A methodology to training and optimize artificial neural networks weights and connections

Cleber Zanchettin and Teresa B. Ludermir

Abstract—This work presents a new methodology that integrates the heuristics tabu search, simulated annealing, genetic algorithms and backpropagation in a pruning and constructive way. The approach obtained promising results in the simultaneous optimization of the artificial neural network architecture and weights of four classification and one prediction problem.

I. INTRODUCTION

The performance of an artificial neural networks depends upon the selection of proper weight connections and network topology during the neural network training. Due to the complex nature of neural networks training, even simple functions can have very complex error surfaces. Since the nature of the neural networks learning algorithms is to local convergence, it can be demonstrated that solutions are highly dependent upon the initial random draw of connection weights. If these initial weights are located on a local grade, which is probable, the learning algorithm will likely become trapped in a local solution that may or may not be the global solution.

Another critical subject involved in the neural network training is the relation stability versus plasticity on the architecture definition. A lack of network connections can render a neural network unable to solving the investigated problem as a result of the inadequacy of adjustable parameters. Whereas an excess of connections can cause overfitting in the training data and fail to have an adequate generalization capacity.

Basically, there are four approaches to define the neural network architecture [13]: (1) the Empiric approach, consists of testing several topologies until finding one that present satisfactory results; (2) the Search optimization approach, consists to generate variations of a neural network and to combine the best characteristics of this new network to improve the performance; (3) the Prunning approach, consists of optimizing a network by the elimination of the elements (network connections) that don't influence the generalization performance; and (4) the Constructive approach, the networks begin with a minimum topology and the architecture is built during the training process.

This paper presents a new methodology to optimize Multi-Layer Perceptron neural networks (MLP) that integrates the main potentialities these approaches: (1) search optimization to generate new solutions; (2) prunning to eliminate connections and optimize the network size; and (3) the constructive way to find the best network topology.

Cleber Zanchettin and Teresa B. Ludermir are with the Centro de Informática, University of Pernambuco, Recife, PE P. O. Box 7851, 50.732-970, BRA (phone: +55-81-2126-8430; fax: +55-81-2126-8438; email: cz@cin.ufpe.br and tbl@cin.ufpe.br)

Genetic Algorithms (AG) [7], Simulated Annealing (SA) [9] and Tabu Search (TS) [6] are the most popular from the optimization iterative algorithms. All three optimization heuristics have similarities [17]: (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, 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; all that is required is to have a suitable solution representation, a cost function, and a mechanism to traverse the search space; and (5) Under certain conditions, they asymptotically converge to an optimal solution.

To perform the search network architecture optimization, this new methodology integrate these three heuristics, combining the potentialities to overcome their limitations. In experiments, to validate the method, are performed four classification simulations: (1) The odor recognition problem in artificial noses [5]; (2) Diagnose diabetes of Pima indians [4]; (3) Fisher’s Iris data set [2]; (4) Thyroid data set [16]. And one prediction simulation: (1) Mackey-Glass time series [10].

Next section describes the investigated optimization techniques. In Section 3 and Section 4 are presented the performed experiments and discussion. Section 5 contains the final remarks.

II. SEARCH HEURISTICS DESCRIPTION

A. Genetic Algorithms

Genetic algorithms emerged through the analogy between optimization, species genetic mechanisms and natural evolution. They are based on the chromosome representation of the optimization variables in the reproduction process and genetic operators such as crossover and mutation [8].

The genetic algorithm is characterized by a parallel search of the state space as against a point-by-point search through conventional optimization techniques. The parallel search is accomplished by keeping a set of possible solutions for the optimization problem, called population. An individual in the population is a string of symbols and is an abstract representation of the solution. The symbols are called genes and each string of genes is termed a chromosome. The individuals in the population are evaluated through a fitness measure. The population of chromosomes evolves from one generation to the next through the use of two types of genetic operators: (1) unary operators, such as mutation and inversion, which
thus, the algorithm chooses the new solution that produces the largest improvement or the smallest deterioration in the cost function. This strategy allows the method to escape from local minima. A tabu list is used to store a certain amount of recently visited solutions. The solutions in tabu list are marked as forbidden to subsequent iterations. The tabu list registers \( T \) last visited solutions. When the list is full, a new movement is registered in substitution to the older movement kept on the list.

