly the same patterns and also misclassify the same patterns, their degree of diversity is null, in spite of being distinct classification models. As soon as the networks are generated, another BAIS algorithm is applied to select subsets of classifiers from the candidates previously generated. In our approach, both stages-generation and selection of neural networks-are formulated as distinct combinatorial optimization problems with many building blocks. Experiments on 16 well-known datasets have been carried out to evaluate the effectiveness of the proposed methodology.

This paper is organized as follows. The related works are presented in Section II. Section III describes the BAIS in detail. In Section IV, we depict the application of BAIS to both generate and select components of ensembles of neural networks. The experimental results are outlined and analyzed in Section V. Finally, in Section VI we draw some concluding remarks and present the further steps of the research.

II. Related Works

As stated in Section I, one important feature to be explored in the construction of ensembles is the diversity among the components. Brown *et al*. [20] presented a taxonomy of methods for achieving diversity, divided into three branches: 1) starting point in hypothesis space; 2) set of accessible hypotheses; and 3) hypothesis space traversal.

In the first category, the diversity is obtained by creating different initial settings, such as generating artificial neural networks with distinct initial synaptic weights. Since usually the resulted components are not diverse enough, the methods are generally not going to achieve good performance.

The second category includes methods that promote diversity by manipulating either the training dataset (e.g., bagging [21], boosting [22], random subspace [23]) or structural aspects of the components, such as: 1) type of the hypothesis model (e.g., neural networks with distinct topologies); 2) flexibility of the single hypothesis model considered (e.g., number of hidden nodes in MLP neural networks); and 3) type of basis function in composite models (e.g., neurons with distinct activation functions in MLP neural networks).

Finally, the methods belonging to hypothesis space traversal are subdivided into two groups: penalty methods and evolutionary methods. The former consists of methods that add a penalty term in the error function aiming at producing biased components whose errors tend to be negatively correlated. The negative correlation learning (NCL), proposed by Liu *et al*. [24], [25], is the most representative algorithm in this category. Several works were proposed based on NCL. Liu *et al*. [26] present an ensemble learning algorithm based on NCL and evolutionary computation. Islam *et al*. [27] present a constructive approach to build ensembles of neural networks trained using NCL. Chen *et al*. [28], [29] have investigated the overfitting problem in NCL and have proposed an additional regularization term in the ensemble, giving origin to the regularized NCL (RNCL) algorithm. Regarding the second group, evolutionary methods evolve a population of neural networks using a mechanism to maintain diversity in the population, such as fitness sharing. The entire population at the final generation, or a subset of individuals chosen by some selection criterion, is then considered as candidates to compose the ensemble.

In our proposal, the diversity is achieved by the hybridization of part of the aforementioned methods. An AIS evolves not only the topology of the neural networks, but also the more proper activation function per neuron. Besides, the algorithm tends to generate neural networks whose errors are negatively correlated.

AIS is a computational paradigm inspired by the immunological system of vertebrates and designed for solving a wide range of problems, such as optimization, clustering, and pattern recognition [30], [31]. Recently, considerable efforts have been reported in the literature advocating the generation of ensembles by means of AISs. Some comparative experiments carried out in these proposals have indicated that immune-inspired algorithms are competitive approaches endowed with inherent abilities to promote and preserve diversity in the population of candidates to compose the ensemble, and also with mechanisms to automatically control the population size along the search. In what follows, we briefly outline some proposals that apply AIS to generated ensembles.

García–Pedrajas and Fybe [4], [5] developed an AIS to design ensembles of MLP neural networks for classification problems. The algorithm evolves both the structure of the networks and the weight vectors. However, all neurons utilize the same activation function. Comparative experiments on several real-world classification problems have indicated that the immune-inspired algorithm presents an overall advantage over traditional methods, such as bagging, boosting, and random subspace.

Pasti *et al*. [6]–[8] applied two immune-inspired algorithms, namely opt-aiNet [32] and omni-aiNet [33], to train MLPs to compose an ensemble of classifiers. In their work, only the weight vector are optimized while the network's architecture is predefined and remain fixed throughout the optimization process. The results show that ensembles generated by the immune algorithms contain more diverse components than that produced by other bioinspired approaches, such as genetic algorithm and particle swarm optimization.

Barbosa *et al*. [9] proposed an algorithm based on the clonal selection theory for generating ensembles of MLPs to solve regression problems. The algorithm evolves the architecture and the weight vector of the networks.

Zhang *et al*. [10] applied an AIS only to select the best components of an ensemble, from a given set of support vector machines [34] created using the bagging method.

Castro *et al*. [35] developed an AIS based on clonal selection and immune network theories for constructing ensembles of fuzzy-rule-based systems. Each candidate solution denotes a whole fuzzy rule base. At the end of the execution of the algorithm, the best solutions are taken as components of the ensemble.

Although AISs have been applied successfully to ensemble learning, some aspects of their functioning can be improved aiming at an expressive gain in performance. As the complexity and scale of the problem handled by the algorithm

increase, the performance becomes more and more associated with a proper choice of the design parameters. Moreover, since many problems can be decomposed into subproblems, it is noticeable that antibodies (candidate solutions) in the population will contain partial solutions to the global problem. These partial solutions are called building blocks. The existing AISs suffer from the lack of ability to identify and manipulate these interactions among the variables of the problem. One way to allow AISs to handle building blocks effectively is to make the algorithm learn the relationships among the variables by using statistical information extracted from the entire set of promising solutions.

Regarding the inclusion of a probabilistic model to improve the performance of the AIS, it is interesting to notice the difference between BAIS and the algorithm proposed by Wu and Fyfe [36]. Their algorithm utilizes cross-entropy to perform cloning and mutation. The cross-entropy method can achieve the global optimum by defining a family of probability density functions, and the algorithm makes adaptive changes to the probability density function according to the Kullback–Leibler cross-entropy. In brief, they have an iterative algorithm working on a probability density function with a specific set of parameters. They update the parameters at each iteration in order to make the event in which we are interested more likely at the next iteration. Consequently, their algorithm encourages convergence to similar solutions.

On the other hand, BAIS replaces the cloning and mutation operator with a probabilistic model, specifically a Bayesian network, capable of capturing the relationship among the variables of the problem and dealing effectively with building blocks. Different from the algorithm proposed in [36], BAIS discards similar solutions and insert new ones in order to maintain diversity in the population.

III. BAIS

Aiming at alleviating the shortcomings of traditional AISs, we have proposed recently a novel immune-inspired algorithm which has the mutation and cloning operators replaced by a probabilistic model in order to generate new antibodies [15]. From a conceptual point of view, the selected set of promising solutions can be interpreted as a sample drawn from an unknown probability distribution. An estimation of this probability distribution would allow the immune algorithm to generate new solutions that are somehow similar to the ones contained in the original selected solutions. Additionally, a probability distribution can effectively capture a building block structure of a problem, even without any prior information. The probabilistic model used in our proposal is a Bayesian network due to its capability to properly represent the most expressive interactions among the variables.

The pseudo-code of the proposed algorithm, called BAIS, is presented in Algorithm 1. In BAIS, the initial population is generated at random. From the current population, the best solutions are selected. A Bayesian network that properly fits the selected antibodies is constructed. A number of new antibodies sampled from the network are then inserted into the population and those similar to and with lower fitness than the selected ones are eliminated. Next, few individuals are generated randomly and inserted into the population in order to favor diversity, yielding a better exploration of the search space.

