Hybrid Training Method for MLP: Optimization of Architecture and Training

Cleber Zanchettin, Teresa B. Ludermir, and Leandro Maciel Almeida

***Abstract*—The performance of an artificial neural network (ANN) depends upon the selection of proper connection weights, network architecture, and cost function during network training. This paper presents a hybrid approach (GaTSa) to optimize the performance of the ANN in terms of architecture and weights. GaTSa is an extension of a previous method (TSa) proposed by the authors. GaTSa is based on the integration of the heuristic simulated annealing (SA), tabu search (TS), genetic algorithms (GA), and backpropagation, whereas TSa does not use GA. The main advantages of GaTSa are the following: a constructive process to add new nodes in the architecture based on GA, the ability to escape from local minima with uphill moves (SA feature), and faster convergence by the evaluation of a set of solutions (TS feature). The performance of GaTSa is investigated through an empirical evaluation of 11 public-domain data sets using different cost functions in the simultaneous optimization of the multilayer perceptron ANN architecture and weights. Experiments demonstrated that GaTSa can also be used for relevant feature selection. GaTSa presented statistically relevant results in comparison with other global and local optimization techniques.**

***Index Terms*—Genetic algorithms (GAs), multilayer perceptron (MLP), optimization, simulating annealing, tabu search (TS).**

I. Introduction

OPTIMIZATION is the process of finding the best solution for a problem from a group of possible solutions. An optimization problem has an objective function and a group of restrictions, both related to the decision variables. Genetic algorithms (GA) [14], simulated annealing (SA) [18], and tabu search (TS) [13] are iterative algorithms used to solve different combinatory optimization problems.

These three algorithms are the most popular from a class of optimization algorithms known as general iterative algorithms. All three optimization heuristics have similarities [29]: 1) they are approximation (heuristic) algorithms, i.e., they do not assure the finding of an optimal solution; 2) they are blind in that they do not know when they have reached an optimal solution, and therefore, they must be told when to stop; 3) they have a “hill climbing” property, i.e., they occasionally accept uphill (bad) moves; 4) they are general, i.e., they can easily be engineered to implement any combinatorial optimization problem; and 5) under certain conditions, they asymptotically converge to an optimal solution.

A manual selection of the artificial neural network (ANN) parameters involves difficulties such as the following: the exponential number of parameters that need to be adjusted, the need for *a priori* knowledge on the problem domain and ANN functioning in order to define these parameters, and the requirement of an expert when such knowledge is lacking. In most cases, the choice of parameters is performed manually through the trial and error method, which is tedious, less productive, and error prone. Furthermore, when the complexity of the problem domain increases and when optimized networks are desired, manual searching becomes quite difficult and unmanageable. This paper presents a hybrid method based on global and local optimization techniques (GaTSa), which automatically optimizes the ANN architecture and performance. This method is based on the integration of SA, TS, GA, and backpropagation (BP). In this paper, the performance of GaTSa is investigated in the simultaneous optimization of multilayer perceptron (MLP) architecture and weights.

GaTSa presents some interesting characteristics: 1) search optimized for generating new solutions; 2) pruning to eliminate connections and to optimize network size; and 3) a constructive approach for finding the best network architecture. With the help of experiments, we investigate different cost functions for searching the best ANN architecture in ANN optimization. The experiments demonstrate that GaTSa can also be used for a relevant feature subset selection. During network architecture optimization, the network input processing units may be eliminated in accordance with the GaTSa performance. Thus, an exclusion of the irrelevant network inputs is obtained in the ANN optimization process.

This paper makes the following major contributions: 1) the extension of TSa [21] into GaTSa; 2) the empirical evaluation of GaTSa on 11 public-domain data sets and five cost functions; 3) the possibility of using GaTSa to feature selection; and 4) the use of the factorial experimental design to estimate the configuration parameters of GaTSa.

II. Related Work

A number of approaches in the literature have used the integration of TS, SA, and GA for specific applications. This section describes only those works that are more or less similar to our work. An integration of the three heuristics was initially proposed by Fox [11]. Tsai *et al.* [34] used a hybrid algorithm to feedforward the ANN architecture and parameter design. Palmes *et al.* [25] used a mutation-based algorithm to train

Manuscript received June 7, 2010; revised November 24, 2010; accepted January 2, 2011. Date of publication February 10, 2011; date of current version July 20, 2011. This work was supported by CNPq/CAPES/FACEPE (Brazilian research agencies). This paper was recommended by Associate Editor C.-T. Lin.

The authors are with the Center of Informatics, Federal University of Pernambuco, 50740-540 Recife, Brazil (e-mail: cz@cin.ufpe.br; tbl@cin.ufpe.br; lma3@cin.ufpe.br).

Digital Object Identifier 10.1109/TSMCB.2011.2107035

TABLE I
COMPARISON BETWEEN THE INVESTIGATED ALGORITHMS

	Fox (1992)	Stepniewski et al. (1997)	Ludermir et al. (2006)	GaTSa
Heuristic	GA, SA, TS and BP	GA and SA	SA, TS and BP	GA, SA, TS and BP
Initialization	One solution	One solution	One solution	Population of solutions
Neighborhood	Generate new solutions from one solution	Generate new solutions from one solution	Generate new solutions from one solution	Evolve the population by genetic micro-evolutions
Evolving	Genetic operators	Genetic operators	Inversion operators	Genetic operators
Search order	-	Designer define the maximum architecture	Designer define the maximum architecture	Start from minimum and find the best architecture
Visitation	Keep a list of visited solutions	-	Randomlly visit solutions	Keep a list of visited solutions
Acceptance	Based in GA cost function	Based in Metropolis criterium	Based in Metropolis criterium	Based in Metropolis criterium
Stop criteria	Iteration number	Iteration number	Iteration number and GL_5	Iteration number and GL_5
Local tunning	-	-	Using Backpropagation	Using Backpropagation

MLP. Gepperth and Roth [12] used an evolutionary multiobjective process to optimize the feedforward architectures. Works using SA and TS for ANN optimization are scarce. The recognition of signal responses using SA and BP was implemented for the training of an MLP with a fixed architecture containing two hidden layers [26].

The SA method was successfully used in some global optimization problems, as can be seen in Corana *et al.* [7] and Sexton *et al.* [31]. In Stepniewski and Keane [32], SA and GA were used to optimize the architectures of MLP. Similar experiments were performed by Sexton *et al.* [31]. Sexton *et al.* [30] repeated the same experiments applying the TS algorithm. Metin *et al.* [24] used SA to optimize the ANN architectures applied to expert diagnosis systems. In Hamm [15], SA was used to optimize the ANN weights. Mart and El-Fallahi [23] presented a new proposal based on TS for MLP weight optimization.

The integration of TS, SA, and BP (TSa) was proposed by the authors in [21] for MLP architecture and weight optimization. The method combines accepting the new solution scheme of SA with the multiple search of TS. In this approach, a set of new solutions with a fixed size (the maximum network architecture needs to be defined) is generated in each iteration, and the best one (i.e., the one with the lowest cost) is selected according to the cost function, as performed by TS. In the present study, some experiments compare the performance of TSa with GaTSa.

Table I provides a comparison among GaTSa and three other approaches with their differences and similarities. In all approaches, SA, TS, and GA are typically used to adjust the weights between processing units in fixed architectures. GaTSa performs the simultaneous optimization of the MLP architecture and weights, using a constructive way to find the best network architecture and pruning to eliminate connections and to optimize the network size.

III. Feature Selection Methods

Feature selection is based on the reaction of the cross-validation data set classification error due to the removal of features (inputs). Feature selection with ANN can be thought of as a special case of architecture pruning, where the input features, rather than the hidden neurons or weights, are pruned [28].

In the literature, feature selection methods are categorized as wrapper, embedded, and filter methods [4]. In the wrapper method, subsets are evaluated and partitioned based on the

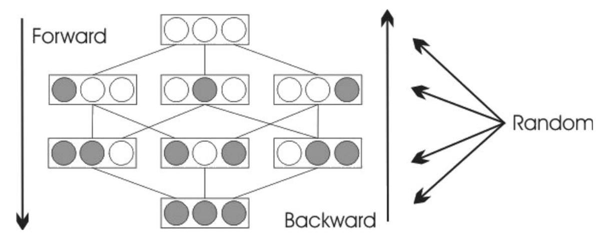

Fig. 1. Forward, backward, and random search strategies.

output of the base classifiers before making the final decision. In the filter method, subsets are partitioned prior to training. This method is less expensive, and the selection of architecture and base classifiers does not have to be made in advance. The embedded method is a native part of the classifier itself and is implemented by the learning-method evaluation criteria for selecting the most relevant attributes.

We use the classical feature selection methods, namely, hill-climbing [35], random bit climber [8], best-first [1], beam search [1], and Las Vegas [20], in a wrapper way to evaluate the performance of GaTSa in optimal feature subset selection. These methods were chosen because they are simple and similar to GaTSa.