In the present work, a neighborhood with 20 solutions is used, and the algorithm chooses the best non-tabu solution. The proximity criterion \[18\] was used to compare two solutions. A new solution is considered identical to the tabu solution one if: (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 \( \pm N \) of the corresponding connection weight in the tabu solution. The parameter \( N \) is a real number with a value of 0.001. A maximum number of 100 iterations is allowed. The stop criterion \( GL_5 \) was also employed.

D. Integration of Simulated Annealing and Tabu Search

The integration of Simulated Annealing and Tabu Search (TSa) to the Neural Network optimization was proposed by [1]. The method combines the accepting new solution scheme of Simulate Annealing with the search and tabu list scheme of the method Tabu Search. In this approach, a set of solutions with a fixed size (the maximum network topology need to be defined) is generated in each iteration, and the best one (i.e., the one with lower cost) is selected according to the cost function, as performed by tabu search. However, the best solution is not always accepted since this decision is guided by a probability distribution, which is the same used by simulated annealing. During the algorithm execution the topology and the weights are optimized, and the best solution found so far (sBSF) is stored.

In this method, to generate a new solutions, from the current solution \( s = (C, W) \), a new solution \( s' = (C', W') \) is generated, where \( C' = (c'_1, c'_2, ..., c'_{N_{max}}) \) and \( W' = (w'_1, w'_2, ..., w'_{N_{max}}) \), in the following way: a random number is generated from an uniform distribution in \([0, 1] \). The new connectivity bit \( c'_i \) for network connection \( i \) is given by:

\[
d_i' = \begin{cases} 
    \overline{c}_i, & \text{if } \alpha \leq p \\
    c_i, & \text{if } \alpha > p
\end{cases} \tag{1}
\]

where \( \overline{c}_i \) is the inverse of the bit \( c_i \), and \( p \) is the probability of inverting each connectivity bit. Then, another random number \( \beta \) is generated from an uniform distribution in the interval \([-1.0, +1.0] \), and the new network weight \( w'_i \) of connection \( i \) is given by:

\[
w'_i = w_i + \beta \tag{2}
\]

Therefore, the generation mechanism acts as follows: first the connectivity bits for the current solution are changed according to a given probability \( (p) \). This operation deletes some network connections and creates new ones. Then, a
random number taken from an uniform distribution in [-1.0, +1.0] is added to each connection weight. These two steps change both topology and connection weights to produce a new solution.

E. Integration of Simulated Annealing, Tabu Search and Genetic Algorithms in a Constructive way

The simulated annealing 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 tabu search method, in contrast, evaluates one group of new solutions at each iteration (instead of only one solution as in simulated annealing). This makes a tabu search faster, as it generally needs less iterations to converge. The genetic algorithm evolution, in turn, involves a sequence of iterations, where a group of solutions evolves through selection processes and reproduction. This process, which is more elaborate than the other algorithms, can result in solutions with a larger quality.

These observations motivated the proposal of an optimization methodology that combines the main potentialities of genetic algorithms, simulated annealing and tabu search in an effort to avoid their limitations.

In general terms, the method works in the following manner: the initial solution have a randomly size, at each iteration, a group of new solutions is generated, starting from the micro-evolution of the current population, as in genetic algorithms. The cost of each solution is evaluated, and the best solution is chosen, as in tabu search. However, differently from a tabu search, this solution is not always accepted. The acceptance criterion is the same used in the simulated annealing algorithm - if the chosen solution has a smaller cost than the current solution, it is accepted; otherwise, it can either be accepted or not, depending on a probability calculation. This probability is given by the same expression used in the simulated annealing method. The visited solutions are marked as tabu, as in a tabu search. In the course of the search the chromosome size is increase, in a constructive way, to find best solutions, according the acceptance criterion. During the optimization process, only the best solution found is stored, that is, the final solution comes back through the method.