Algorithm 1 BAIS

```
Begin
  Initialize the population of antibodies;
  While stopping condition is not met do
    Evaluate the population;
    Select the best antibodies;
    Build the Bayesian network;
    Sample new antibodies;
    Suppress antibodies;
    Insert new antibodies randomly;
  End while
End
```

Besides the capability to maintain diversity in the population, BAIS also performs multimodal optimization, adjusts dynamically the size of the population according to the problem, and most important, identifies and manipulates building blocks.

As a direct consequence of these characteristics, the search process becomes more robust in the sense that the variation in performance over multiple re-executions is reduced, when compared with approaches not based on estimations of distributions. In addition, the number of function evaluations until the algorithm reaches a certain level of performance is considerably reduced.

Two aspects should receive special attention. The first one is related to the way of building the Bayesian network from the selected individuals and the other one is how to use the network to generate new solutions. In the next section we explain in detail how to perform these two tasks.

A. Bayesian Network: Learning and Sampling

Notice that these two tasks, i.e., learning and sampling, are fundamental to the BAIS algorithm. Learning the Bayesian network for a given set of promising solutions corresponds to estimating their joint distribution. Sampling new instances according to the network leads to new candidate solutions to the problem.

Formally, a Bayesian network for a set of variables $X = \{X_1, X_2, \ldots, X_n\}$ is a directed acyclic graph whose nodes are variables of the problem and the edges indicate relationships of dependence among the connected variables [37], [38]. For instance, if there is an edge from node X_1 to node X_2, we say that variable X_1 is parent of variable X_2 and, therefore, the value of X_2 is in a conditional dependence on the value of X_1. If a variable X_i has a set of parents, denoted here by π_{X_i}, its probability distribution is said to be conditional and is expressed by $P(X_i|\pi_{X_i})$. On the other hand, the probability distribution of a variable X_j which has no parents is expressed by its unconditional distribution $P(X_j)$.

1) Learning: The Bayesian network learning from a dataset can be stated as follows. Given a collection of observed data, find the network model to explain these data with maximum

likelihood. By finding the network we mean to provide the structure of the graph as well as the probability distribution of each variable that best fits the data.

One usual approach to this task is to adopt a procedure for searching the space of all possible candidate network structures, given a metric that can provide a relative score to each point in the space. Thus, the problem of learning a Bayesian network reduces to the problem of searching for a model that yields the highest score, given the observed data.

The heuristic search utilized by BAIS begins with an empty network, i.e., with no edges. Next, the probability distribution of each variable is estimated using the dataset and the score of the network is computed. The search process proposes small changes to the structure in order to obtain a network with higher score than the previous one. These small changes can be accomplished by adding or deleting an edge, or reversing the edge direction. Various algorithms can be used as the search engine. Due to its effectiveness in this context, BAIS applies the well-known hill climbing algorithm.

Regarding the scoring metrics, there are several measures proposed in the literature. Most of them evaluate a structure S taking into account the likelihood. BAIS utilizes the so-called K2 metric, proposed by Cooper and Herskovits [39]. Given a Bayesian network structure S and assuming the dataset D is complete (there are no missing values) and that there are no prior knowledge of the models, the likelihood takes the form

$$p(D|S) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \qquad (1)$$

where n is the number of instances, q_i denotes the number of possible instances of parents of x_i, r_i is the number of possible values of X_i, and N_{ijk} is the number of cases where X_i takes the kth value with its parents taking their jth value. To avoid round errors during the multiplication of probabilities, often the logarithm of the function is applied.

2) Sampling: Once the Bayesian network is built, we can generate new instances using the joint probability distribution $P(X) = \prod_{i=1}^{n} P(X_i|\pi_{X_i})$. To accomplish this task, the probabilistic logic sampling (PLS) algorithm [40] is chosen. PLS finds an ancestral ordering of the nodes in the Bayesian network and instantiates one variable at a time in a forward manner, i.e., a variable is not sampled until all its parents have already been sampled, taking advantage of how a Bayesian network defines a probability distribution. In BAIS, the number of sampled instances is half of the size of the current population.

IV. BAIS Applied to Learn Ensembles of Neural Networks

As already stated, the construction of an ensemble is performed by three steps: generation of candidates; selection of components; and combination of components.

The main motivation to use an ensemble is the expected gain in performance when compared with the performance of the components in isolation. This goal is achieved if the components are uncorrelated and present a good individual performance. Thus, the generation and selection of networks are the crucial steps to support the ensemble paradigm.

TABLE I
Set of Candidate Activation Functions

Function ID	Function	Expression
f_0	Null function	$y = 0$
f_1	Hyperbolic tangent	$y = \tan\mathrm{h}(x)$
f_2	Cosine	$y = \cos(x)$
f_3	Sine	$y = \sin(x)$
f_4	Hyperbolic sine	$y = \sin\mathrm{h}(x)$

Therefore, in this section we will show that BAIS is a promising tool for evolving and selecting networks to compose an ensemble once it is able to produce several diverse and high-quality solutions at a single run (a desired characteristic during the generation phase), and it can identify and preserve interactions of the variables (a relevant knowledge to be explored during the generation and selection phases).

Different from coevolutionary approaches, which both evolve the candidate networks and select the most useful components to ensemble at the same time using two populations, in our proposal these tasks are executed separately in two stages. Therefore, BAIS is applied twice. In this context, we notice that the dataset must be divided into four subsets: training, validation, selection, and testing. From the training and validation data, BAIS evolves the candidate networks. As soon as the networks are generated, BAIS is applied again to choose the networks that will compose the ensemble taking into account the selection data. Finally, the performance of the ensemble is measured using the test data.

The rest of this paper is restricted to classification problems and ensembles of MLP networks, although regression problems and other kind of components could also be considered with few modifications in the algorithm.

A. Generation of Neural Networks

In this first stage of our methodology, BAIS is responsible for evolving the most adequate MLP architecture for a given problem, whereas the weight vectors are adjusted by a second-order algorithm.

Usually, most of the proposed methodologies have concentrated on learning connection weights or network topology taking into account predefined activation functions of the same type for all neurons [41]. However, if the activation function of each hidden neuron can be properly and automatically defined, then better rates of convergence may be achieved [42], [43]. In this paper, we adjust not only the number of neurons in the hidden layer but also the kind of activation function for each neuron. Thus, we also expect that neural networks with distinct generalization capabilities can be generated.

Next, we describe the codification scheme, the fitness function, and the suppression mechanism of BAIS.

1) Coding: In the implementation of the algorithm, each candidate network is an antibody while the training dataset represents an antigen. In this paper, the topologies are restricted to single hidden layer feedforward networks fully connected. The hidden layer has at least one neuron and at

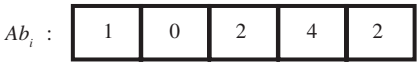

Fig. 1. Example of antibody encoding an MLP neural network.

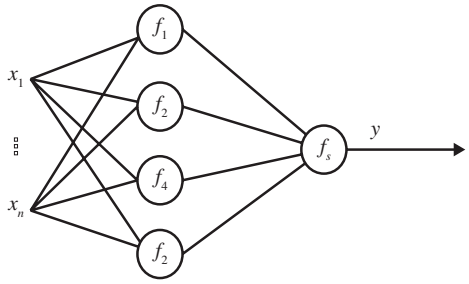

Fig. 2. Neural network with activation functions defined by the antibody of Fig. 1.

most F neurons with possibly distinct activation functions. The activation functions are selected out from a finite set of candidates, given in Table I. They represent a subset of the most frequently adopted activation functions in the literature. The null function, denoted f_0, creates the possibility of reducing the effective number of neurons in the hidden layer.