The input selection process is based on some saliency measure aiming to remove less relevant inputs. Fig. 1 explains the three search strategies. For GaTSa, the input connections with minor statistical relevance will be the first to be removed. If GaTSa eliminates all input connections of one input processing node, this input is eliminated from the network architecture.

IV. Integration of SA, TS, and GA in a Constructive Way

The SA method has the ability to escape from local minima through the choice between accepting or discarding a new solution that increases cost (uphill moves). The TS method, in contrast, evaluates a group of new solutions at each iteration (instead of only one solution as in SA). This makes TS faster as it generally needs less iterations to converge. The GA evolution, in turn, involves a sequence of iterations, where a group of solutions evolves through selection processes and reproduction. These observations motivated the proposal of an optimization method (GaTSa) that combined the main advantages of GA, SA, and TS in order to avoid their limitations.

Fig. 2 shows a summary of the major features borrowed (between parentheses) from different heuristics. GaTSa works in the following manner: the initial solution has a minimum valid

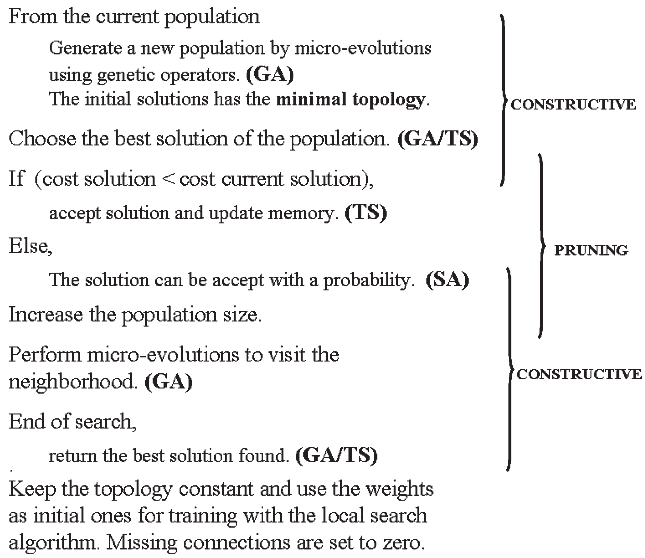

Fig. 2. GaTSa composition.

architecture size. A group of new solutions is generated at each iteration, starting from the microevolution of the current population, as in GA. The cost of each solution is evaluated, and the best solution is chosen, as in TS. However, unlike TS, this solution is not always accepted. The acceptance criterion is the same as that used in the SA algorithm—if the chosen solution has a smaller cost than the current solution, it is accepted; otherwise, it can either be accepted or rejected depending on a probability calculation. This probability is given by the same expression as that used in the SA method. Previously visited solutions are marked as tabu, as in TS. During the search, the chromosome size is increased in a constructive manner in order to find the best solutions according to the acceptance criterion. By the end of the optimization process, only the best solution is returned.

The proposed technique has two well-defined stages: a global search phase, where it makes use of the capacity for generating new solutions from the GA, cooling process, and cost function of the SA as well as the memory characteristics of the TS technique, and a local search phase, where it makes use of characteristics such as gradient for a more precise solution adjustment.

During the global search stage, the proposed algorithm can also optimize the number of hidden processing units in the hidden layer. Most optimization methods based on pruning require the designer to set up the ANN maximum architecture. Hence, the algorithm can eliminate connections from this structure to find a network architecture with an acceptable result. The GaTSa algorithm is based on a constructive methodology. Thus, the initial solution represents the minimum valid architecture—the number of nodes in the input layer gives the number of problem attributes, the hidden layer has only one node, and the output layer has nodes corresponding to the number of classes in the problem. For the optimization process, new nodes are added as and when required. New nodes are created with all connections between adjacent layers and are eventually eliminated according to their influence on the MLP performance.

The pseudocode of GaTSa is presented in Algorithm 1. Let S be a group of solutions and f be a real cost function. The proposed algorithm searches the global minimum s, such that $f(s) \leq f(s'), \forall s' \in S$. The process finishes after $I_{\max}$ iterations or if the stop criterion based on the validation error is satisfied. The best found solution S_{BSF} (*best so far*) is returned. The cooling process updates temperature T_i of iteration i to each I_T algorithm iteration. At each iteration, a new population with k solutions of size z is generated. A genetic microevolution of g_n generations is used to generate this population from the current population. The microevolution combines the best population solutions and, in the process, creates and eliminates network connections, like a pruning process. The initial solution is coded with the minimum valid network architecture, and new hidden nodes are added following the constructive process.

Algorithm 1—Pseudocode of the proposed algorithm

```
P_0 ← initial population with K solutions s_k
    and size s_z
List ← ∅ (tabu list)
T_0 ← initial temperature
Update S_BSF with s_k of the P_0 (best solution
 found so far)
Update List ← S_BSF
For i = 0 to I_max − 1
 If i + 1 is not a multiple of I_T
   T_{i+1} ← T_i
 End − If
 Else
   T_{i+1} ← new temperature
   If validation based stopping criteria
    are not satisfied
     Stop global search execution
   End − If
   Increase the size of the population P_i
   P_i ← P_z
 End − Else
 For j = 0 to g_n
   Generate a new population P' from P_i
   P_i ← P'
 End − For
 Choose the best solution s_k from P_i
   and is not in tabu list
 If f(s') < f(s_k)
   s_{k+1} ← s'
   List ← List − (oldest solution) + s_k
 End − If
 Else
   s_{k+1} ← s'with probability e^{(f(s')−f(s_k)/T_{i+1})}
   List ← List − (oldest solution) + s_k
   If f(s_{k+1}) < f(S_BSF)
    Update S_BSF
   End − If
 End − Else
End − For
Keep the architecture contained in S_BSF
constant and use the weights as initial ones
for training with the BP algorithm
```

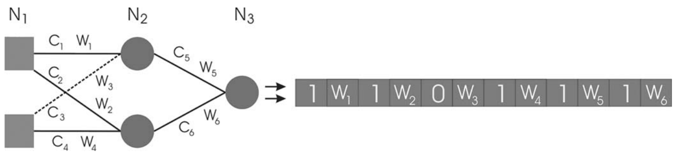

Fig. 3. MLP codification sample.

A. Representation of Solutions

The MLP architecture definition depends on the choice of the number of layers and the number of hidden nodes in each of these layers. All MLP architectures have a single hidden layer network, containing connections only between adjacent layers. The network architecture contains $N1$ input nodes, $N2$ hidden nodes, $N3$ output nodes, and $BN2$ and $BN3$ as the bias for the hidden and output layers, respectively.

Normally, parameters $N1$ and $N3$ are problem dependent according to the data preprocessing and the number of input and outputs features, but $N2$, $BN2$, and $BN3$ must be defined in the ANN implementation. Thus, the maximum number of connections is given by

$$N_{\max} \equiv (N1 \times N2 + BN2) + (N2 \times N3 + BN3). \quad (1)$$

Each solution s is coded in a vector that represents the C connections and the W weights among the ANN nodes. C means the connectivity, containing a set of bits that represent the network architecture, and W contains real numbers that represent the network weights

$$s \equiv (C, W) \quad (2)$$

$$C \equiv (c_1, c_2, \ldots, c_{N_{\max}}) \quad c_i \in \{0, 1\}, \quad i = 1, 2, \ldots, N_{\max} \quad (3)$$

$$W \equiv (w_1, w_2, \ldots, w_{N_{\max}}) \quad w_i \in \Re, \quad i = 1, 2, \ldots, N_{\max} \quad (4)$$

where $\Re$ is the set of real numbers.

Fig. 3 shows a sample of the vector that represents the MLP architecture. Thus, connection i is specified by two parameters: 1) a connectivity bit (c_i), which is equal to one if the connection exists and zero otherwise, and 2) the connection weight (w_i), which is a real number. If the connectivity bit is equal to zero, its associated weight is not considered since the connection does not exist in the network.

The initial solution s_0 is an MLP network with the minimum architecture ($N1$ and $N3$ are problem dependent) fully connected (i.e., $c_i = 1, \ i = 1, 2, \ldots, N_{\max}$), and the initial weights are randomly generated from a uniform distribution in the interval $[-1.0, +1.0]$.

B. Cost Function

Unlike the constructive algorithms that generate a solution only at the end of the process, the iterative algorithms derive possible solutions for each iteration. The cost function is used to evaluate the performance of successive iterations and to select a solution that minimizes an objective function.

Different cost functions can be used to evaluate the quality of a solution. Other authors had investigated the hole of the cost functions [17]. In this paper, five ways of performing cost evaluation were investigated.