The pseudo-code of the proposed method is presented in Algorithm 1. Let \( S \) be a group of solutions and \( f \) 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 the temperature \( T_i \) of the iteration \( i \) to each \( I_T \) algorithm iterations. At each iteration, a new population with \( k \) solutions of size \( z \) is generated. A genetic micro-evolution of \( g_n \) generations is used to generate this population from the current population. The micro-evolution combines the best population solutions and in the process create and eliminate network connections, like a pruning process. The initial solution is coded with the minimum valid network topology and new hidden nodes are added according the constructive process. It is interesting to remember that each solution contains information on the topology and weights of a MLP neural network. Moreover, at the end of the global search (GaTSA), a hybrid training is used, combining the proposed method with a local search technique. The local search technique can be implemented, for instance, by the well-known backpropagation algorithm.

F. Representation of Solutions

In this work, all MLP topologies have a single hidden layer network, containing only connections between adjacent layers. The maximal topology must be defined, which contains \( N1 \) input nodes, \( N2 \) hidden nodes and \( N3 \) output nodes. The parameters \( N1 \) and \( N3 \) are problem-dependent, according to data preprocessing and to the number of input features and outputs, but \( N2 \) must be defined in the Neural Network implementation. Thus, the maximum number of connections is given by:

\[
N_{max} \equiv N1 + N2 + N3
\]

(3)

Each solution is composed of two vectors: (a) the connectivity vector \( C \), containing a set of bits which represent the network topology; and (b) the connection vector \( W \), containing real numbers which represent the network weights.

\[
s \equiv (C, W)
\]

(4)

\[
C \equiv (c_1, c_2, ..., c_{N_{max}}), c_i \in \{0, 1\}, i = 1, 2, ..., N_{max}
\]

(5)

\[
W \equiv (w_1, w_2, ..., w_{N_{max}}), w_i \in \mathbb{R}, i = 1, 2, ..., N_{max}
\]

(6)

where \( \mathbb{R} \) is the set of real numbers.

Thus, the connection \( i \) is specified by two parameters: a connectivity bit \( (c_i) \), which is equal to 1 if the connection exists in the network, and zero otherwise; and 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 a MLP network with the minimum topology (i.e., $c_1 = 1, i = 1, 2, ..., N_{\text{max}}$), and the initial weights are randomly generated from an uniform distribution in the interval [-1.0, +1.0].

\[ G. \text{ Cost Function} \]

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

\[ \gamma(x) \in \{1, 2, ..., N_C\}, \forall x \in P_t \quad (7) \]

In the experiment the winner-takes-all classification rule was used. For this reason, the number of output units ($N$) is equal to the number of classes ($N_C$).

Being $o_k(x)$ the output value of the output unit $k$ for the pattern $x$, the class assigned to pattern $x$ is defined as:

\[ \phi(x) \equiv \arg \max_k o_k(x), \forall x \in P_t, k \in \{1, 2, ..., N\} \quad (8) \]

The network error for the pattern $x$ is defined as follows:

\[ \varepsilon(x) \equiv \begin{cases} 1, & \phi(x) \neq \gamma(x), \\ 0, & \phi(x) = \gamma(x). \end{cases} \quad (9) \]

Therefore, the classification error for the training set $P_t$, which represents the percentage of incorrectly classified training patterns, can be defined as:

\[ E(P_t) \equiv \frac{100}{\#P_t} \sum_{x \in P_t} \varepsilon(x) \quad (10) \]

where $\#P_t$ is the number of patterns in the set $P_t$.

