DCDIS A Supplement, Advances in Neural Networks, Vol. 14(S1) 328--337
Copyright@2007 Watam Press

Global Optimization Methods for Designing and Training Feedforward Artificial Neural Networks

Cleber Zanchettin and Teresa B. Ludermir
Centro de Informática - Federal University of Pernambuco
Recife, PE P. O. Box 7851, 50.732-970, Brazil
{cz,tbl}@cin.ufpe.br

***Abstract*— This paper presents a new method that integrates tabu search, simulated annealing, genetic algorithms and backpropagation in both a pruning and constructive manner. The approach obtained promising results in the simultaneous optimization of an artificial neural network architecture and weights. With the proposed method, we investigate four cost functions for global optimization methods: the average method, weight-decay, multi-objective optimization and combined multi-objective/weight-decay. The experiments are performed in four classifications and one prediction problem.**

I. Introduction

The performance of an artificial neural network depends upon the selection of proper weight connections and network topology during network training. Due to the complex nature of neural network training, even simple functions can have very complex error surfaces. Since the nature of neural network learning algorithms is local convergence, it can be demonstrated that solutions are highly dependent upon the initial random drawing 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 neural network training is the stability versus plasticity relation in the architecture definition. A lack of network connections can render a neural network unable to solve 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 defining the neural network architecture [32]: (1) The empirical approach consists of testing several topologies until finding one that presents satisfactory results; (2) the search optimization approach consists of generating variations of a neural network and combining the best characteristics of this new network in order to improve performance; (3) the pruning approach consists of optimizing a network through the elimination of the elements (network connections) that have no influence over the generalization performance; and (4) the constructive approach, in which the networks begin with a minimum topology and the architecture is built during the training process.

Genetic Algorithms (AG) [13], Simulated Annealing (SA) [21] and Tabu Search (TS) [12] are the most popular of the iterative optimization algorithms. All three optimization heuristics have similarities [37]: (1) They are approximation (heuristic) algorithms, i.e., they do not ensure 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 for 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 search network architecture optimization, this new methodology integrates these three heuristics, combining the advantages in order to overcome the limitations.

This paper presentes a new methodology for optimizing Multi-Layer Perceptron neural networks (MLP) that integrates the main potentialities of these approaches: (1) search optimization for generating new solutions; (2) pruning to eliminate connections and optimize network size; and (3) the constructive approach for finding the best network topology.

Four cost functions for global optimization will be investigated in our evaluation: (1) average method, which is the average of the classification error and the percentage of connections the network uses; (2) weight-decay mechanism, inspired by weight decay backpropagation implementation; (3) multi-objective optimization strategy in genetic operators; and (4) a combination of the weight-decay mechanism and multi-objective optimization strategy. All approaches seek to minimize both network performance and complexity.

In experiments to validate the method, four classification simulations are performed: (1) The odor recognition problem in artificial noses [8]; (2) Diabetes diagnoses in Pima indians [4]; (3) Fisher’s Iris data set [2]; (4) Thyroid dysfunction data set [36]; and one prediction simulation: (1) Mackey-Glass time series [26].

The next section describes the optimization techniques investigated. Section 3 contains some related work. The experiments performed and discussion are presented in Section 4 and Section 5. Section 6 contains the final remarks.

II. Description of Search Heuristics

A. Genetic Algorithms

Genetic algorithms emerged from the analogy between optimization, the genetic mechanisms of species 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 [17].

A genetic algorithm is characterized by a parallel search of the state space rather than a point-by-point search through conventional optimization techniques. The parallel search is accomplished by maintaining a set of possible solutions for the optimization problem, known as a 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 alter the genetic structure of a single chromosome; and (2) higher-order operators, referred to as crossovers, which obtain a new individual by combining genetic material from two selected parent chromosomes [13].

In the experiments performed, each chromosome represents a neural network architecture. The initial population was defined with a size of 10 chromosomes. The chromosomes are classified by Rank Based Fitness Scaling [3]. Parent selection for the next generation is accomplished in a probabilistic manner, using Universal Stochastic Sampling [3]. Elitism was also used, with a probability of 10%. For the combination of the parent chromosomes, the crossover operator Uniform Crossover [41] was used, with a probability of 80%. The mutation operator used was Gaussian Mutation [39], with a probability of 10%. The stop criteria were: (1) the GL_5 criterion defined in *Proben1* [35], (based on the classification error for the validation set); and (2) a maximum number of 200 generations.