The antibodies are coded with integer numbers, denoting the index of the activation functions previously presented. For instance, suppose we have defined for a given problem that the maximum number of neurons in the hidden layer is 5, i.e., $F = 5$. Fig. 1 depicts a possible antibody for this case. The network coded in the antibody Ab_i is illustrated in Fig. 2.

Since the second activation function in the antibody Ab_i of Fig. 1 is the null function, the network topology is composed of only four neurons in the hidden layer. The activation function of the output layer, f_s, is always the logistic function and it is not considered in the codification. Note that in this example the neural network has just one output node, but could has more output nodes.

Notice that the weights of the network are not considered in the codification, since they are adjusted by means of a conjugate gradient algorithm [44].

2) Initial Population: To initialize the population, all activation functions have the same probability of being chosen, except the Null function, f_0, which has a higher probability, thus favoring more parsimonious neural networks.

3) Fitness Function: The fitness function is defined based on the performance of the neural networks, which is calculated as a function of the number of training patterns correctly classified. The fitness function is expressed by

$$Fit(Ab_i) = \frac{NPCC(Ab_i)}{NP} \tag{2}$$

where $NPCC(Ab_i)$ is the number of patterns correctly classified by the network coded in the antibody Ab_i and NP is the total number of instances in the training set.

4) Suppression: In this phase, similar antibodies are eliminated in order to avoid redundancy and thus maintain diversity. The similarity between the networks is not estimated based on their weight vectors or on their topology but using their individual input-output behavior. Thus, the permutation problem [45] does not appear.

TABLE II
OUTPUTS FOR TWO CLASSIFIERS

	Ab_m correct	Ab_m incorrect
Ab_i correct	N^{11}	N^{10}
Ab_i incorrect	N^{01}	N^{00}

We adopted the disagreement pairwise measure [46], [47], which for two classifiers, Ab_i and Ab_m, is calculated as

$$Dis_{i,m} = \frac{N^{01} + N^{10}}{N^{11} + N^{10} + N^{01} + N^{00}} \tag{3}$$

where N^{ab} is the number of instances x_j from $\mathbf{X}$ for which $y_{ji} = a$ and $y_{jm} = b$, as shown in Table II. The denominator is equal to the total number of instances. This disagreement index varies from 0 to 1, indicating similar classifiers (close to 0) or diverse classifiers (close to 1).

We considered (3) to calculate the pairwise degree of similarity between the classifiers. Individuals with a degree of similarity below a certain threshold are eliminated from the population, keeping only the ones with the highest fitness.

The diversity for a total of L classifiers is given by

$$Dis = \frac{2}{L(L-1)} \sum_{i=1}^{L-1} \sum_{m=i+1}^{L} Dis_{i,m}. \tag{4}$$

Equation (4) will be applied to assess the diversity of the whole ensemble, as described in Section V.

B. Selection of Components

As soon as the process of generating individual networks is concluded, the BAIS algorithm is applied again to propose several subsets of neural networks taken from the complete set of neural networks generated in the previous stage. Each subset will be used to implement an ensemble.

This is a combinatorial optimization problem and certainly has some building blocks. Therefore, we expect that BAIS will be able to identify and manipulate these building blocks effectively.

Next, we describe the antibody representation, the fitness function, and the suppression mechanism.

1) Coding: In this stage, each antibody $Ab_i = (net_{i1}, net_{i2}, \ldots, net_{iM})$ is coded as a sequence of binary digits of length M, where M is the number of networks generated in the previous stage. Each binary digit net_{ij} is associated with one network. If $net_{ij} = 1$, then ensemble i will have the neural network j as one of its components. Note that every antibody represents actually a specific subset of classifiers to compose the ensemble.

2) Initial Population: In this stage, one antibody is generated selecting all networks obtained in the previous stage, i.e., this antibody is a vector containing the number "1" in all positions. The remaining antibodies are generated at random.

TABLE III
CHARACTERISTICS OF THE DATASETS

Dataset	# Instances	# Attributes	# Classes
WDBC	569	30	2
Bupa	345	6	2
Ionosphere	350	34	2
Pima	768	8	2
Sonar	208	60	2
Card	690	15	2
Vote	435	16	2
Hepatitis	155	19	2
Iris	155	4	3
Wine	178	13	3
Glass	214	10	6
Satellite	6435	36	6
Letter	20000	16	26
Soybean	683	35	19
Image	2310	19	7
Vehicle	946	18	4

TABLE IV
AVERAGE PERFORMANCE OF THE BEST NEURAL NETWORK OVER 30 EXECUTIONS CONSIDERING THE TEST DATA

Dataset	Accuracy (%)	No. of neurons
WDBC	98.2 $\pm$ 0.6	5.2
Bupa	73.8 $\pm$ 1.7	3.9
Ionosphere	93.8 $\pm$ 1.3	5.2
Pima	79.1 $\pm$ 1.5	3.6
Sonar	84.7 $\pm$ 2.1	6.1
Card	83.4 $\pm$ 1.2	5.7
Vote	96.3 $\pm$ 0.9	7.3
Hepatitis	82.5 $\pm$ 1.2	8.5
Iris	97.6 $\pm$ 0.9	3.2
Wine	98.7 $\pm$ 1.4	5.4
Glass	67.3 $\pm$ 3.2	7.1
Satellite	86.9 $\pm$ 3.7	9.3
Letter	79.2 $\pm$ 4.4	7.6
Soybean	92.1 $\pm$ 1.6	5.2
Image	95.9 $\pm$ 1.8	4.8
Vehicle	77.4 $\pm$ 2.2	6.5

3) Fitness Function: The fitness function used in this stage to evaluate an antibody takes into account the performance of the whole ensemble associated with the antibody, when applied to the selection data. If the ensemble has an even number of components and half of them produce a class and the other half the other class, the classification provided by the best classifier regarding the training data will be adopted as the final output.

4) Suppression: In this stage, we adopted the Hamming distance to calculate the degree of divergence between the antibodies. The Hamming distance is normalized, i.e., divided by the length of the antibody. Individuals with a degree of divergence below certain threshold ϵ are excluded from the population.

V. COMPUTATIONAL EXPERIMENTS

This section describes the experiments carried out to evaluate the proposed algorithm. The main purposes of the experiments are: 1) to analyze the performance of BAIS when applied to generate ensembles of neural networks for classification problems, and 2) to compare the generated ensembles with those ones generated by other algorithms.

The next subsections describe the characteristics of the datasets, present the parameter values utilized during the experiments, and discuss the obtained results.

A. Description of the Datasets

During the experiments, we considered 16 well-known benchmarks from UCI Repository of Machine Learning Databases [48]: Wisconsin Diagnose Breast Cancer (WDBC), Bupa, Ionosphere, Pima, Sonar, Card, Vote, Hepatitis, Iris, Wine, Glass, Satellite, Letter, Soybean, Image Segmentation, and Vehicle. Table III summarizes the characteristics of each one, giving the total number of instances and the number of attributes. We adopted these problems because they are widely used in the literature and, therefore, we can compare the performance of our methodology with alternative approaches.