The percentage of connections used by the network is given by:

\[ \psi(C) \equiv \frac{100}{N_{\text{max}}} \sum_{i=1}^{N_{\text{max}}} c_i \quad (11) \]

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

\[ f(s) \equiv \frac{1}{2} (E(P_t) + \psi(C)) \quad (12) \]

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

\[ f(s) \equiv \frac{1}{2} (\text{SEP}(P_t) + \psi(C)) \quad (13) \]

The SEP error is given by:

\[ \text{SEP} \equiv 100 \frac{o_{\text{max}} - o_{\text{min}}}{N_C \#P_t} \sum_{p=1}^{\#P_t} \sum_{i=1}^{N_c} (\phi(x)_{pi} - \gamma(x)_{pi})^2 \quad (14) \]

where $o_{\text{min}}$ and $o_{\text{max}}$ are the minimum and maximum values of output coefficients in the problem representation (assuming these are the same for all output nodes).

Therefore, the algorithm tries to minimize both network performance and complexity. Only valid networks (i.e., networks with at least one unit in the hidden layer) were considered.

\[ H. \text{ Insertion of new hidden nodes} \]

The constructive process is used to add new hidden nodes in the network topology. In the beginning of the search process the probability of add new nodes is bigger, but to perform a better error surface exploration, the addiction of new nodes need to be controlled. New nodes are added in the network topology according a rule. According to this rule, the new probability of a hidden node to be added is equal to the current multiplied by a factor ($\epsilon$), which is smaller than 1 but close to 1. The initial probability $\lambda_0$ and the factor $\epsilon$ must be defined in the implementation, as well as $I_{\text{max}}$ (number of iterations between two consecutive probability variations) and $I_{\text{max}}$ (maximum number of iterations). Thus, probability of insertion new hidden nodes $\lambda_i$ of iteration $i$ is given by:

\[ \lambda_i \equiv \begin{cases} \epsilon \lambda_{i-1}, & i \neq kI_{\text{max}}, k = 1, 2, ..., \frac{I_{\text{max}}}{I_{\text{max}}} \\ \lambda, & \text{otherwise}. \end{cases} \quad (15) \]

\[ I. \text{ Generation Mechanism for the New Solutions} \]

The initial solution is randomly generated with $N1$ and $N3$ being problem dependent values and $N2 = \mu$, $\mu = 1, 2, ..., N3$. The initial population is defined with a size of 10 chromosomes. To generate a new solution, from the current solution $s = (C, W)$, a new solution $s' = (C', W')$ is generated by genetic micro-evolution of $g_n$ generations. The chromosomes are classified by Rank Based Fitness Scaling [3]. The parents chosen for the next generation is accomplished in a probabilistic manner, using Universal Stochastic Sampling [3]. Elitism was not used and for the combination of the parent chromosomes, the crossover operator Uniform Crossover [20] was used, with a probability of 80%. The crossover operation is performed combining the parts of the parent chromosomes with the same length, as in the below sample. The mutation operator used was the Gaussian Mutation [19], with a probability of 10%.

**Uniform Crossover**

| Parent A | 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 0 |
| Child A  | 1 1 1 0 0 1 0 1 0 1 1 1 1 |
| Child B  | 0 1 0 1 1 1 0 0 1 1 0 0 |

\[ J. \text{ Cooling Schedule and Stopping Criteria} \]

The cooling strategy chosen is the geometric cooling rule. According to this rule, the new temperature is equal to the current multiplied by a temperature factor ($\gamma$), which is smaller than 1 but close to 1. The initial temperature
$T_0$ and the temperature factor $r$ must be defined in the implementation, as well as $I_T$ (number of iterations between two consecutive temperature updates) and $I_{\text{max}}$ (maximum number of iterations). Thus, temperature $T_i$ of iteration $i$ is given by:

$$T_i = \begin{cases} rT_{i-1}, & \text{if } i = kI_T, k = 1, 2, ..., I_{\text{max}}, \\ T_t, & \text{otherwise}. \end{cases} \quad (16)$$

The optimization process stops if: (1) the $GL_5$ criterion defined in Proben1 [15] is met (based on the classification error or SEP of the validation set); or (2) the maximum number of iterations is achieved. For the implementation of the $GL_5$ criterion, the classification error or SEP for the validation set is evaluated at each $I_T$ iterations.