B. Simulated Annealing

The simulated annealing method is different from other search methods in that uphill moves are occasionally accepted in order to escape from local minima. The search process consists of a sequence of iterations. Each iteration consists of randomly changing the current solution to create a new solution in its neighborhood. Once a new solution is created, the corresponding change in the cost function is computed to decide if the new solution can be accepted. If the new solution cost is lower than the current solution cost, the new solution is accepted. Otherwise, the Metropolis criterion is verified [29], based on the Boltzmann probability. A random number d in a $[0, 1]$ interval is generated from a uniform distribution. If $\delta \leq e^{\frac{\Delta E}{T}}$, where ΔE is the change in the cost function and T is a parameter called temperature, then the new solution is accepted as the current solution. If not, the current solution is unchanged and the process continues from the current solution.

The algorithm was originally derived from thermodynamic simulations. Thus, the parameter T is referenced as temperature and the temperature reduction process is called the cooling process. The cooling strategy chosen was *geometric cooling rule*. According to this rule, the new temperature is equal to the current temperature multiplied by a temperature factor (smaller than, but close to one) [33]. The initial temperature is set at 1 and the temperature factor is set at 0.9. The temperature is decreased every 10 iterations, with a maximum number of 1000 iterations. The GL_5 stop criterion was also used.

C. Tabu Search

Tabu search is an iterative search algorithm characterized by the use of a flexible memory. In this method, each iteration consists of the evaluation of a certain amount of new solutions (neighborhood moves). The best of these solutions (in terms of cost function) is accepted. However, the best candidate solution may not improve the current solution. Thus, the algorithm chooses a new solution that either produces the greatest 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 on the tabu list are marked as forbidden for subsequent iterations. The tabu list registers T last visited solutions. When the list is full, a new movement is registered, replacing the older movement kept on the list.

In the present work, a neighborhood with 20 solutions was used and the algorithm chose the best non-tabu solution. The proximity criterion [38] was used to compare two solutions. A new solution is considered identical to the tabu solution 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 GL_5 stop criterion was also employed.

III. Related Work

A number of approaches in the literature have used the integration of tabu search, simulated annealing and genetic algorithms for specific applications. An integration of the three heuristics was proposed by Mantawy et al. [27], Liu et al. [25] and Jiang et al. [18]. In Li et al. [23], genetic algorithms and simulated annealing were combined for engineering the optimization of planning processes.

In artificial neural network training, simulated annealing, tabu search and genetic algorithms were used separately in several applications [15], [34], [38]. In most approaches, the aim was to minimize the local convergence problem of the backpropagation algorithm. Among the three approaches, genetic algorithms are the most widely used in artificial neural network optimization. In a number of works, there are simultaneously optimized configuration parameters and network architecture [15]. Other authors use genetic algorithms to simultaneously optimize parameters and initial values for weight connections among processing units [20]. Garca-Pedrajas et al. [10] proposed new crossover operators for neural network training using genetic algorithms.

Lacerda [22] studied the application of genetic algorithms for based radial artificial neural networks. Tsai et al. [42] used a hybrid algorithm to feedforward artificial neural network architecture and parameter design. Palmes et al. [31] use a mutation-based algorithm to train MLP nodes. Gepperth and Roth [11] used an evolutionary multi-objetive process to optimize feedforward architectures. Works using simulated annealing and tabu search for artificial neural network optimization are scarcer. Some applications using simulated annealing and backpropagation were implemented for the training of a MLP with a fixed topology containing two hidden layers. The problem addressed was the recognition of sonar responses [34].

The simulated annealing method was successfully used in some global optimization problems, as can be seen in Corana et al. [7] and Sexton et al. [39]. In Stepniewski and Keane [40], simulated annealing and genetic algorithms were used to optimize architectures of MLP neural networks. Similar experiments were performed by Sexton et al. [39], but the candidate solutions were represented by vectors of real numbers containing all of the network weights. The same experiments were performed by Sexton et al. [38] applying the tabu search algorithm. Metin et al. [28] used simulated annealing to optimize artificial neural network architectures applied to expert diagnosis systems. In Hamm [14] simulated annealing was used to optimize artificial neural networks weights. Karaboga and Kalinli [19] proposed a parallelism-based tabu search model to train recurrent artificial neural networks with the aim of identifying dynamic systems. Cannas et al. [5] used tabu search to optimize artificial neural network architectures for time series prediction.

The integration of tabu search, simulated annealing and backpropagation (TSa) was proposed by Yamazaki [1] for MLP artificial neural network architecture and weight optimization. Lins [24] proposed a few modifications to the Yamazaki model in order to increase the search space and modify the cooling schedule process. Thus, the three approaches are normally used to adjust connection values among processing units in fixed topologies.

A. Integration of Simulated Annealing, Tabu Search and Genetic Algorithms in a Constructive Manner

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 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 of better quality.

These observations motivated the proposal of an optimization methodology that combines the main advantages 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 has a minimum valid architecture size. A group of new solutions is generated at each iteration, 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. Previously visited solutions are marked as tabu, as in a tabu search. In the course of the search, the chromosome size is increased in a constructive manner in order to find the 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.