The datasets were partitioned as follows: 40% for training, 10% for validation, 25% for selection, and 25% for testing. This partitioning was performed randomly at each execution of the algorithm. For each dataset, we applied the algorithm 30 times.

B. Experimental Setup

The parameters used in our experiments are common to the 16 problems and to the two stages of the algorithm. These parameter values were obtained empirically by preliminary experiments.

The initial population contains 50 antibodies and the stopping condition of the algorithm is the maximum number of generations, defined here as 50. The maximum number of hidden neurons, F, was defined as 10 for all datasets. The number of outputs nodes varies per dataset. The optimization of the weights for all neural networks were carried out by a conjugate gradient algorithm [44].

The synthesis of the Bayesian network utilized by BAIS was performed as described in Section III-A, using the K2 metric and a hill-climbing search algorithm. In order to penalize the complexity of the model, we have imposed a constraint in the number of parents that a node can have. By our previous experience on Bayesian network learning, we know that when the complexity of the network is too high, it is more likely to detect spurious correlations on the data. Thus, each variable can have only two parents. Once the network is built, we apply the PLS algorithm to generate new individuals. At each generation, the best nb antibodies are selected to build the Bayesian network. We adopt $nb = 70\%$ of the population size. The number of samples generated is half of the size of the current population. The amount of random individuals inserted into the population in order to create diversity is around 5% of the current population size.

The suppression threshold was set to 0.10 in the first stage and to 0.50 during the second stage of the algorithm. This parameter is very important, mainly in the first stage. It has direct influence on the diversity of the components

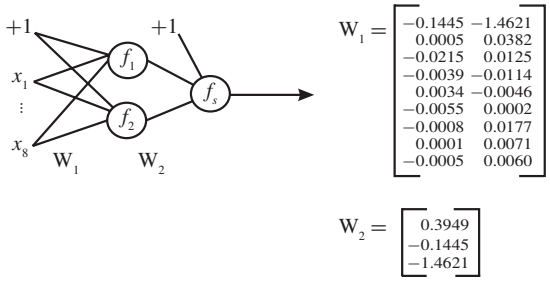

Fig. 3. Best neural network for the Pima dataset.

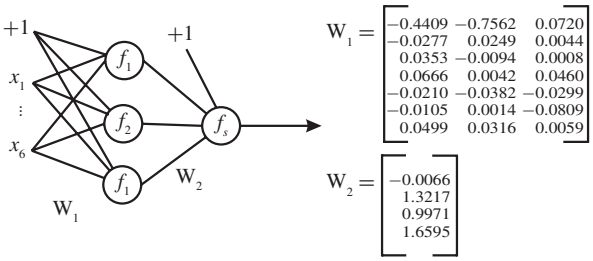

Fig. 4. Best neural network for the Bupa dataset.

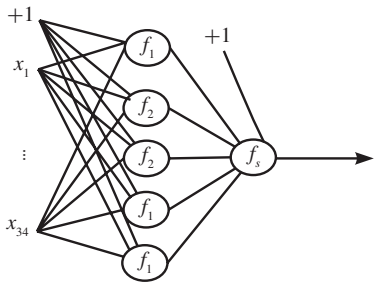

Fig. 5. Best neural network for the Ionosphere dataset.

and on the size of the population. With values smaller than 0.10, the algorithm obtained very similar classifiers. We know that similar classifiers do not contribute to the ensemble. In addition, fewer antibodies were suppressed. Consequently, the number of fitness evaluation increases. On the other hand, with values above 0.10 the algorithm is forced to search for very distinct classifiers. As a result, very good classifiers are discarded and the final population contains few classifiers with poor generalization capability.

C. Individual Performance of the Neural Networks

Firstly, we have investigated the capability of BAIS to generate accurate and diverse individual neural networks. Afterwards, we show that BAIS is able to select a subset of these networks to compose an effective ensemble of classifiers.

Table IV presents an average of the performance for the best single individual (notice we are employing a population-based search procedure) over 30 executions for the test data. The third column of the table is the average number of neurons in the hidden layer.

We can see from Table IV that the algorithm is able to generate parsimonious neural networks with a smallnumber of neurons in the hidden layer and high generalization capability in the 16 classification tasks. The high quality of the solutions may be attributed to the use of alternative activation functions for each neuron combined with the capability of BAIS to identify and preserve the building blocks.

From the perspective of the search process, it is interesting to notice an important characteristic of BAIS, the population size grows and shrinks dynamically along the search, allowing a more efficient exploration/exploitation of the search space. This particular characteristic plays an important role on the algorithm's capability to perform multimodal search. During the experiments, we could observe that BAIS is able to generate not just one but several different high-quality neural network topologies in a single run.

In addition, we have also observed that BAIS found high-quality solutions in a small number of iterations in all the cases. This indicates that identifying and preserving building blocks, together with the capability to perform multimodal search, make the algorithm to converge quickly.

Next, we present the graphical representation of some obtained neural networks. These models are the best ones generated over 30 executions of the algorithm. Figs. 3–5 depict the neural network obtained for the Pima, Bupa, and Ionosphere datasets, respectively, with the weight vector omitted in Fig. 5.

TABLE V
AVERAGE CLASSIFICATION RATE OF THE ENSEMBLES OVER 30 EXECUTIONS ON THE TEST DATA

Dataset	Accuracy (%)	Generated Components	Selected Components	Gain (%)
WDBC	98.7 ± 1.1	8.6	3.4	0.5
Bupa	76.2 ± 2.1	15.2	9.7	2.4
Ionosphere	97.1 ± 2.3	19.7	10.7	3.3
Pima	81.8 ± 1.7	12.1	7.3	2.7
Sonar	86.4 ± 2.9	17.4	12.8	1.7
Card	87.6 ± 0.9	16.9	11.6	4.2
Vote	97.2 ± 1.6	11.3	7.7	0.9
Hepatitis	84.1 ± 2.2	13.7	6.9	1.6
Iris	98.8 ± 1.3	6.7	6.4	1.2
Wine	99.3 ± 1.2	7.5	5.9	0.6
Glass	73.6 ± 2.7	18.1	10.8	6.3
Satellite	89.7 ± 3.4	19.2	14.5	2.8
Letter	81.5 ± 3.7	21.8	13.6	2.3
Soybean	96.9 ± 1.1	8.9	7.7	4.8
Image	97.8 ± 2.0	7.2	6.8	1.9
Vehicle	81.3 ± 3.1	15.3	8.2	3.9

TABLE VI
AVERAGE ENSEMBLE DIVERSITY CONSIDERING 30 RUNS ON THE TEST DATA

Dataset	Diversity	Dataset	Diversity
WDBC	0.08	Iris	0.19
Bupa	0.28	Wine	0.16
Ionosphere	0.24	Glass	0.26
Pima	0.29	Satellite	0.29
Sonar	0.35	Letter	0.34
Card	0.31	Soybean	0.37
Vote	0.33	Image	0.22
Hepatitis	0.26	Vehicle	0.32

TABLE VII
AVERAGE CLASSIFICATION RATE OVER 30 EXECUTIONS. THE NUMBER UNDER PARENTHESIS INDICATES THE AVERAGE NUMBER OF COMPONENTS IN THE ENSEMBLE