The $GL_5$ criterion is a good approach for avoiding over-fitting to the training set. The classification error for the validation set $P_v$ is given by $E(P_v)$, which is calculated according to Equation 11. In this way, denoting by $V(k)$ the classification error $E(P_v)$ at iteration $i = kI_T$, $k = 1, 2, ..., I_{\text{max}}$ the generalization loss parameter ($GL$) is defined as the relative increase of the validation error over the minimum-so-far (in percent):

$$GL(k) = \left(\frac{V(k)}{\min_{j \leq k} V(j)} - 1\right) \quad (17)$$

The $GL_5$ criterion stops the execution when the parameter $GL$ becomes higher than 10% [15].

### III. Experiments

In experiments were used four classification problems: (1) The odor recognition problem in artificial noses [5]; (2) Diagnose diabetes of Pima indians [4]; (3) Fisher’s Iris data set [2]; (4) Thyroid data set [16]. And one prediction problem: (1) Mackey-Glass time series [10].

#### A. Artificial nose data set

In this problem, the aim is to classify odors from three different vintages (years 1995, 1996 and 1997) of the same wine (Almadn, Brazil). A prototype of an artificial nose was used to acquire the data. This prototype is composed of six distinct polypyrrol based gas sensors, built by electrochemical deposition of polypyrrol using different types of doping agents. Three data acquisitions were performed for each vintage of wine, by recording the resistance value of each sensor at every half second during five minutes. The data set has 6 inputs, 3 outputs and 1,800 examples. More details of the data set and the artificial nose prototype can be found in [5].

#### B. Diabetes data set

The data set contains the diagnose diabetes of Pima indians, based on personal data (e.g. age, number of time pregnant) and the results of medical examinations (e.g. blood pressure, body mass index), try to decide whether a Pima indian individual is biabtes positive or not. The data set has 8 inputs, 2 outputs and 768 examples. There are not absent values, but exist values not representatives. The dataset was obtained from [4].

#### C. Iris data set

Fisher’s Iris data set contains 150 random samples of flowers from the iris species setosa, versicolor, and virginica collected by [2]. From each species there are 50 observations for sepal length, sepal width, petal length, and petal width in centimeters. This dataset was obtained from [4].

#### D. Thyroid data set

The data set contain information related to thyroid dysfunction. The problem is to determine whether a patient has a normally functioning thyroid, an under-functioning thyroid (hypothyroid), or an over-active thyroid (hyperthyroid). There are 7,200 cases in the data set with 3,772 from the year 1985 and 3,428 from 1986. The hyperthyroid class represents 2.3% (166 cases) of the data points, the hypothyroid class accounts for 5.1% (368 cases) of the observations, while the normal group makes up the remaining 92.6% (6,666 cases). This highly unbalanced data set is a notoriously difficult problem for traditional classification methods. For each of the 7,200 cases, there are 21 attributes with 15 binary and 6 continuous variables used to determine in which of the three classes the patient belongs. This dataset was obtained from [4].

#### E. Mackey Glass data set

In experiments the neural network were used to predict points of the time series that result of the Mackey-Glass equation integration [10], given by:

$$\frac{dx}{dt} = -bx(t) + a \frac{x(t-\tau)}{1 + x(t-\tau)} \quad (18)$$

It is a time series with chaotic behavior, recognized as a reference in the study of the learning and generalization capacity of different architectures of neural networks and neuro-fuzzy systems. To obtain the time series value at integer points, fourth order Runge-Kutta method was applied to generate 1,000 data points. The time step used assume the values $x(0) = 1.2, \tau = 17$, and $x(t) = 0$ for $t < 0$. The neural network training was done using 500 data points ($t = 118$ to 618), using 250 data points to validation ($t = 618$ to 868), by giving 4 inputs ($t-18, t-12, t-6$ and $t$) and we attempted to predict the output ($t + 6$). The neural network were tested with another 250 data points ($t = 867$ to 1.118).