1) Average Method: The cost $f(s)$ of solution s is given by the mean of the classification error for the training set and the percentage of the connections used by the network. For classification problems, the cost $f(s)$ of solution s is

$$f(s) = \frac{1}{2}\left(E(P_t) + \psi(C)\right) \quad (5)$$

$$\psi(C) = \frac{100}{N_{\max}} \sum_{i=1}^{N_{\max}} c_i. \quad (6)$$

Considering N_C classes in the data set, the true class of pattern x from the training set P_t is defined as

$$\gamma(x) \in \{1, 2, \ldots, N_C\} \qquad \forall x \in P_t. \quad (7)$$

For prediction problems, the cost $f(s)$ of solution s is given by the mean of the squared error percentage (SEP) for the training set and the percentage of the connections used by the network:

$$f(s) = \frac{1}{2}\left(SEP(P_t) + \psi(C)\right) \quad (8)$$

$$SEP = 100 \frac{o_{\max} - o_{\min}}{N_c \# P_t} \sum_{p=1}^{\# P_t} \sum_{i=1}^{N_c} \left(\phi(x)_{pi} - \gamma(x)_{pi}\right)^2 \quad (9)$$

where $\phi(x)_{pi}$ is the predicted class of pattern x and $o_{\min}$ and $o_{\max}$ are the minimum and maximum values of the output coefficients in the problem representation (assuming that these are the same for all output nodes).

2) Weighted Average Method: In the experiments, the network parameters, network performance, network connection percentage, and percentage of the hidden nodes are weighted by the parameters α, β, and κ, respectively.

For the classification problems

$$f(s) = \frac{(E(P_t) * \alpha) + (\psi(C) * \beta) + (pN * \kappa)}{(\alpha + \beta + \kappa)}. \quad (10)$$

For the prediction problems

$$f(s) = \frac{(SEP(P_t) * \alpha) + (\psi(C) * \beta) + (pN * \kappa)}{(\alpha + \beta + \kappa)} \quad (11)$$

where pN is the percentage of the used hidden node connections, $\alpha = 1$, $\beta = 0.5$, and $\kappa = 0.25$. These values were determined empirically by previous experiments.

3) Weight Decay: The method was initially proposed as an implementation to improve the BP algorithm for the preference bias of a robust ANN that is insensitive to noise [16], [36]. In a network architecture, the weight decay mechanism performs differentially toward zeroes by reinforcing large weight connections and weakening small weight connections. As small weights can be used by the network to code noise patterns, this weight decay mechanism is considered especially important in noisy data.

The weight decay mechanism is used in the GaTSa cost function to eliminate solutions with small weights that can be used by the ANN to code noise patterns. The GaTSa cost function is presented in the following:

$$f(s)=\frac{1}{2}\sum_{i} E(P_t)+\frac{1}{2}\psi(C)+\frac{1}{2}\mu\sum_{ij} W_{ij}^2/\left(1+W_{ij}^2\right) \quad (12)$$

where W_{ij} is the weight connection from node j to node i.

4) Multiobjective Optimization: This is the search for simultaneously minimizing the u components f_k, $k = 1, \ldots, u$, of a vector function f of a variable s in a universe u, where

$$f(s) = (f_1(s), \ldots, f_n(s)) \,. \quad (13)$$

Most problems usually have no unique global solutions, but they have a set of equally efficient or noninferior alternative solutions, known as the Pareto-optimal set [10]. The Pareto-optimal solutions consist of all solutions for which the corresponding objective cannot be improved in any one dimension without degradation in another.

In the present work, the multiobjective strategy is used in genetic operators to evolve the population, considering two goals to be minimized—the MLP size and its generalization. In contrast to the objective problem, the ranking of a population in the multiobjective case is not unique.

5) Multiobjective Optimization and Weight Decay: The fifth cost function investigated comprises a combination of multiobjective optimization and weight decay strategies.

C. Insertion of New Hidden Nodes

The constructive process is used to add new hidden nodes in the network architecture. The search process starts with a high probability of adding new nodes in the network architecture, but in order to perform a better error surface exploration, the addition of new nodes is controlled by a probability. This probability decreases over time by the multiplication of the actual probability by a factor (ϵ), which is smaller than but close to one. The initial probability λ_ϵ and the factor ϵ must be defined in the implementation as well as in I_λ (the number of iterations between two consecutive probability variations) and $I_{\max}$ (the maximum number of iterations). Thus, the probability of the insertion of new hidden nodes λ_i at iteration i is given by

$$\lambda_i \equiv \begin{cases} \epsilon\lambda_{i-1}, & \text{if } i = kI_\lambda, \qquad k = 1, 2, \ldots, \frac{I_{\max}}{I_\lambda} \\ \lambda_{i-1}, & \text{otherwise.} \end{cases} \quad (14)$$

D. New Solution Generation Mechanism

The initial solution is randomly generated, with $N1$ and $N3$ being problem-dependent values and $N2 \in 1, 2, \ldots, N3$. The initial population is defined with a size of ten chromosomes. From the current solution $s = (C, W)$, the new solution $s' = (C', W')$ is generated by the genetic microevolution of g_n generations. The chromosomes are classified by rank-based fitness scaling [2]. Parent selection for the next generation is accomplished in a probabilistic manner using universal stochastic sampling [2]. Elitism was not used, and the crossover operator uniform crossover [33] was used for the combination of parent chromosomes, with a probability of 80%. The crossover operation was performed by combining the parts of the parent chromosomes that have the same length as in the succeeding sample. The mutation operator used was the Gaussian mutation [31], with a probability of 10%.

	Uniform Crossover														
Parent A	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Parent B	0	0	0	0	0	0	0	0	0	0	0				
Mask	0	1	0	1	1	1	0	0	1	1	0				
Child A	1	1	1	0	0	1	0	0	1	0	0	1	1	1	1
Child B	0	1	0	1	1	1	0	0	1	1	0.				

E. Stop Criteria

The optimization process stops for the following reasons: 1) the GL_5 criterion defined in Proben1 [27] is met (based on the classification error or SEP of the validation set), or 2) the maximum number of iterations is reached. For the implementation of the GL_5 criterion, the classification error or SEP for the validation set is evaluated at each I_T iteration.

The GL_5 criterion is a good approach for avoiding overfitting to the training set. The classification error for the validation set P_v is given by $E(P_v)$. Thus, if $V(k)$ denotes the classification error $E(P_v)$ at iteration $i = kI_T$, $k = 1, 2, \ldots, I_{\max}/I_T$, the *generalization loss* parameter (GL) is defined as the relative increase in the validation error over the minimum-so-far. The GL_5 criterion stops the execution when the parameter GL becomes higher than 10%

$$GL(k) \equiv \left(\frac{V(k)}{\min_{j\le k} V(j)} - 1\right). \quad (15)$$

F. Feature Subset Selection

In training and improving the network weights and connections, the GaTSa method is able to eliminate the input connections of the ANN architecture. The input processing node represents a feature of the data set. The input connections with the highest usage frequency have the highest importance in the classification and prediction tasks, and the inputs with minor statistical relevance will possibly be the first to be removed. If GaTSa eliminates all input connections of one input processing node, that particular input is eliminated from the network architecture.

Thus, the proposed method can be used to feature subset selection, reducing the problem dimensionality and, consequently, the complexity of the generated ANN. The feature subset selection process performed by GaTSa is a combination of the wrapper and embedded definitions.

G. Local Search Algorithm

Global optimization techniques are relatively inefficient for minimum local search. In this case, it is important to improve the performance of the ANNs training the best architectures found in the global search phase with a local search method. This strategy represents the second phase of the proposed optimization method: without changing the architecture generated by the global search, the final network produced is used as a

TABLE II
CHARACTERISTICS OF THE USED DATA SETS

Database	Examples	Features			Class	
		b	c	t	b	c
Artificial Nose	5,400	0	6	6	3	0
Iris	150	0	4	4	3	0
Diabetes	768	0	8	8	2	0
Thyroid	7,200	9	6	21	3	0
Card	690	0	51	51	2	0
Cancer	699	0	9	9	2	0
Glass	214	0	9	6	6	0
Heart-Cleveland	303	0	35	35	2	0
Horse	364	0	58	58	3	0
Soybean	683	0	82	82	19	0
MGlass	1,000	0	4	4	0	1

TABLE III
METHOD CONFIGURATION PARAMETERS

Parameters	MLP	GA	TS	SA	TSa	GaTSa
Iterations	1.000	500	100	1.000	100	100
Population size	-	10	20	1	10	10
Initial Temperature	-	-	-	1	1	1
Temp. reduction factor	-	-	-	0.9	0.9	0.9
Micro-Evolutions	-	-	-	-	-	10
Crossover rate	-	20%	-	-	-	20%
Mutation rate	-	10%	-	-	-	10%
Elitism	-	10%	-	-	-	-
Tabu size list	-	-	20	-	-	20
Bit inversion	-	-	20%	20%	20%	-
GL stop criteria	5	5	5	5	5	5
BP learning rate	0.001	-	-	-	0.001	0.001
BP momentum rate	0.7	-	-	-	0.7	0.7

starting point in the local search. In this way, the connections and weights obtained by global optimization are preserved. The missing connections between nodes have their weights initialized to zero. This architecture is used as the start search point for local optimization. This combination of global and local optimization techniques is often referred to as hybrid training. In this paper, the BP algorithm using the sum squared error (SSE) was chosen for local search optimization.