Dataset	BAIS	EENCL	Coevolution	BOA	AIS
WDBC	98.7 (3.4)	97.2 (9.6)	96.6 (10) ⋆	98.1 (11.3)	97.7 (7.5) ⋆
BUPA	76.2 (9.7)	75.7 (17.8)	76.1 (10)	74.4 (12.5) ⋆	75.3 (8.2) ⋆
Ionosphere	97.1 (10.7)	94.5 (23.4) ⋆	93.8 (10) ⋆	94.9 (10.4) ⋆	95.6 (9.6) ⋆
Pima	81.8 (7.3)	77.6 (16.7) ⋆	78.3 (10) ⋆	78.4 (13.1) ⋆	79.2 (11.3) ⋆
Sonar	86.4 (12.8)	86.3 (19.2)	79.6 (10) ⋆	84.9 (11.8) ⋆	85.7 (14.8)
Card	87.6 (11.6)	86.1 (13.4) ⋆	86.2 (10) ⋆	84.5 (15.2) ⋆	87.2 (10.6)
Vote	96.3 (7.7)	94.7 (14.3) ⋆	93.4 (10) ⋆	93.7 (14.6) ⋆	95.9 (12.3)
Hepatitis	84.1 (6.9)	83.8 (11.6)	82.5 (10) ⋆	81.8 (9.7) ⋆	82.8 (10.4) ⋆
Iris	98.8 (6.4)	98.2 (18.5)	97.6 (10) ⋆	97.9 (13.8) ⋆	97.5 (14.6) ⋆
Wine	99.3 (5.9)	98.1 (16.2) ⋆	97.7 (10) ⋆	98.2 (14.6) ⋆	98.6 (12.4)
Glass	73.6 (10.8)	69.6 (19.6) ⋆	66.2 (10) ⋆	71.9 (12.1) ⋆	70.3 (17.1) ⋆
Satellite	89.7 (14.5)	87.1 (12.7) ⋆	88.2 (10) ⋆	88.9 (15.1)	87.4 (11.6) ⋆
Letter	82.5 (13.6)	76.8 (24.2) ⋆	79.5 (10) ⋆	78.1 (22.4) ⋆	80.0 (14.5) ⋆
Soybean	96.9 (7.7)	94.2 (13.4) ⋆	94.5 (10) ⋆	95.3 (9.8) ⋆	94.7 (11.2) ⋆
Image	97.8 (6.8)	96.7 (9.4) ⋆	97.3 (10)	96.9 (5.7) ⋆	97.6 (11.1)
Vehicle	81.3 (8.2)	77.9 (15.2) ⋆	76.4 (10) ⋆	79.2 (7.6) ⋆	80.8 (9.7) ⋆

We have also detected the most frequent type of activation function that appears in the generated networks. In the networks for the Bupa, Ionosphere, and Pima datasets, the *hyperbolic tangent* appears regularly, while for the WDBC the *cosine* is the most common. The *sine* function often appears for the Sonar dataset. Conversely, the *hyperbolic sine* is the function that does not appear so frequently as the others, considering all datasets.

D. Performance of the Ensembles

As demonstrated in the previous subsection, multiple high-performance neural networks were generated at each run of the algorithm. Now, BAIS can select some of them to compose an ensemble, aiming to achieve an improvement in performance. The method used to combine the individual outputs was majority voting [1]. The average performance of the obtained ensembles over 30 executions can be seen in Table V. The average number of ensemble components is presented in the fourth column. The last column shows the achieved percentage gain in performance over the best neural network taken in isolation, regarding the correct classification rate.

Notice that the comparison is made using information that is not available *a priori*, which makes our criterion the most severe one. In fact, the individual neural network with the best performance in the test dataset is privileged information, and therefore not available in usual applications. As a consequence, the fifth column of Table V is expressing the worst scenario for the ensemble. So, on average, the gain in performance by the ensemble will be better than what is indicated by the fifth column of Table V.

From Table V, we can see that the ensemble of neural networks improved the classification accuracy, outperforming the single best classifier in all cases (see Table IV). To evaluate the statistical significance of the difference among the performance of the ensemble and the performance of the single best classifier, the t-test was employed. Except for the WDBC, Iris, and Wine datasets, the gain in performance is statistically significant (at the level of 95%, $\alpha = 0.05$) in all the other datasets.

We believe that the small gain in performance obtained in the case of the WDBC, Iris, and Wine datasets comes from the fact that they are very simple problems and, as a result, all the candidate solutions generated by BAIS presented very good generalization capability even when applied in isolation. Therefore, the degree of diversity between the components is very low (see Table VI) and the use of an ensemble with this characteristic does not contribute to the improvement in the correct classification rate.

We have calculated the degree of diversity using (4) for the generated ensembles on the 16 datasets. Table VI presents the average of the degree of diversity over 30 runs. As expected, we can see that the proposed methodology has achieved a good level of diversity among the components of the ensembles.

E. Comparative Results

In this subsection, we compare the results obtained by BAIS with those generated by four bio-inspired algorithms: evolutionary ensembles with NCL (EENCL), a coevolutionary algorithm, the Bayesian optimization algorithm (BOA), and a standard AIS.

The EENCL algorithm [26] evolves MLP neural networks with logistic transfer functions using evolutionary programming. The number of hidden nodes is specified by the user. To maintain diversity in population, fitness sharing, and NCL have been used to encourage the formation of different species. These species are determined by clustering the individuals in the population using the k-means algorithm. The resulting clusters correspond to different species. The fittest individual from each species is selected to compose the ensemble. The number of clusters is determined by the user. The parameters used in EENCL are the same as those adopted by [26]: population size 25, number of generations 200, minimum and maximum number of clusters 3 and 25, respectively. The hidden layer contains 10 neurons.