Algorithm 1 - Pseudo-code of the proposed algorithm

```
1.  $P_0 \leftarrow$ initial population with $K$ solutions $s_k$
.         and size $s_z$
2.  $T_0 \leftarrow$ initial temperature
3.  Update $S_{BSF}$ with $s_k$ of the $P_0$ (best solution
.   found so far)
4.  For $i = 0$ to $I_{max} - 1$
5.       If $i + 1$ is not a multiple of $I_T$
6.           $T_{i+1} \leftarrow T_i$
7.       Else
8.           $T_{i+1} \leftarrow$ new temperature
9.           Increase the size of the population $P_i$
10.          $P_i \leftarrow P_z$
10.          If validation based stopping criteria
.               are not satisfied
11.                 Stop global search execution
12.      For $j = 0$ to $g_n$
13.          Generate a new population $P'$ from $P_i$
14.          $P_i \leftarrow P'$
15.      Choose the best solution $s_k$ from $P_i$
16.      If $f(s') < f(s_k)$
17.          $s_{k+1} \leftarrow s'$
18.      Else
19.          $s_{k+1} \leftarrow s'$ with probability $e^{\frac{f(s')-f(s_k)}{T_{i+1}}}$
20.          If $f(s_{k+1}) < f(S_{BSF})$
21.               Update $S_{BSF}$
22. Keep the topology contained in $S_{BSF}$ constant
.   and use the weights as initial ones for
.   training with the backpropagation algorithm
```

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 iteration. 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, creates and eliminates network connections, like a pruning process. The initial solution is coded with the minimum valid network topology and new hidden nodes are added following the constructive process. It is of interest 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.

B. 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 the data preprocessing and the number of input and outputs features, but N2 must be defined in the Neural Network implementation. Thus, the maximum number of connections is given by:

$$N_{max} \equiv N2(N1 + N3) \tag{1}$$

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

$$s \equiv (C, W) \tag{2}$$

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

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

where $\Re$ 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., $c_i = 1, i = 1, 2, ..., N_{max}$) and the initial weights are randomly generated from a uniform distribution in the interval [-1.0, +1.0].

C. 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:

$$\gamma(x) \in \{1, 2, ..., N_C\}, \forall x \in P_t \tag{5}$$

In the experiment, the *winner-takes-all* classification rule was used in which the number of output units ($N3$) is equal to the number of classes (N_C).

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

$$\phi(x) = arg \max o_k(x), \forall x \in P_t, k \in \{1, 2, ..., N_3\} \tag{6}$$

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

$$\varepsilon(x) = \begin{cases} 1, & if \quad \phi(x) \neq \gamma(x), \\ 0, & if \quad \phi(x) = \gamma(x). \end{cases} \tag{7}$$

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

$$E(P_t) = \frac{100}{\#P_t} \sum_{x_e P_t} \varepsilon(x) \tag{8}$$

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

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

$$\psi(C) = \frac{100}{N_{max}} \sum_{i=1}^{N_{max}} c_i \tag{9}$$

In experiments, four ways of performing cost evaluation were evaluated: (1) mean of the classification error and the percentage of connections; (2) weight decay mechanism inspired by weight decay backpropagation implementation; (3) multi-objective optimization strategy in the genetic operators; and (4) a combination of the weight decay mechanism and multi-objective optimization strategy. The algorithms in all approaches seek to minimize both network performance and complexity. Only valid networks (i.e., networks with at least one unit in the hidden layer) were considered.

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

$$f(s) = \frac{1}{2}(E(P_t) + \psi(C)) \tag{10}$$

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:

$$f(s) = \frac{1}{2}(SEP(P_t) + \psi(C)) \tag{11}$$

The SEP error is given by:

$$SEP = 100 \frac{o_{max} - o_{min}}{N_c \#P_t} \sum_{p=1}^{\#P_t} \sum_{i=1}^{N_c} (\phi(x)_{pi} - \gamma(x)_{pi})^2 \tag{12}$$

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

2) Weight decay: Weight decay was initially suggested as an implementation to improve the Backpropagation algorithm (BP) for the preference bias of a robust neural network that is insensitive to noise [16], [43]. The weight decay mechanism performs in a network architecture differentially toward zeroes by reinforcing large weights and weakening small weights 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 BP algorithm uses the gradient descent method to drive the learning process. The objective is to minimize the network error function:

$$f(s) = \frac{1}{2}\sum_{i}(T_i - O_i)^2 \tag{13}$$

where T_i is the target value and O_i is the output from the network. The method is implemented by adding a bias term B to the original cost function Er_0:

$$f(s) = f(s)_0 + \frac{1}{2}\mu B \tag{14}$$

where μ is the parameter for the importance of the bias term B. The bias term can represent different results regarding the network weights. A bias choice is presented in equation 15. In this equation, it is possible to decay small weights more rapidly than large weights.