V. EXPERIMENT METHODOLOGY

Ten classification (1–10) and one prediction (11) data sets were used in the experiments: 1) the odor recognition problem; 2) diabetes diagnoses; 3) fisher's iris; 4) thyroid dysfunction; 5) credit screening data set; 6) breast cancer; 7) glass identification; 8) heart disease; 9) horse colic; 10) soybean; and 11) Mackey–Glass time series [22]. The classification data sets were obtained from the UCI repository [3] and Mackey–Glass time series data set from [22].

In Table II, a summary of the used databases along with the number of examples is presented. It includes the number of binary (b), continuous (c), and total (t) features and the number of binary (b) classes. In the selection of data sets to be used, we seek databases with different characteristics, mixing problems of classification and prediction. We are mainly concerned with diversity in the experiments in order to support the obtained results.

A. Training Methodology

The local training algorithm employed is the BP method using SSE. The configuration parameters of all methods are summarized in Table III. Training stops for the following reasons: 1) the GL_5 criterion [27] is satisfied twice (to avoid initial oscillations in validation errors); 2) the training progress criterion is met, with $P_5(t) < 0.1$ [27]; and 3) the maximum number of iterations is reached.

We used 30 twofold iterations [5]. At each iteration, data were randomly divided into halves. One half was the input for the algorithm (65% for training and 35% for the validation set), and the other half was used to test the final solution (test set). All network units implemented the hyperbolic tangent activation function. The patterns were normalized to the range $[-1, +1]$.

In the experiments, each solution represents an ANN architecture. For the GA and GaTSa experiments, the chromosomes (solution) are classified by rank-based fitness scaling [2]. Parent selection for the next generation is accomplished in a probabilistic manner using universal stochastic sampling [2]. Elitism—the best chromosomes are preserved for the next generation—was used in GA. For the combination of parent chromosomes, the crossover operator uniform crossover [33] was used. The mutation operator was Gaussian mutation [31].

For the TS, TSa, and GaTSa experiments, the proximity criterion [29] was used to compare two solutions. A new solution is considered identical to the tabu solution for the following reasons: 1) each connectivity bit in the new solution is identical to the corresponding connectivity bit in the tabu solution, and 2) each connection weight in the new solution is within $\pm a$ of the corresponding connection weight in the tabu solution. The parameter a is a real number with a value of 0.001.

B. Subset Selection

1) Representation of Solutions:

a) Classical methods: In classical feature selection methods, each subset is represented by a vector that defines the selected and nonselected attributes. In the experiments, we have used different search strategies considering the characteristics of the investigated search technique. With the hill-climbing, best-first, and beam search, we use the forward, backward, and random search strategies. The three search strategies are shown in Fig. 1, where the white circles represent the not selected features and the shaded circles represent the selected features. The proposed GaTSa method implements its own representation topology.

The forward strategy starts with the empty set and gradually adds the features. The backward strategy starts with the full set and deletes the features. The random approach starts from a random set and randomly performs the addition and removal of features. Unlike other strategies, the random bit climber method has removed and added the attributes during the search process. Thus, in order to carry out the search in different directions, we use different initial states—initial solution without features, with all features, and randomly selected features. The Las Vegas and proposed GaTSa methods implement their own search strategies.

2) Performance Classification:

a) Classical methods: To determine the classification accuracy for the classical methods (hill-climbing, best-first, beam

TABLE IV
RESULTS FOR THE MLP NEURAL NETWORKS

	Class.(%)/SEP	Std.	Input	Hidden	Connec.
Artificial Nose	6.30	1.40	6	10	90
Iris	6.83	2.34	4	5	32
Thyroid	7.38	1.77	21	10	240
Diabetes	27.08	0.91	8	10	100
Card	27.12	6.90	51	8	424
Cancer	8.61	4.59	9	10	110
Glass	64.06	1.7	9	9	135
Heart-Cleveland	24.04	3.77	35	9	333
Horse	38.63	0.81	58	10	610
Soybean	85.50	3.50	82	10	1.010
Mackey-Glass	1.43	2.45	4	4	50

search, random bit climber, and Las Vegas), a KNearest neighbor (k-NN) classification algorithm was used [9]. This strategy was used to reduce the experiment time. The similarity function used is straightforward, and it relies on equally weighted features. To compute the similarity distance between two scaled instances, we use the Euclidian distance. In the k-NN algorithm, the value of k is seven, defined in an empirical way.

In the experiments for feature selection, the database was divided into two groups—50% of the patterns was used in the feature selection process, and the remainder was used for testing the obtained subsets. The quality of each subset was given by the performance of the classifier. The cross-validation process was used to obtain the classifier performance.

The fivefold cross-validation in subset search and the tenfold cross-validation in subset evaluation were used. This difference in performance assessment was necessary in order to reduce the computational cost of the search process for the optimal subset of features.

b) GaTSa: Table VI presents the maximal topology of the experiments. In all ANN architectures, the $N1$ and $N3$ values are problem dependent, and N2 was obtained from the experiments in Section VI-A (for a fair comparison, a fixed GaTSa architecture is used).

VI. RESULTS AND DISCUSSION

A. MLP Experiments

Apart from the GaTSa method, the other investigated optimization techniques require a good initial network architecture (maximum architecture) for a successful ANN architecture optimization. To define this architecture, experiments were performed with random architecture sizes on each one of the data sets. For all data set experiments in each architecture, we used 30 twofold iterations. Table IV presents the SEP and the classification error of the test set obtained in the training of a fully connected MLP by using a gradient descent with momentum BP.

B. Optimization Methodology Experiments

1) GaTSa—Fixed Architecture Experiments: Table V presents the average performance of each investigated optimization technique, starting the search with the same network architecture as in the artificial nose data set. These results were obtained for each technique as the optimization of the number of connections and weight connection values of an MLP. The parameters evaluated were the following: 1) the SEP and classification error (class) of the training, validation, and test sets; 2) the algorithm iteration number; 3) the ANN connection number; and 4) the temperature value.

TABLE V
GaTSa FIXED ARCHITECTURE OPTIMIZATION IN ARTIFICIAL NOSE

	Training		Validation		Test			
Technique	SEP	Class	SEP	Class	SEP	Class	Iterat.	Connec.
TS	18.74	5.44	18.86	5.88	18.75	5.3805	51	11.42
SA	19.65	6.91	19.76	7.47	19.65	6.9331	715	11.77
GA	21.66	15.88	21.73	16.52	21.66	15.9240	315	16.64
GaTSa	18.69	3.58	18.76	3.81	18.69	3.5664	46	8.33
GaTSa + BP	4.78	-	2.41	-	2.14	2.8684*	86	8.33*
BP	6.30	-	3.15	-	2.84	6.7854	90	36

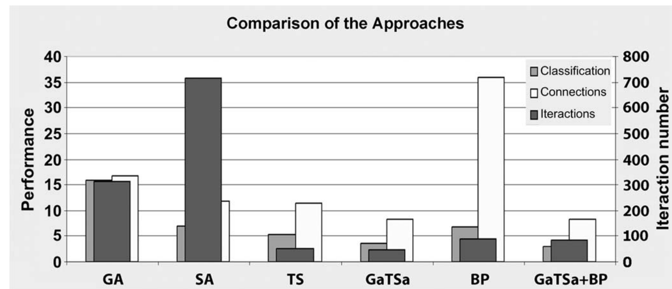

Fig. 4. GaTSa performance optimization in artificial nose.

TABLE VI
MAXIMUM MLP ARCHITECTURE USED IN THE EXPERIMENTS

	N1	N2	N3	N_{max}
Artificial Nose	6	10	3	90
Iris	4	5	3	32
Thyroid	21	10	3	240
Diabetes	8	10	2	100
Card	51	8	2	424
Cancer	9	10	2	110
Glass	9	9	6	135
Heart-Cleveland	35	9	2	333
Horse	58	10	3	610
Soybean	82	10	19	1.010
Mackey-Glass	4	4	1	50

The technique that combined the heuristics of TS, SA, and GA obtained the best performance. This technique worked better without using the local search heuristic (GaTSa without BP, performing only the global search phase) to optimize the ANN connection values. Using GaTSa + BP, the average classification error was 2.86%, with an average of 8.33 connections from 36 possible connections in a fully connected ANN. Using a fully connected network, the local optimization technique BP obtained an average error of 6.30%.