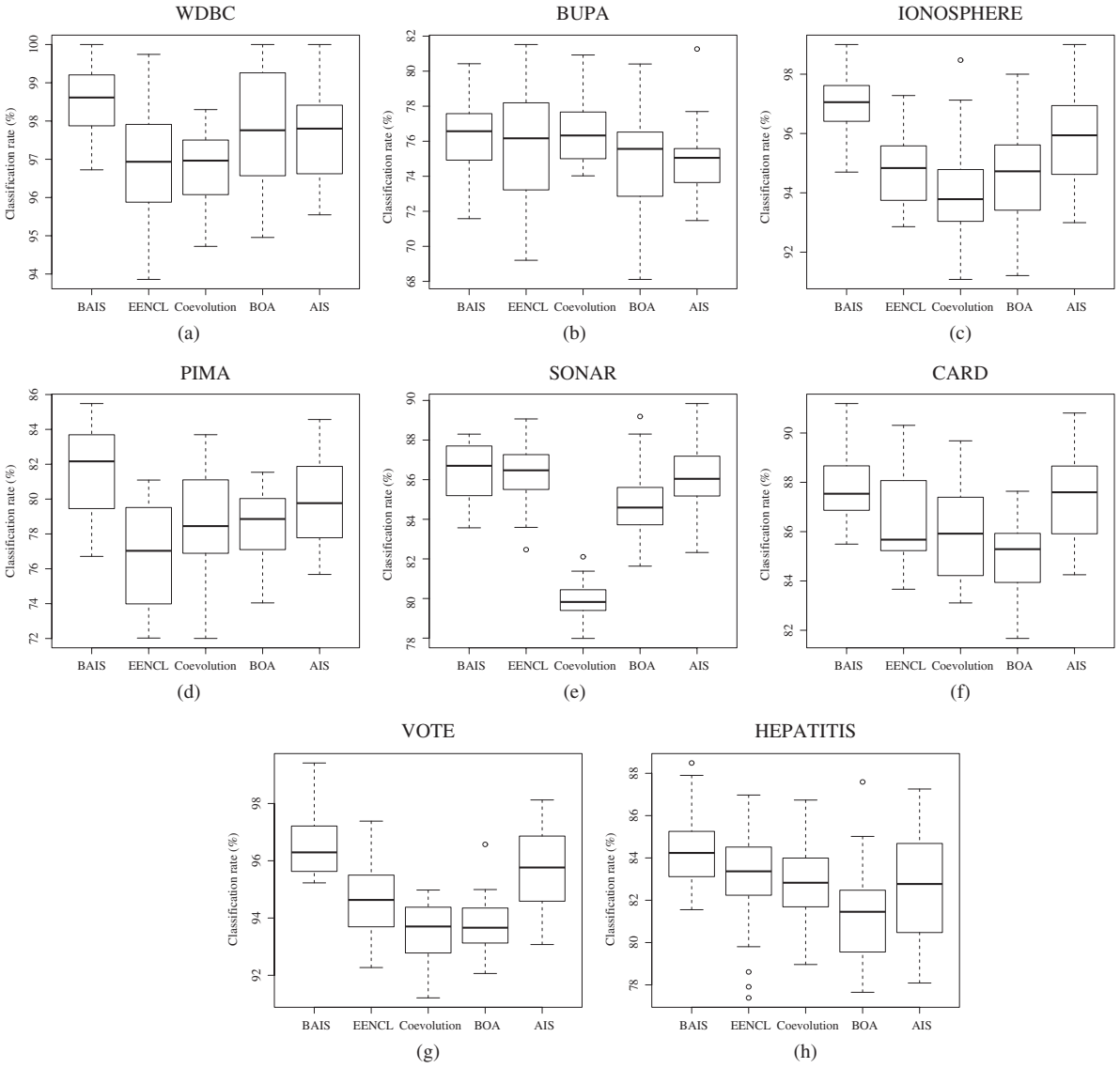

Fig. 6. Boxplots for the datasets (a) WDBC, (b) Bupa, (c) Ionosphere, (d) Pima, (e) Sonar, (f) Card, (g) Vote, and (h) Hepatitis.

The coevolutionary approach [49] evolves two separate populations cooperatively. One population consists of a number of generalized MLPs and evolves both weight vector and topology of the network using evolutionary programming. The other population evolves ensembles formed by subsets of elements of the first population using a steady-state genetic algorithm [50]. Each generation of the whole system consists of a generation of the population of neural networks followed by a generation of the population of ensembles. The parameter values are the same suggested by [49]. The population of networks has 30 individuals and the population of ensembles has 100 individuals, each one representing an ensemble composed of 10 networks. The maximum number of generations was set as 30.

BOA [51] is an estimation of distribution algorithm that also utilizes a Bayesian network to model the promising solutions. Like BAIS, the Bayesian network is built using the K2 algorithm and new individuals are sampled using the PLS algorithm. The population size was set as 200 and the maximum number of iterations was 100.

Finally, we have implemented a standard AIS aiming at establishing a figure of merit considering the performance of AIS with and without the Bayesian approach. The initial population was set as 50 and the maximum number of iteration was 100. Each selected antibody gives origin to two clones.

In order to perform a fairer comparison, we let BOA and the standard AIS evolve the type of activation function for each neuron, besides the number of nodes in the hidden layer. Like BAIS, the maximum number of hidden neurons was set as 10 for both algorithms. In addition, the standard AIS and BOA are also composed of two stages: the generation of the candidates and the selection of the components.

Table VII shows the comparative results. The number in parenthesis indicates the average number of components in the ensemble. To evaluate the statistical significance of the difference among the performance of the algorithms, the

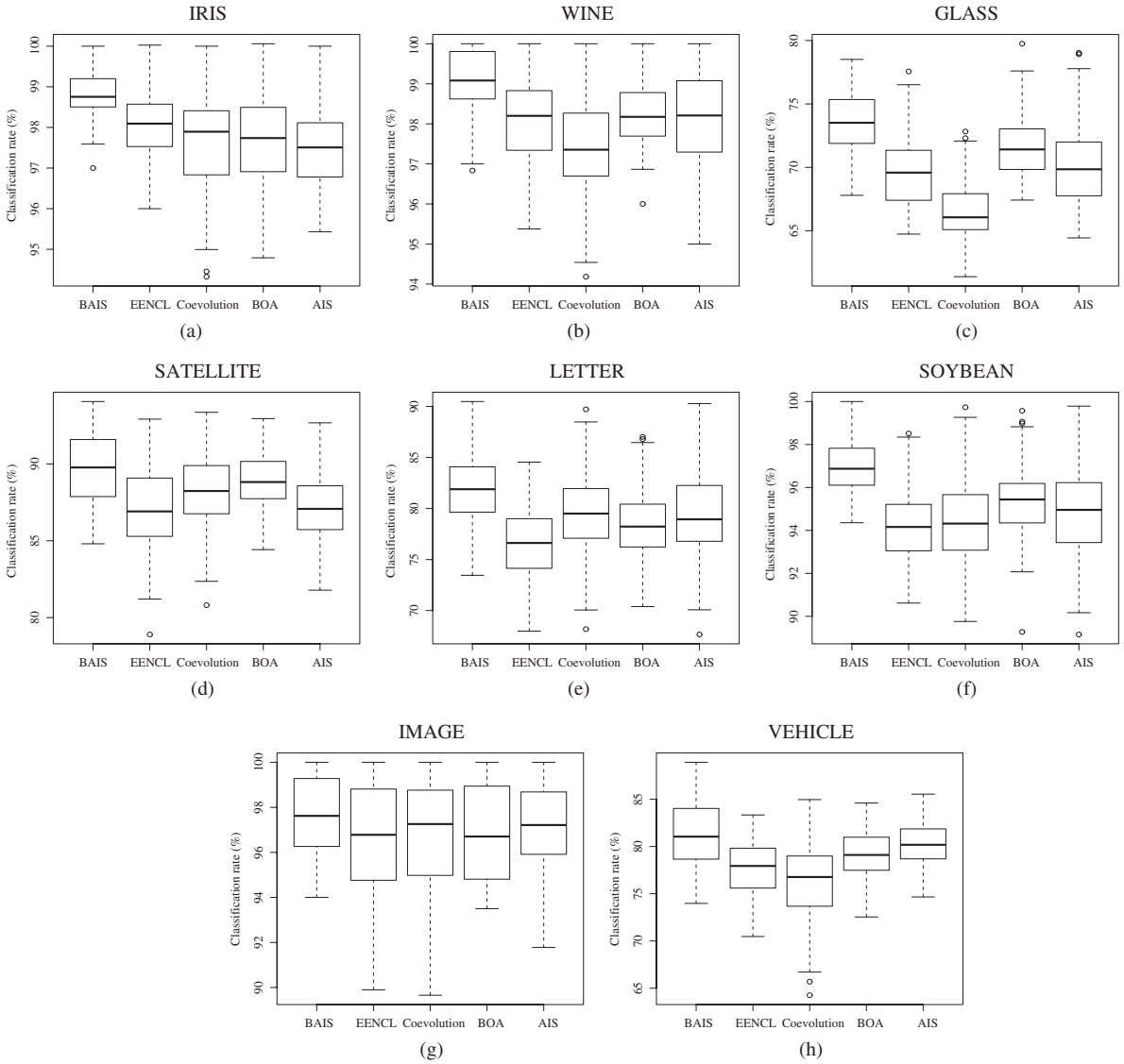

Fig. 7. Boxplots for the datasets (a) Iris, (b) Wine, (c) Glass, (d) Satellite, (e) Letter, (f) Soybean, (g) Image, and (h) Vehicle.

t-test was employed with a significance level of 95% ($\alpha = 0.05$). The symbol $\star$ denotes there is a statistical difference between the results obtained by BAIS and the compared algorithm.