In the classification problems the data for training and test the artificial neural network were divided as follows: 50% of the patterns from each class were assigned randomly to the training set, 25% were assigned to the validation set, and 25% were reserved to test the network, as suggested by Proben1 [15]. All networks units implemented the hyperbolic tangent activation function. The patterns were normalized to the range [-1, +1] and the processing units were implemented by hyperbolic tangent activation function.

The performance of the optimization methods are impacted by the choice of its parameters, but there are no rules for
adjusting the configuration to produce better results. Thus, the configuration used in this work, which was chosen after some preliminary experiments, may not have been optimal for the problem. A more rigorous parameter exploration may have generated better results, but this work does not intend to present an exhaustive exploration of the adjustable parameters, which would be very time consuming. This work aims to show that good results have been achieved by simulated annealing for the optimization problem, despite its difficulty for parameter adjustment.

To perform a better comparison among the methods, at the end of the search process, the MLP architecture optimized by all methods is kept constant, and the weights are taken as the initial ones for training with the backpropagation algorithm, with the same training parameters, in order to perform a fine-tuned local search, like performed in GaTSa method.

IV. RESULTS AND DISCUSSION

A. MLP Experiments

Except for GaTSa method, the other investigated optimization techniques need a good initial network topology (maximum topology) to obtain success in the neural network architecture optimization. To define this topology, experiments were accomplished with different architecture topologies in each one of the data sets.

For all data set experiments, and each topology, 10 runs were performed with 30 distinct random weight initializations. Table I presents the Squared Error Percentage (SEP) and the classification error of test set obtained in the training of a full-connected Multi-Layer Perceptron Neural Network by using a gradient descent with momentum backpropagation. The learning rate was set to 0.001, and the momentum term to 0.7.

<table>
<thead>
<tr>
<th>Artificial Nose</th>
<th>Iris</th>
<th>Thyroid</th>
<th>Diabetes</th>
<th>Mackey Glass</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2</td>
<td>Mean test classification error (%)</td>
<td>SEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>33.6296</td>
<td>19.0598</td>
<td>10.2000</td>
<td>-</td>
</tr>
<tr>
<td>03</td>
<td>-</td>
<td>18.2051</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>04</td>
<td>17.8123</td>
<td>7.9487</td>
<td>9.2704</td>
<td>27.8819</td>
</tr>
<tr>
<td>05</td>
<td>-</td>
<td>6.8376</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>06</td>
<td>14.1185</td>
<td>10.6838</td>
<td>-</td>
<td>30.2951</td>
</tr>
<tr>
<td>07</td>
<td>-</td>
<td>8.9744</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>08</td>
<td>11.1136</td>
<td>13.1519</td>
<td>28.4201</td>
<td>1.9045</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>7.3800</td>
<td>27.0833</td>
<td>1.5804</td>
</tr>
<tr>
<td>12</td>
<td>8.8667</td>
<td>-</td>
<td>27.3264</td>
<td>2.3831</td>
</tr>
<tr>
<td>14</td>
<td>11.9704</td>
<td>7.4824</td>
<td>28.4549</td>
<td>2.7860</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>10.2537</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The best performance of Multi-Layer Perceptron in Artificial Nose data set was the topology using 10 hidden units (which contains 90 connections), a mean classification error of 6.31%. In Iris data set the best results was obtained by the topology with 5 hidden units (which contains 32 connections), a mean classification error of 6.84%. In Thyroid data set the smallest mean classification error (7.38%) was obtained by the topology with 10 hidden units (using 240 unit connections). The full connected MLP presented the best performance in Diabetes data set using 10 hidden units (which contains 72 connections), a mean classification error of 27.08%. In the prediction problem, Mackey Glass data set, the best performance of the MLP was found in topology using 4 hidden units (which contains 20 connections), a squared error percentage of 1.43.