Fig. 4 shows the graphs comparing the performances of the investigated techniques. The proposed technique obtained the best results regarding the classification error, the final network connection number, and the number of iterations needed for architecture optimization.

2) GaTSa—Variable Architecture Experiments: Table VI presents the maximal architecture ($N1$—input units, $N2$—hidden units, $N3$—output units, and $N_{\max}$—maximum number of connections) for SA, TS, GA, and TSa. In all ANN architectures, the $N1$ and $N3$ values are problem dependent, and $N2$ was obtained by experiments (Table IV). For GaTSa, the

TABLE VII
OPTIMIZATION TECHNIQUE PERFORMANCE IN MLP OPTIMIZATION

		SA	TS	GA	TSa	GaTSa+BP
Artificial Nose	Class. (%)	3.3689	3.2015	13.3884	1.4244	0.7914*
	Stand. Dev.	3.1223	3.3121	2.9800	2.342	2.1222
	Input	5.9400	5.9667	4.9667	5.8800	5.2267
	Hidden	7.8067	8.0667	1.9333	7.0567	4.4867
	Connec. (%)	39.3000	40.7036	31.3556	32.3370	36.2254
Iris	Class. (%)	12.6496	12.4786	2.5641*	4.6154	5.2564
	Stand. Dev.	6.3212	7.9887	6.3221	5.3221	4.9881
	Input	2.8500	2.8767	2.4333	2.7100	3.3600
	Hidden	2.7567	3.4867	1.5667	2.6567	4.1333
	Connec. (%)	26.0728	25.9375	22.9792	24.2603	31.8538
Thyroid	Class. (%)	7.3813	7.3406	7.2850	7.3322	7.1509
	Stand. Dev.	1.0112	1.2112	1.356	1.001	0.8901
	Input	20.7700	20.7700	12.8333	20.3700	7.1233
	Hidden	7.2267	7.4667	2.4000	6.3900	2.0833
	Connec. (%)	34.8875	35.8916	14.3902	29.8111	12.6400
Diabetes	Class. (%)	27.1562	27.4045	25.9948	25.8767	27.0615
	Stand. Dev.	2.8737	2.8641	4.2954	3.2626	3.1088
	Input	7.7600	7.7800	4.9667	7.5633	1.5100
	Hidden	5.2700	5.3700	1.9333	4.5300	1.2000
	Connec. (%)	30.3833	30.8167	18.7033	25.5067	9.0975*
Card	Class. (%)	23.4690	18.0426	31.7248	21.2694	15.2422*
	Stand. Dev.	8.8920	2.8878	6.0588	5.6037	1.2111
	Input	50.4667	50.5333	51.000	50.4000	50.333
	Hidden	5.1333	5.3333	7.0000	5.3000	5.000
	Connec. (%)	41.7453	44.5126	25.1222	43.4103	38.4419
Cancer	Class. (%)	7.1729	7.2779	7.4220	6.2846	7.1920
	Stand. Dev.	3.0291	3.1348	7.5863	4.4274	4.0314
	Input	8.9000	8.8333	9.000	8.9333	6.0300
	Hidden	6.2333	5.5333	8.1667	5.5333	3.2600
	Connec. (%)	35.7273	33.4545	57.6060	34.3636	27.3379*
Glass	Class. (%)	58.3810	56.4127	58.0317	57.7778	55.1428
	Stand. Dev.	4.7989	7.0948	4.6451	6.2893	6.0820
	Input	8.9667	8.9000	8.9667	8.833	7.400
	Hidden	7.1000	6.5333	8.9667	6.4000	6.500
	Connec. (%)	31.8025	32.9630	55.6518	31.0370	31.1730
Heart-Cleveland	Class. (%)	24.9227	23.5099	30.0221	24.9007	22.2075
	Stand. Dev.	5.4095	3.5801	7.8017	4.0174	3.4422
	Input	34.8000	34.7000	35.000	34.7000	29.93
	Hidden	6.2667	6.0000	7.6333	5.6667	5.433
	Connec. (%)	42.0320	41.6116	61.3903	41.0711	38.3292
Horse	Class. (%)	39.9227	38.6996	41.5385	39.2857	38.6996
	Stand. Dev.	2.6736	1.1054	4.5529	2.4434	1.5851
	Input	57.5333	57.5000	58.000	57.7667	50.030
	Hidden	5.9333	5.2667	9.3000	4.1333	5.1000
	Connec. (%)	32.7760	28.9617	60.7639	26.7978	35.8939
Soybean	Class. (%)	82.7647	81.2059	76.6961	84.7451	62.9412*
	Stand. Dev.	10.6589	12.6064	2.9971	9.5589	5.6785
	Input	81.9000	81.9333	81.9667	81.8333	82.000
	Hidden	6.8000	6.2667	10.000	6.7667	21.033
	Connec. (%)	31.8680	32.6634	55.6930	29.9868*	45.4494
Mackey Glass	SEP Test	2.0172	0.8670	0.6542	0.6847	0.72164
	Stand. Dev.	0.9088	0.9011	0.7200	0.8210	0.9123
	Input	3.6167	3.7967	2.2667	3.4567	1.0533
	Hidden	1.9000	2.2700	1.3667	1.8933	1.0100
	Connec. (%)	19.2600	24.1400	15.9533	17.1334	11.4923*

same values with $N1$ and $N3$ are used, but the value of $N2$ is optimized together with the network weights and connections in a constructive manner.

Table VII displays the average performance of each investigated optimization technique. These results were obtained for each technique by the optimization of the number of connections and weight connection values of an MLP. The cost function used was average. The parameters evaluated were the following: 1) the SEP and classification error (class) of the test set; 2) the mean number of the input units; 3) the mean number of the hidden units; and 4) the percentage of network connections.

For all data sets, the optimized ANN obtains a lower classification error than those obtained by MLP without architecture optimization (Table IV), and the mean number of connections is lower than the maximum number allowed. In most of the simulations, the best performance to optimize the ANN architecture was obtained by GaTSa.

In the experiments with GaTSa, the average of the number of connections was computed relative to the maximum network architecture generated, rather than it being calculated with the maximum fixed architecture (as in the other models). This seemed to be the fairest approach. However, it seemed to have harmed the model because, most of the time, GaTSa generated an architecture with less connections than the maximum allowed.

A paired-difference t test with a 95% confidence level was applied in order to confirm the statistical significance of these conclusions. In Table VII, statistically significant results are indicated by asterisk. Statistically, GaTSa achieves better optimization of the architecture input nodes. Despite its finding architectures with a smaller number of hidden nodes, all methods were statistically equivalent regarding the optimization of these units. This is an indication that the constructive strategy works in the definition of the number of hidden nodes. The MLP performance obtained from the optimized ANN was statistically equivalent for the thyroid, diabetes, cancer, glass, Heart-Cleveland, horse, and Mackey–Glass data sets. The GaTSa method obtained better results in the artificial nose, card, and soybean data set, whereas GA had the best performance in the iris data set.

3) GaTSa—Cost Function Influence: In the experiments, the maximal architecture is defined in Table VI. Table VIII displays the experimental results with different cost functions. The evaluated cost functions are the following: the average method (average), weighted average (WA), weight decay (WD), multiobjective (MO), and combination of multiobjective and weight decay (MO+WD). The parameters evaluated were the following: 1) the SEP and classification error (class) of the test set; 2) the mean number of the input processing units; 3) the mean number of the hidden processing units; and 4) the percentage of the network connections.

In the artificial nose data set, the best classification results were obtained by the multiobjective approach, and the best architecture optimization was found by the weight decay method. The combination of weight decay and genetic operators using multiobjective optimization presented the best performance in the iris, horse, soybean, Heart-Cleveland, card, and glass data sets. The weight decay presented the best optimization performance in the thyroid, diabetes, and Mackey–Glass data sets. In the cancer data set, the multiobjective cost function presented the best optimization performance.

The rate of success of using each of the five cost functions in the optimization techniques was studied. Besides the better performance of weight decay, the different cost functions presented unequal behavior in each data set. This behavior can be explained by the presence of noise in the data sets.

Noisy data sets are complex problems in ANN training. In some analyses, the artificial nose, diabetes, and thyroid data sets presented absolute deterministic and absolute random noises.