As seen from Table VII, BAIS has been able to achieve a better generalization performance than the other algorithms for all datasets. In addition, the ensembles found by BAIS have fewer components than the ensembles generated by the competitors (especially WDBC, Pima, Vote, Hepatitis, Iris, and Wine data).

When compared with EENCL and the coevolutionary algorithm, BAIS algorithm may have achieved the best results as a consequence of the use of alternative activation functions for each neuron and because of its capability to identify and preserve building blocks.

When compared with BOA and the standard AIS (algorithms that also evolve the type of activation function), BAIS has produced a superior performance in all cases due to its efficient mechanism to explore/exploit the search space. The standard AIS, with its mutation operator, cannot capture the relationships among the variables. Besides, it is more likely to disrupt partial solutions found occasionally. On the other hand, BOA can deal with building blocks but there are some difficulties in applying it effectively. One difficulty is related to the loss of diversity in the population.

Another interesting fact observed during the comparative experiments is that BAIS has produced the best performance in terms of the number of iterations needed to find the best solutions. The competitors needed a higher number of iterations to achieve similar results.

We also present the boxplot comparing the five algorithms for all datasets. The graphical results can be seen in Figs. 6 and 7.

F. Discussion

As stated before, BAIS offers conceptual advantages over the competitors and they are described below.

The first advantage is related to the effective mechanism of BAIS to perform a multimodal search, finding diverse high-quality local optima quickly. In addition, the initial population size is not crucial to BAIS because of its capability to control the population size along the search process in response to the particularities of the problem.

Other aspect to be highlighted is the effective maintenance of building blocks. With this mechanism, BAIS avoids disrupting the partial solutions found so far, leading to a great improvement in the quality of the candidate solutions even in the first iterations. While BAIS found high-quality solutions in few generations, the other methodologies needed more generations to achieve a similar performance.

Moreover, with the absence of cloning and mutation operators, we do not have to worry about the proper choice of parameter values. The same does not occur with EENCL, the coevolutionary algorithm, and the standard AIS, since we need to specify parameter values such as mutation rate, population size in order to avoid premature convergence, and the number of clones.

Another motivation to use BAIS relies on the parsimony of the obtained solutions. In all datasets, the ensembles are composed of a few components with a small number of hidden neurons.

Regarding the implementation of BAIS, we notice that the algorithm does not require a large amount of computational resources [52]. Although a Bayesian network has to be produced at each step, the proposed methodology still preserves the computational tractability due to the restriction of at most two parents for each node in the network.

It is difficult to present here a comparison between the computational time required by the algorithms, since they were implemented using different languages and they were executed in different computers. BAIS was written using the C++ language and each algorithm was executed in a different computer, all with processor Intel Core 2 Duo. However, roughly comparing the computational cost of BAIS, EENCL, coevolutionary algorithm, BOA and AIS, we could observe that BAIS requires much less fitness evaluations than the other algorithms, and thus tends to produce a slightly better execution time.

Nonetheless, we are currently investigating some enhancements to be incorporated into the algorithm in order to alleviate the computational cost to design the Bayesian network: sporadic building of the network, parallelization, and incremental learning of the network.

VI. Conclusion

Under the machine learning perspective, the design of ensembles of neural networks can be viewed as a three-stage process: generation of a pool of candidate networks, selection of some networks from this pool, and the combination of the individual outputs to produce a single output. Since these tasks are very difficult to be achieved manually, several evolutionary algorithms have been proposed for helping the design of ensembles.

In this paper, we have proposed a BAIS to design ensembles of neural networks for classification problems. BAIS is applied to both generation and selection of components, and it differs from classical immune-inspired algorithms in the mechanisms to evolve the solutions. Instead of using traditional cloning and mutation operators, BAIS builds a probabilistic model of joint distribution of promising solutions and uses this model to generate new candidate solutions. The probabilistic model is founded on a Bayesian network, which is responsible for automatically capturing the expressive interactions of the variables of the problem and, consequently, identifying and preserving building blocks (partial solutions to the problem).

First, BAIS evolves a population of single-hidden layer MLP networks with alternative activation function for each neuron and an adjustable number of hidden neurons. The purpose is to produce high-quality classifiers with a degree of diversity in their input-output behavior. As soon as the candidate networks are generated, BAIS is applied to generate subsets of classifiers to be further combined.

Experiments on 16 datasets have indicated the effectiveness of the proposed methodology for designing accurate ensembles of neural network classifiers. When compared with other approaches reported in the literature, the results are consistently better, particularly when the average number of components of the ensemble is considered. In fact, though the Bayesian framework is responsible for an increment in the computational cost to synthesize the ensemble, the ensemble itself is more parsimonious, on average, when compared with the ensembles generated by all the other approaches. We may interpret this result as highly desirable when we think in real-world application of machine learning tools, a more parsimonious end solution, which tends to be synthesized multiple times (e.g., the ensemble being part of a commercial alarm system designed in hardware), is achieved at the cost of a more demanding computational methodology to produce the solution. Notice that the methodology to produce the solution should be applied just once. And the number of function evaluations (a usually adopted performance criterion for population-based search algorithms) is significantly reduced for BAIS, even when compared to other estimation of distribution algorithms, which may be interpreted as the main advantage of applying a Bayesian framework together with an immune-inspired approach. In fact, BAIS is an estimation of distribution algorithm which self-controls the population size along the search and which is capable of promoting and preserving diversity among the individuals of the population.

Although promising results have been obtained, we are currently investigating some aspects that can be further improved, such as alternative metrics for evaluating the Bayesian networks and other algorithms for sampling new individuals. We are also analyzing the performance of BAIS when applied to synthesize ensembles of neural networks for regression problems. Of course, the BAIS approach has not been designed to be applied solely to synthesize ensembles, but can be adapted to deal with other challenging optimization problems.

REFERENCES

[1] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.

[2] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in Neural Information Processing System*, vol. 7. Cambridge, MA: MIT Press, 1994, pp. 231–238.

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

[4] N. García-Pedrajas and C. Fyfe, "Construction of classifier ensembles by means of artificial immune systems," *J. Heuristics*, vol. 14, no. 3, pp. 285–310, Jun. 2008.

[5] N. García-Pedrajas and C. Fyfe, "Immune network based ensembles," *Neurocomputing*, vol. 70, nos. 7–9, pp. 1155–1166, Mar. 2007.

[6] R. Pasti and L. N. de Castro, "Bio-inspired and gradient-based algorithms to train MLPs: The influence of diversity," *Inf. Sci.*, vol. 179, no. 10, pp. 1441–1453, Apr. 2009.

[7] R. Pasti, L. N. de Castro, G. P. Coelho, and F. J. Von Zuben, "Neural network ensembles: Immune-inspired approaches to the diversity of components," *Natural Comput.*, vol. 9, no. 3, pp. 625–653, Sep. 2010.

[8] R. Pasti and L. N. de Castro, "The influence of diversity in an immune-based algorithm to train MLP networks," in *Proc. 6th Int. Conf. Artif. Immune Syst.*, 2007, pp. 71–82.