B. Optimization Methodologies Experiments

For SA, TS, GA and TSa, the maximal topology in Artificial Nose data set contains six input units, ten hidden units and three output units ($N1 = 6$, $N2 = 10$ and $N3 = 3$, the maximum number of connections ($N_{max}$) is equal to 90). In Iris data set the maximal topology contains $N1 = 4$, $N2 = 5$, $N3 = 3$ and $N_{max} = 32$. For the Thyroid data set the maximal topology contains $N1 = 21$, $N2 = 10$, $N3 = 3$ and $N_{max} = 240$. In Diabetes data set the maximal topology contains $N1 = 8$, $N2 = 10$, $N3 = 2$ and $N_{max} = 100$. In Mackey Glass experiments the maximal topology contains $N1 = 4$, $N2 = 4$, $N3 = 1$ and $N_{max} = 50$. In all Neural Networks topologies $N1$ and $N3$ values are problem dependent and $N2$ was obtained in experiments of previous section. For GaTSa the same values for $N1$ and $N3$ are used, but the value of $N2$ is optimized together the network weights and connections in a constructive way.

In Table II are presented the average performance of each investigated optimization technique. These results were obtained for each technique in the optimization of the number of connections and weight connection values of an MLP artificial neural network. The parameters evaluated were: (1) Squared Error Percentage (SEP) and the classification error (Class) of test set; (2) Mean number of Input processing units; (3) Mean number of hidden processing units; and (4) Percentage of network connections. The following table displays the average results of 10 simulations. Each simulation contains 30 different runs of the algorithms.

For all data sets, the neural networks obtain low classification error when compared to those obtained by MLP networks without topology optimization (Table I), and the mean number of connections is much lower than the maximum number allowed. In all data sets the best optimization performance was obtained by the proposed methodology. For Artificial Nose data set the classification error was around 0.79% (the full connected MLP obtained a classification error of 6.30%) and the mean percentage number of connections 36% of the maximum number allowed. In Iris data set, the best classification error was of 5.26% (6.84% in a full connected MLP) and the mean percentage connections 31.85%. In Thyroid data set the mean classification error was 7.15% (7.38% in a full connected MLP) and the mean percentage connections 31.85%. For Diabetes data set the mean classification error was of 27.06% (the full connected MLP obtained an error of 27.08%) using mean percentage connection of 9.10%. For the prediction problem Mackey Glass data set, the obtained squared error percentage was 0.72 (1.43 in a full connected
TABLE II
OPTIMIZATION TECHNIQUES PERFORMANCE

<table>
<thead>
<tr>
<th>Artificial</th>
<th>SA</th>
<th>TS</th>
<th>GA</th>
<th>TSa</th>
<th>GaTSa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nose</td>
<td>Class. (%)</td>
<td>3.3689</td>
<td>3.2015</td>
<td>3.3689</td>
<td>3.2015</td>
</tr>
<tr>
<td></td>
<td>Connec. (%)</td>
<td>3.3689</td>
<td>3.3689</td>
<td>3.3689</td>
<td>3.3689</td>
</tr>
<tr>
<td>Iris</td>
<td>Class. (%)</td>
<td>12.6496</td>
<td>12.6496</td>
<td>12.6496</td>
<td>12.6496</td>
</tr>
<tr>
<td></td>
<td>Connec. (%)</td>
<td>12.6496</td>
<td>12.6496</td>
<td>12.6496</td>
<td>12.6496</td>
</tr>
<tr>
<td>Thyroid</td>
<td>Class. (%)</td>
<td>3.3813</td>
<td>3.3813</td>
<td>3.3813</td>
<td>3.3813</td>
</tr>
<tr>
<td></td>
<td>Connec. (%)</td>
<td>3.3813</td>
<td>3.3813</td>
<td>3.3813</td>
<td>3.3813</td>
</tr>
<tr>
<td>Diabetes</td>
<td>Class. (%)</td>
<td>27.1509</td>
<td>27.1509</td>
<td>27.1509</td>
<td>27.1509</td>
</tr>
<tr>
<td></td>
<td>Connec. (%)</td>
<td>27.1509</td>
<td>27.1509</td>
<td>27.1509</td>
<td>27.1509</td>
</tr>
<tr>
<td>Mackey</td>
<td>SEP Test</td>
<td>2.0172</td>
<td>2.0172</td>
<td>2.0172</td>
<td>2.0172</td>
</tr>
<tr>
<td>Glass</td>
<td>Input</td>
<td>3.6167</td>
<td>3.6167</td>
<td>3.6167</td>
<td>3.6167</td>
</tr>
<tr>
<td></td>
<td>Hidden</td>
<td>1.9000</td>
<td>1.9000</td>
<td>1.9000</td>
<td>1.9000</td>
</tr>
<tr>
<td></td>
<td>Connec. (%)</td>
<td>19.2600</td>
<td>19.2600</td>
<td>19.2600</td>
<td>19.2600</td>
</tr>
</tbody>
</table>