TABLE VIII
COST FUNCTION EXPERIMENTAL RESULTS

		Average	WA	WD	MO	MO+WD
Artificial Nose	Class. (%)	11.8595	11.0807	7.5462	7.0407	12.6237
	Stand. Dev.	2.3466	2.2329	4.0355	3.4536	6.3426
	Input	5.9967	6.000	5.9233	5.9967	5.9867
	Hidden	9.1533	8.5133	7.5600	8.4833	8.6200
	Connec. (%)	50.2400	53.3834	33.2433	49.6500	42.0467
Iris	Class. (%)	6.1197	5.4615	6.9316	4.2735	3.9829
	Stand. Dev.	4.2170	2.1070	3.9070	2.0110	3.0100
	Input	3.5667	3.8367	3.8167	3.8567	3.1467
	Hidden	3.7967	3.1900	4.4367	3.5500	3.0767
	Connec. (%)	13.3867	16.1367	18.1167	15.8600	9.8000
Thyroid	Class. (%)	7.1024	7.1798	6.8196	6.9270	6.8609
	Stand. Dev.	1.6500	1.4510	2.0981	1.3070	1.9080
	Input	20.6733	20.9300	20.7800	20.9800	20.9767
	Hidden	7.1833	3.5633	7.8067	8.5233	8.5167
	Connec. (%)	83.8800	99.3200	91.6700	114.6667	115.0933
Diabetes	Class. (%)	28.4583	28.4323	25.7552	28.2639	25.8542
	Stand. Dev.	3.5030	3.3450	3.9070	3.0070	4.3480
	Input	7.7367	7.9267	7.7767	7.9700	7.9667
	Hidden	5.6100	2.3067	5.2000	4.4133	6.3100
	Connec. (%)	31.8433	38.1833	31.7433	43.2700	42.0367
Card	Class. (%)	17.1027	18.7694	39.9516	21.6279	40.4554
	Stand. Dev.	2.5686	4.0774	9.0395	3.1156	9.9026
	Input	50.9667	50.9667	50.53333	50.9667	50.8000
	Hidden	6.6000	2.6000	5.1667	6.1333	5.1000
	Connec. (%)	242.33	260.50	198.66	247.86	216.5667
Cancer	Class. (%)	5.1576	5.3295	17.8415	5.7784	21.8911
	Stand. Dev.	2.3070	2.0629	10.7535	1.8510	14.9891
	Input	8.9667	9.0000	8.8000	9.0000	8.9667
	Hidden	7.000	3.6000	5.4000	7.2333	6.8000
	Connec. (%)	45.9667	52.33	33.66	50.93	43.90
Glass	Class. (%)	51.9365	48.5397	73.1111	54.8571	73.7143
	Stand. Dev.	6.3850	5.4823	11.3728	4.6428	13.3558
	Input	8.9667	9.0000	8.7000	8.9667	8.9333
	Hidden	8.9000	8.6667	7.4000	8.7667	8.6667
	Connec. (%)	65.13	79.7667	39.2333	66.90	53.2667
Heart-Cleveland	Class. (%)	21.6777	21.4570	35.5629	23.7969	35.7616
	Stand. Dev.	3.1829	5.2134	8.7515	3.7420	8.7651
	Input	34.9333	34.9333	34.8667	34.9667	34.9333
	Hidden	7.3667	3.5000	6.0333	7.3333	6.1000
	Connec. (%)	185.16	200.06	151.40	189.06	167.2333
Horse	Class. (%)	39.3040	39.6520	47.755	41.4286	50.0000
	Stand. Dev.	2.91000	2.7075	10.3868	3.5096	11.0089
	Input	58.0000	58.0000	57.9333	58.0000	57.9333
	Hidden	9.3000	4.0667	7.9000	8.3667	8.0333
	Connec. (%)	354.60	353.80	303.00	338.86	310.36
Soybean	Class. (%)	65.9000	64.0000	69.2900	77.432	74.5202
	Stand. Dev.	4.2161	4.4703	5.2232	3.7609	3.2086
	Input	82.000	82.000	819333	81.9667	81.9000
	Hidden	10.000	10.000	10.000	10.0000	10.0000
	Connec. (%)	641.60	681.6333	514.83	569.36	505.86
Mackey Glass	SEP Test	0.6215	0.8070	0.2721*	0.5745	0.6293
	Stand. Dev.	0.9130	0.8910	0.9820	0.9202	0.9800
	Input	1.1067	1.4100	1.0533	2.0933	1.7867
	Hidden	1.0000	1.0000	1.0067	1.0100	1.1800
	Connec.	2.1533	2.5433	2.0833*	3.6300	3.3233

The sources of absolute deterministic noise are computational errors and systematic measurement errors. Absolute random noise is typical in optimization problems such as adaptation, learning, and pattern recognition. This noise in the data sets probably influenced the experiments, but the average performance of the cost functions was confirmed.

The better performance of weight decay and multiobjective approaches demonstrates the capacity of this method in restricting the type of functionality that the network can produce by favoring networks that produce smoother functions. Smooth output functions are generally more likely to represent the underlying functions of the real-world data. Moreover, the use of weight decay can modify the error surface of a given problem in such a way as to reduce the growth of large update values. In addition, the parameter μ was empirically determined. A fine tuning in this parameter may improve the results.

TABLE IX
FEATURE SELECTION PERFORMANCE

	Iris		Diabetes		Thyroid	
	Attrib.	Class.	Attrib.	Class.	Attrib.	Class.
All Features	4	6	8	27.53	21	7.04
MLP	4	6.83	8	27.08	21	7.38
BF-B	2	4.29	5	31.58	10	1.44
BF-R	2	3.71	4.5	30.79	8.2	2.61
BF-F	1	4.29	2	30.00	5	1.08
BS-B	2	4.29	5	31.58	10	1.44
BS-R	1.9	4.14	3.9	30.21	8.9	2.19
BS-F	1	4.29	4	31.32	5	1.08
HC-B	2	4.29	5	31.58	10	1.44
HC-R	1.9	15.14	5.4	31.50	12.9	3.98
HC-F	1	4.29	2	30.00	5	1.08
RB-All	2	4.29	4	28.68	2	1.08
RB-R	1.8	4.71	4.4	30.68	2	1.32
RB-None	2	4.29	7	30.00	5	1.08
Las Vegas	1	4.29	4	31.32	8.9	1.78
GaTSa	3.36	5.23	1.51	27.06	7.12	7.15

The use of multiobjective optimization in genetic operators presented interesting results in some data sets but exhibited a poor performance in most. The main problem with this approach is the construction of the Pareto ranking. There is no efficient algorithm for checking nondominance in a set of feasible solutions. Traditional algorithms have serious performance degradation as the size of the population and the number of objectives increase [6]. The multiobjective approach presented some outlier results in the experiments.

Another possible problem was to choose the best solution when there were several solutions at the same position in the Pareto ranking. In such a case, the solution with the lower classification error was chosen. However, there is no guarantee that this solution has a small number of connections. Possibly, the best solution would be to choose the solution with the best average of classification errors and connection number. These problems hampered the performance of the combination of weight decay and multiobjective optimization.

The best problem search space exploration was achieved with the use of the combination of weight decay and multiobjective approach. This method generated solutions with low complexity architecture and low number of errors.

4) GaTSa—Feature Subset Selection: Table IX displays the results of the k-NN classifier (for classical methods) and MLP (for the GaTSa method) in data sets with all attributes. The k-NN was used to determine the classification accuracy of the solutions (network input configuration) obtained by the classical methods. The labels Attrib. and Class. mean the number of features selected by the technique and the classification performance, respectively.

This table displays the average performance—the number of attributes following feature selection and performance classification—containing the results of the feature selection technique with the forward (F), backward (B), and random (R) search strategies, e.g., the best-first experiments are labeled as BF-F, BF-B, and BF-R, respectively.

The performance of the algorithms was obtained in a tenfold cross-validation process. The same method was used to evaluate the precision of the classifier with all features. The results of the random strategy correspond to the average of ten runs due to the random characteristics of this model.

In the experiments, there were similarities between the best-first (BF) and beam search (BS) algorithms. Although using a $beam$ size of 15 and having the number of expansions without improvement (ϱ) set at 50, the two algorithms presented very close results. It is possible that the difference between the methods is only apparent in data sets with more attributes. In the performed experiments, the explored search space was the same in both methods.

The results from the best-first and beam search were obtained with a high computational effort. For example, in the thyroid data set, over 2000 subsets were evaluated before the algorithms presented the best found solution. This result was found even with setting a low number of expansions without improvement.

The number of selected features reflected most of the differences between the variants (forward, backward, and random). The forward variant obtained solutions with the least attributes, mostly in the thyroid data set (better than 76% reduction in data set dimensionality).

The best-first and beam search methods obtained interesting results regarding feature selection. The k-NN classifier also exhibited a good performance. In the thyroid data set, k-NN obtained better results than the MLP that is fully connected with all attributes (MLP obtained a 7.38% classification error). The characteristics of the data set can explain these results. This database has a nonbalanced data distribution. The class probability distributions are 5.1%, 96.2%, and 2.3%, respectively.

Unlike previous experiments, the hill-climbing forward variant obtained worse results than the backward and random variants. The characteristic of the forward strategy is the evaluation of a small number of subsets.