[9] B. Barbosa, L. Bui, H. Abbass, L. Aguirre, and A. Braga, "Evolving an ensemble of neural networks using artificial immune systems," in *Proc. 7th Int. Conf. Simul. Evol. Learn.*, 2008, pp. 121–130.

[10] X. Zhang, S. Wang, T. Shan, and L. Jiao, "Selective SVMs ensemble driven by immune clonal algorithm," in *Proc. EvoWorkshops*, 2005, pp. 325–333.

[11] F. O. de França, G. P. Coelho, P. A. D. Castro, and F. J. Von Zuben, "Conceptual and practical aspects of the aiNet family of algorithms," *Int. J. Natural Comput. Res.*, vol. 1, no. 1, pp. 1–35, 2010.

[12] L. N. de Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Trans. Evol. Comput.*, vol. 6, no. 3, pp. 239–251, Jun. 2002.

[13] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA: MIT Press, 1992.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[15] P. A. D. de Castro and F. J. Von Zuben, "BAIS: A Bayesian artificial immune system for the effective handling of building blocks," *Inf. Sci.*, vol. 179, no. 10, pp. 1426–1440, Apr. 2009.

[16] P. A. D. de Castro and F. J. Von Zuben, "Feature subset selection by means of a Bayesian artificial immune system," in *Proc. 8th Int. Conf. Hybrid Intell. Syst.*, Barcelona, Spain, Sep. 2008, pp. 561–566.

[17] P. A. D. Castro and F. J. Von Zuben, "MOBAIS: A Bayesian artificial immune system for multi-objective optimization," in *Proc. 7th Int. Conf. Artif. Immune Syst.*, Phuket, Thailand, Aug. 2008, pp. 48–59.

[18] P. A. D. de Castro and F. J. Von Zuben, "Multi-objective Bayesian artificial immune system: Empirical evaluation and comparative analyses," *J. Math. Model. Algorithms*, vol. 8, no. 2, pp. 151–173, 2009.

[19] P. A. D. de Castro and F. J. Von Zuben, "Multi-objective feature selection using a Bayesian artificial immune system," *Int. J. Intell. Comput. Cybern.*, vol. 3, no. 2, pp. 235–256, 2010.

[20] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: A survey and categorisation," *Inf. Fusion*, vol. 6, no. 1, pp. 5–20, Mar. 2005.

[21] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

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

[23] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.

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

[25] Y. Liu and X. Yao, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 29, no. 6, pp. 716–725, Dec. 1999.

[26] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Trans. Evol. Comput.*, vol. 4, no. 4, pp. 380–387, Nov. 2000.

[27] M. M. Islam, X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 820–834, Jul. 2003.

[28] H. Chen and X. Yao, "Regularized negative correlation learning for neural network ensembles," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1962–1979, Dec. 2009.

[29] H. Chen and X. Yao, "Multiobjective neural network ensembles based on regularized negative correlation learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 12, pp. 1738–1751, Feb. 2010.

[30] D. Dasgupta, "Advances in artificial immune systems," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 40–49, Nov. 2006.

[31] S. Sarafijanovic and J.-Y. Le Boudec, "An artificial immune system approach with secondary response for misbehavior detection in mobile ad hoc networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 5, pp. 1076–1087, Sep. 2005.

[32] L. N. de Castro and J. Timmis, "An artificial immune network for multimodal function optimization," in *Proc. World Congr. Comput. Intell.*, vol. 1. Honolulu, HI, 2002, pp. 699–704.

[33] G. P. Coelho and F. J. Von Zuben, "Omni-aiNet: An immune-inspired approach for omni optimization," in *Proc. 5th Int. Conf. Artif. Immune Syst.*, Oeiras, Portugal, Sep. 2006, pp. 294–308.

[34] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1999.

[35] P. A. D. Castro, G. P. Coelho, M. F. Caetano, and F. J. Von Zuben, "Designing ensembles of fuzzy classification systems: An immune-inspired approach," in *Proc. 4th Int. Conf. Artif. Immune Syst.*, Banff, AB, Canada, Aug. 2005, pp. 469–482.

[36] Y. Wu and C. Fyfe, "Exploratory data analysis with artificial immune systems," *Evol. Intell.*, vol. 1, no. 2, pp. 159–169, 2008.

[37] F. V. Jensen, *An Introduction to Bayesian Networks*. London, U.K.: UCL Press, 1996.

[38] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.

[39] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Mach. Learn.*, vol. 9, no. 4, pp. 309–347, Oct. 1992.

[40] M. Henrion, "Propagating uncertainty in Bayesian networks by probabilistic logic sampling," in *Proc. 2nd Annu. Conf. Uncert. Artif. Intell.*, vol. 2. 1988, pp. 149–163.

[41] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.

[42] E. M. Iyoda and F. J. Von Zuben, "Hybrid neural networks: An evolutionary approach with local search," *Integr. Comput.-Aided Eng.*, vol. 9, no. 1, pp. 57–72, Jan. 2002.

[43] Y. Liu and X. Yao, "Evolutionary design of artificial neural networks with different nodes," in *Proc. IEEE Int. Conf. Evol. Comput.*, Nagoya, Japan, May 1996, pp. 670–675.

[44] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Netw.*, vol. 6, no. 4, pp. 525–533, 1993.

[45] P. J. Angeline, G. M. Saunders, and J. P. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 54–65, Jan. 1994.

[46] L. I. 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.

[47] D. B. Skalak, "The sources of increased accuracy for two proposed boosting algorithms," in *Proc. AAAI Integr. Multiple Learn. Models Workshop*, 1996, pp. 1–6.

[48] A. Frank and A. Asuncion. (2010). *UCI Machine Learning Repository* [Online]. Available: http://archive.ics.uci.edu/ml

[49] N. García-Pedrajas, C. Hervás-Martínez, and D. Ortiz-Boyer, "Cooperative coevolution of artificial neural network ensembles for pattern classification," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 271–302, Jun. 2005.

[50] D. Whitley, "The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 116–121.

[51] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. Genetic Evol. Comput. Conf.*, vol. 1. Orlando, FL, 1999, pp. 525–532.

[52] B. M. Broom and D. Subramanian, "Computational methods for learning Bayesian networks from high-throughput biological data," in *Bayesian Inference for Gene Expression and Proteomics*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

Pablo A. Dalbem Castro received the M.Sc. degree in computer science from the Federal University of São Carlos, São Paulo, Brazil, in 2004, and the Ph.D. degree in computer engineering from the University of Campinas (UNICAMP), São Paulo, in 2009.

He is currently with the Laboratory of Bioinformatics and Bioinspired Computing, Department of Electrical and Computer Engineering, UNICAMP. His current research interests include bioinspired computing, Bayesian networks, data mining, feature selection, multiobjective optimization, and artificial neural networks.

Fernando José Von Zuben (M'91–SM'10) received the Dr.E.E. degree from the University of Campinas (UNICAMP), São Paulo, Brazil, in 1996.

He is currently the Head of the Laboratory of Bioinformatics and Bioinspired Computing, and also an Associate Professor at the Department of Computer Engineering and Industrial Automation, UNICAMP. He coordinates open-ended research projects in these topics, tackling real-world problems in the areas of decision making, pattern recognition, and discrete and continuous optimization. His current research interests include computational intelligence, bioinspired computing, multivariate data analysis, and machine learning.