Between the isolated search techniques, the superiority of the results obtained by TS on SA, in most of the cases, can be explained because at each iteration, TS evaluates 20 new solutions, therefore it has a greater chance to find better solutions in the search space. The SA method evaluates a single new solution at each iteration, thus it needs more iterations to find good solutions. The maximum number of iterations for SA was 1,000, but this quantity may not have been enough for allowing a satisfactory exploration in the search space. In contrast with SA, TS used a maximum of 100 iterations in experiments.

For the proposed methodology, the mean number of connections was lower than all remaining approaches. It can be seen that the method is able to perform a better exploration in the topology search space due to the combination of the advantages of GA, SA and TS, in order to generate MLP networks with small number of connections and high classification performance. In the search process irrelevant connections are eliminated from the network topology in a prunning way. The integration of SA and TS have the same characteristics, but the use of GA operators incorporate more domain specific knowledge in the process search.

All the approaches implemented in this work are able to eliminate input units in MLP topologies. Therefore, it is important to verify which input features are discarded and which ones are more relevant in the neural networks results. In experiments, the proposed methodology performed a better exploration in the architecture search space than the remaining approaches, generating a larger number of topologies, which do not need all these inputs. The inputs with the highest usage frequency have the highest importance in the classification or prediction task. This information can be used in real applications to reduce the database complexity and improve the performance of the classifier. In Figure 1 are presented the mean results of the MLP topology optimization by all investigated techniques in the five data sets.

![Fig. 1. Topology optimization by the optimization techniques.](image)

A paired-differences t test with 95% confidence level [12] was applied in order to confirm the statistical significance of these conclusions. Statistically the GaTSa method gets...
to better optimize the architecture input nodes. In spite of finding topologies with smaller hidden nodes number, all methods were statistically equivalents to optimize these units. However, this is an indication that the constructive strategy, to the definition of hidden nodes number, works. The obtained MLP performance, of the optimized neural networks, was statistically equivalent to the Thyroid, Diabetes and Mackey Glass data sets. The method GaTSA obtained better results in the Artificial Nose data set and the GA in the Iris data set.

V. CONCLUSIONS

This work presented a methodology that integrates the heuristics of tabu search, simulated annealing, genetic algorithms and backpropagation. This method uses concepts of search optimization, pruning and constructive training. In the simultaneous optimization of the connection number and connection values of the Multi-Layer Perceptron neural network, this technique obtained promising results in comparison with the isolated techniques and a hybrid method.

The proposed methodology combines strategies of global and local searches, presenting excellent results regarding the investigated solution space, computational cost and search time. Is important to remember the investigated problem involves a critical subject, the stability versus plasticity relation in the training of artificial neural networks.

Considering the data sets used in this work, the methodology was able to generate automatically MLP topologies with much fewer connections than the maximum number allowed. The results also generate interesting conclusions about the importance of each input feature in the classification and prediction task.

Future investigations need to consider another fitness functions and mechanisms to insert new hidden nodes in the neural network architecture in the search process.

ACKNOWLEDGMENT

The authors would like to thank CNPq, CAPES and FINEP (Brazilian research agencies) for their financial support.

REFERENCES