In the Las Vegas simulations, the maximum number of subsets generated without improvement was 50. The Las Vegas method does not exhibit variations in the search strategy. This algorithm is characterized as random search. It cannot perform the forward or backward strategies.

The simulations carried out with the Las Vegas and beam search algorithms (random) obtained the best results in the fisher iris data set where only one attribute was selected. In the thyroid and diabetes data sets, the best algorithms were hill-climbing and random bit climber (random), with two and five attributes selected, respectively. The method classification performance was similar in almost all experiments. The classification degradation regarding simulations with all attributes was low.

Most of the algorithms using the forward variant presented less subset evaluation than the backward and random strategies. The evaluation of a few subsets can hinder the selection of interrelated attributes (attributes that produce better results when combined than when isolated).

The forward method is faster than its backward counterpart. This is expected because the forward method starts with small subsets and enlarges them, whereas the backward method starts with large subsets and shrinks them. It is computationally more expensive to determine the criterion value for large subsets than for small subsets.

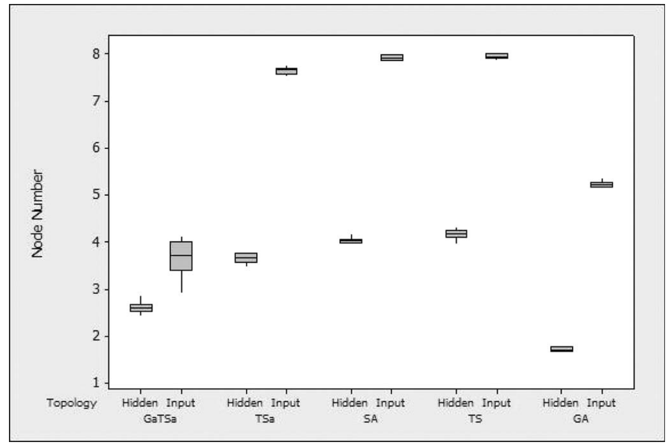

Fig. 5. Topology optimization of the optimization techniques.

The proposed GaTSa method obtained interesting results in relevant feature selection. GaTSa obtained results that are very close to classical feature selection methods in the problems investigated. Even if it is not being specifically developed for relevant feature selection, the GaTSa method, even though in an indirect way, is able to eliminate features that are nonrelevant to learning algorithms.

In order to find suitable solutions, the remaining search techniques require the definition of a good initial ANN architecture. The method may eliminate network connections through pruning. GaTSa is able to automatically optimize the network size in the search.

In case of GaTSa, the mean number of connections was less than all remaining approaches. It can be seen that the method is able to perform a better exploration in the architecture search space due to a combination of the advantages of GA, SA, and TS in order to generate an MLP with a small number of connections and high classification performance. In the search process, irrelevant connections are eliminated from the network architecture through pruning. The integration of SA and TS has the same characteristics, but the use of GA operators incorporates more domain-specific knowledge in the search process. Fig. 5 shows a summary of the results.

A number of differences between the feature selection methods and the proposed algorithm are evident. One of the main differences is the random characteristics of the model. In this kind of method, the expected value will depend upon the random choices in the algorithm and not on the imposed probability distribution of the input features. This behavior distinguishes the expected processing time in random algorithms from the mean case complexity normally used in deterministic algorithm analysis.

It is important to say that the behavior of the random algorithms may change even when repeatedly applied to the same entrance. Thus, the processing time is a random variable, and the processing time analysis requires a better comprehension of the associated probability distribution.

Intuitively, a higher number of database attributes mean a greater classifier discriminatory power and a greater facility in

TABLE X
SELECTED FEATURES BY THE TECHNIQUES

Method	Database Thyroid (Attributes)											
Beam Search	3	8	5	17	7	18	19	4	6	12	13	15
Best-First	3	8	17	7	21	19	13	12	5	18	15	9
Hill-Climbing	3	12	8	5	17	15	9	4	12	18	19	11
Random Bit Climber	3	8	13	17	15	21	7					
Las Vegas	3	17	8	21	7	10	2	4	5			
GaTSa	3	8	5	17	21	7	12	15	13			
	Database Diabetes (Attributes)											
Beam Search	2	7	5	8	1							
Best-First	2	8	4	5	6	1						
Hill-Climbing	2	8	1	4	6	5	7					
Random Bit Climber	2	1	8	6	5							
Las Vegas	2	5	7	8								
GaTSa	2	8										
	Database Fisher Iris (Attributes)											
Beam Search	4	3										
Best-First	4	3										
Hill-Climbing	4	3	2									
Random Bit Climber	3	4										
Las Vegas	4											
GaTSa	4	3	2									

TABLE XI
GaTSa EXPERIMENT CONFIGURATION

	Factors	Levels	
		Inferior (-1)	Superior (+1)
A	Neighborhood Size	5	10
B	Temperature Factor	0.7	0.9
C	Iterations	50	100
D	Number of Micro-evolutions	5	10
E	Crossover Rate	0.7	0.9
F	Mutation Rate	0.1	0.3
G	Tabu List Size	10	20

TABLE XII
ANOVA TABLE

	S.Squares	D.F	M.Square	F-Ratio	Sig.level
Main Factors					
D	317.128	1	317.128	72.43	0.000
E	233.301	1	233.301	53.28	0.000
Sig. Interac.					
DE	100.888	1	100.888	23.04	0.000
AG	17.692	1	17.692	4.04	0.000

extracting the database knowledge models. In practice, however, the real world presents evidence that this is not always true as many methods suffer the curse of dimensionality—the algorithm computational time increases aggressively with the database attribute number. Moreover, the experiments confirm that the number of samples used to ensure a classification rate increases exponentially with the number of irrelevant attributes present in the data set [19].

It is not possible to compare, in a fair manner, the classification performance of the methods used because the classical feature selection algorithms implemented the classifier k-NN in a wrapper model, whereas GaTSa implements MLP. Despite the higher complexity of GaTSa, in the experiments, we found out that the characteristics of some data sets favored the k-NN model. For example, the thyroid data set has a probability distribution that favored one class (92.6% of the data is from the underfunctioning thyroid class).

Table X displays the attributes selected by each investigated technique. The table presents the attributes that, on average, have more relevance in the database. The attributes are sorted in the order of importance, as defined by the number of citations for each algorithm in each simulation performed. As observed, GaTSa presents very similar feature selection results as the classical methods.

C. *Sensivity Test of the GaTSa Parameters*

The design of the experiments was applied in order to determine the factors with the greatest influence on the system's performance. When analyzing the influence of each of these parameters, the designer should pay most attention to the ones presenting values that are statistically most significant. This makes it possible to avoid the necessity for a detailed analysis of different configurations that might, in fact, lead to the design of various models with very similar behavior patterns.

We expected a small number of parameters to have a great influence on model performance in different databases. In this analysis, we verified the most influential parameters and also the interaction and interrelationship between them. In the study performed with GaTSa, we opted to accomplish a factorial experiment with two levels (2^k factorial experiment), seeking to reduce the amount of experiments done. Table XI presents the controlled factors.

The factors $H =$ sigmoid logistic (type of activation function), $I =$ backpropagation (local optimization algorithm), $J = 0.001$ (learning rate), $K = 0.6$ (momentum), $L = 10$ (interaction number to successive temperature reduction), $M = 5$ (k for $GL(k)$ stop criteria), and $N = 1$ (initial temperature) were fixed during the experiments.

The analyses were accomplished in an aleatory manner. Seven control factors (variables) were considered, each one of them with two levels, resulting in 128 combinations. Each level combination of the control factors was accomplished five times, totaling to 640 analyses. Due to the random characteristics of the model, in each of the 640 analyses, 30 runs of the algorithm were performed (one result is the average of 30 runs) so that 19 200 simulations were performed.

Through the variance analysis of the factorial experiment, considering the statistically significant level of 5% in the F distribution, two factors were identified as having a larger influence on the performance of the MLP optimized by the proposed model.

A deeper analysis revealed that, statistically, a smaller number of generation and a larger crossover rate can induce networks with better performance. The differences in the number of solutions generated in each iteration, the initial value of the temperature, and the size of the tabu list did not significantly influence the model result. It was also found that a smaller number of iterations and a higher mutation rate contribute to the success of the model.

Table XII gives the variance analysis. The more relevant factors are the following: the number of microevolutions in the genetic operators (D), corresponding to $\approx$32.64% of the system variance, and the genetic operator crossover rate (E), corresponding to $\approx$24.04% of the variance. The interaction (variation among the differences between means for different levels of one factor over different levels of another) among the factors was also identified: the number of microevolutions in the genetic operators (D) and genetic operator crossover rate (E) (≈10.39% of the system variance) and the neighborhood size (A) and list tabu size (G), corresponding to ≈1.82% of the total data variance.

The experiments confirmed that, despite the large number of the configurable parameters of the method, very few have a significant influence on the performance of the optimized ANNs. This is an interesting characteristic because even the inexperienced designers can successfully use it. The parameters that are most influential in the method performance were the variables that controlled the solution's evolution in the search space. GaTSa is robust to other settings because it did not show a significant change in the network's generated performance.

VII. Final Remarks

This paper has shown that the combination of GA, SA, and TS in the proposed methodology can be successfully used for the simultaneous optimization of the MLP network topology and weights. According to the experiments, based on the network performance, GaTSa outperforms the TSa previously proposed by the authors. GaTSa surpassed other methods from the literature explored in this paper for most problems, and it is situated among those with better performance for the remaining ones. The experiments have also demonstrated that this method can be used for relevant feature selection. Additionally, the experiments have indicated that the most relevant parameters in GaTSa are the number of microevolutions in the genetic operators and the genetic operators' crossover rate.

In the context of feature selection, the main disadvantage of the proposed method is the difficulty in getting a good performance by the model when some information in the database is missing and when the classification task is not performed with the required low errors. This occurs because the elimination of the connections does not take each input node into consideration. The elimination of an input node only happens if all connections that connect this node are eliminated. The nonselection of a feature occurs as a consequence of the process of connection reduction. The proposed method does not verify the contribution of each input during optimization, and the emphasis is on the contribution of each connection.

As future work, the time complexity of GaTSa must be analyzed, and ways of reducing time consumption must be proposed. Other recently proposed optimization heuristics such as particle swarm optimization could be explored [37]. GaTSa will be explored in solving some real-world problems in science, business, technology, and commerce.

References

[1] D. W. Aha and R. L. Bankert, "A comparative evaluation of sequential feature selection algorithms," in *Proc. 5th Int. Workshop Artif. Intell. Statist.*, 1995, pp. 1–7.

[2] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proc. Int. Conf. Genetic Algorithms Appl.*, 1987, pp. 14–21.

[3] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," University of California, Irvine, CA, 1998.

[4] A. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artif. Intell.*, vol. 97, no. 1/2, pp. 245–271, Dec. 1997.

[5] E. Cantú-Paz and C. Kamath, "An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 5, pp. 915–927, Oct. 2005.

[6] C. A. C. Coello, "An empirical study of evolutionary techniques for multiobjective optimization in engineering design," Ph.D. dissertation, Tulane Univ., New Orleans, LA, 1996.

[7] A. Corana, M. Marchesi, C. Martini, and S. Ridella, "Minimizing multimodal functions of continuous variables with the simulated annealing algorithm," *ACM Trans. Math. Softw.*, vol. 13, no. 3, pp. 262–280, Sep. 1987.

[8] L. Davis, "Bit-climbing, representational bias, and test suite design," in *Proc. 4th Int. Conf. Genetic Algorithms*, 1991, pp. 18–23.

[9] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Hoboken, NJ: Wiley, 2000.

[10] C. M. Fonseca and P. J Fleming, "Multi-objective optimization and multiple constraint handling with evolutionary algorithms—Part I: A unified formulation," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 28, no. 1, pp. 26–37, Jan. 1998.

[11] B. L. Fox, "Uniting probabilistic methods for optimization," in *Proc. Conf. Winter Simul.*, 1992, pp. 500–505.

[12] A. Gepperth and S. Roth, "Applications of multi-objective structure optimization," *Neurocomputing*, vol. 69, no. 7–9, pp. 701–713, Mar. 2006.

[13] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, May 1986.

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

[15] L. Hamm, B. W. Brorsen, and M. T. Hagan, "Global optimization of neural network weights," in *Proc. Int. Joint Conf. Neural Netw.*, 2002, vol. 2, pp. 1228–1233.

[16] G. E. Hinton, "Learning distributed representations of concepts," in *Proc. Annu. Conf. Cogn. Sci. Soc.*, 1986, pp. 1–12.

[17] R. Huber, R. Schwaiger, and H. A. Mayer, "On the role of regularization parameters in fitness functions for evolutionary designed artificial neural networks," in *Proc. World Congr. Neural Netw.*, 1996, pp. 1063–1066.

[18] S. Kirkpatrick, C. D. Gellat, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[19] P. Langley and W. Iba, "Average-case analysis of a nearest neighbor algorithm," in *Proc. 13th Int. Conf. Artif. Intell.*, 1993, pp. 889–894.

[20] H. Liu and R. Setiono, "Feature selection and classification—A probabilistic wrapper approach," in *Proc. 9th Int. Conf. Ind. Eng. Appl. Artif. Intell.*, 1996, pp. 284–292.

[21] T. B. Ludermir, A. Yamazaki, and C. Zanchettin, "An optimization methodology for neural network weights and architectures," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1452–1459, Nov. 2006.

[22] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, Jul. 1977.

[23] R. Martıacute; and A. El-Fallahi, "Multilayer neural networks: An experimental evaluation of on-line training methods," *Comput. Oper. Res.*, vol. 31, no. 9, pp. 1491–1513, Aug. 2004.

[24] N. G. Metin, B. Sahiner, H. Chan, L. Hadjiiski, and N. Petrick, "Optimal selection of neural network architecture for CAD using simulated annealing," in *Proc. EMBS Int. Conf.*, 2000, pp. 1325–1330.

[25] P. P. Palmes, T. Hayasaka, and S. Usui, "Mutation-based genetic neural network," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 587–600, May 2005.

[26] V. W. Porto, D. B. Fogel, and L. J. Fogel, "Alternative neural network training methods," *IEEE Expert*, vol. 10, no. 3, pp. 16–22, Jun. 1995.

[27] L. Prehelt, "Proben1—A set of neural network benchmark problems and benchmarking rules," Univ. Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21, 1994.

[28] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Netw.*, vol. 4, no. 5, pp. 740–747, Sep. 1993.

[29] S. M. Sait and H. Youssef, *Iterative Computer Algorithms With Applications in Engineering: Solving Combinatorial Optimization Problems*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1999.

[30] R. J. Sexton, B. Alidaee, R. E. Dorsey, and J. D. Johnson, "Global optimization for artificial neural networks: A tabu search application," *Eur. J. Oper. Res.*, vol. 106, no. 2/3, pp. 570–584, Apr. 1998.

[31] R. S. Sexton, R. E. Dorsey, and J. D. Johnson, "Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing," *Eur. J. Oper. Res.*, vol. 114, no. 3, pp. 589–601, May 1999.

[32] S. W. Stepniewski and A. J. Keane, "Pruning back-propagation neural networks using modern stochastic optimization techniques," *Neural Comput. Appl.*, vol. 5, no. 2, pp. 76–98, Jun. 1997.

[33] G. Sywerda, "Uniform crossover in genetic algorithm," in *Proc. Int. Conf. Genetic Algorithms*, 1989, pp. 2–9.

[34] J. Tsai, J. Chou, and T. Liu, "Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 69–80, Jan. 2006.

[35] H. Vafaie and K. De Jong, "Robust feature selection algorithms," in *Proc. 5th IEEE Int. Conf. Tools Artif. Intell.*, 1993, pp. 356–363.

[36] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight elimination with application to forecasting," in *Proc. Adv. Neural Inf. Process. Syst. III*, 1991, pp. 875–882.

[37] J. Yu, S. Wang, and L. Xi, "Evolving artificial neural networks using an improved PSO and DPSO," *Neurocomputing*, vol. 71, no. 4–6, pp. 1054–1060, Jan. 2008.

Cleber Zanchettin received the Ph.D. degree in computer science from the Federal University of Pernambuco, Recife, Brazil, in 2008.

He is a Professor at the Center of Informatics, Federal University of Pernambuco. He is a technical reviewer, and he has published papers in the areas of pattern recognition, artificial neural networks (ANNs), and intelligent systems. His research interests include hybrid neural systems and applications of ANNs.

Teresa B. Ludermir received the Ph.D. degree in artificial neural networks from Imperial College, University of London, London, U.K., in 1990.

From 1991 to 1992, she was a Lecturer at Kings College London, London. She joined the Center of Informatics, Federal University of Pernambuco, Recife, Brazil, in September 1992, where she is currently a Professor and the Head of the Computational Intelligence Group. She has published over 200 articles in scientific journals and conference proceedings and three books on neural networks (NNs) and has organized two of the Brazilian Symposium on Neural Networks. She is one of the Editors-in-Chief of the *International Journal of Computational Intelligence and Applications*. Her research interests include weightless NNs, hybrid neural systems, and applications of NNs.

Leandro Maciel Almeida received the B.S. degree in information systems from the Lutheran University of Brazil/University Lutheran Center of Palmas, Tocantins, Brazil, in 2004, and the M.S. degree in computer science from the Federal University of Pernambuco, Recife, Brazil, in 2007, where he is currently working toward the Ph.D. degree in the Center of Informatics.

His research interests include artificial neural networks, pattern recognition, hybrid intelligent systems, and evolutionary computation.