$$B = \sum_{ij} W_{ij}^2/(1 + W_{ij}^2) \tag{15}$$

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

The weight decay mechanism is used in the GaTSa cost function to eliminate solutions with small weights that can be used by the neural network to code noise patterns. The GaTSa cost function is presented in equation 16.

$$f(s) = \frac{1}{2}\sum_{i} E(P_t) + \frac{1}{2}\psi(C) + \frac{1}{2}\mu\sum_{ij} W_{ij}^2/(1+W_{ij}^2) \tag{16}$$

3) Multi-objective optimization: Multi-objective optimization is the search for simultaneously minimizing the n components f_k, $k = 1, ..., n$, of a vector function f of a variable x in a universe u, where,

$$f(x) = (f_1(x), ..., f_n(x)) \tag{17}$$

The problem usually has no unique, global solution, but has a set of equally efficient or non-inferior alternative solutions, known as the Pareto-optimal set [9]. Pareto-optimal solutions consist of all solutions for which the corresponding objective cannot be improved in any dimension without degradation in another. Mathematically, the concept of Pareto optimality is: considering no loss of generality, a minimization problem and two solutions a, $b \in X$. Thus, $a \prec b$ (a to *dominate* b) if:

$$\begin{aligned} &\forall i \in \{1, 2, ..., n\} : f_i(a) \leq f_i(b) \wedge \\ &\quad \exists j \in \{1, 2, ..., n\} : f_i(a) < f_i(b) \end{aligned} \tag{18}$$

In the present work, the multi-objective strategy is used in genetic operators to evolve the population of solutions. As opposed to the objective problem, the ranking of a population in the multi-objetive case is not unique. This is due to concepts such as dominance and preferability, which define partial rather than total orders. In experiments, ranking is obtained in the same way as in Fonseca and Fleming (1998). Thus, the rank of the individual x_i at generation g is the number of individuals in the current population that are preferable to it. This ensures that all preferred individuals in the current population are assigned a zero rank. The best individual is the individual with the best rank. In the case of multiple individuals with the same rank, the best individual between them is the individual with the lesser classification error. In the population, the normalized rank $r^{(g)}/N$ constitutes an estimate of the fraction of the search space preferable to each individual considered.

D. Insertion of new hidden nodes

The constructive process is used to add new hidden nodes in the topology. The search process starts with the probability of adding new, larger nodes, but in order to perform a better error surface exploration, the addition of new nodes is controlled. New nodes are added in the topology following a rule. According to this rule, the new probability of a hidden node being added is equal to the current node multiplied by a factor (ϵ), which is smaller than, but close to 1. The initial probability λ_ϵ and the factor ϵ must be defined in the implementation, as well as I_λ (number of iterations between two consecutive probability variations) and I_{max} (maximum number of iterations). Thus, the probability of the insertion of new hidden nodes λ_i at iteration i is given by:

$$\lambda_i \equiv \begin{cases} \epsilon\lambda_{i-1}, & if \quad i = kI_\lambda, k = 1, 2, ..., \frac{I_{max}}{I_\lambda}, \\ \lambda_i, & otherwise. \end{cases} \tag{19}$$

E. 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. From the current solution $s = (C, W)$, the new solution $s' = (C', W')$ is generated by the genetic micro-evolution of g_n generations. The chromosomes are classified by Rank Based Fitness Scaling [3]. Parent selection for the next generation is accomplished in a probabilistic manner, using Universal Stochastic Sampling [3]. Elitism was not used and the crossover operator Uniform Crossover [41] was used for the combination of the parent chromosomes, with a probability of 80%. The crossover operation is performed by combining the parts of the parent chromosomes that have the same length, as in the sample below. The mutation operator used was the Gaussian Mutation [39], with a probability of 10%.

$$\begin{array}{ll}
 & \overbrace{\hphantom{1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1}}^{Uniform\ Crossover} \\
\text{Parent A} & 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\
\text{Parent B} & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
 & \\
\text{Child A} & 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1 \\
\text{Child B} & 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0
\end{array}$$

F. Cooling Schedule and Stopping Criteria

The cooling strategy selected is the geometric cooling rule. According to this rule, the new temperature is equal to the current temperature multiplied by a temperature factor (r), which is smaller than, 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) e I_{max} (maximum number of iterations). Thus, temperature T_i at iteration i is given by:

$$T_i \equiv \begin{cases} rT_{i-1}, & if \quad i = kI_T, k = 1, 2, ..., \frac{I_{max}}{I_T}, \\ T_i, & otherwise. \end{cases} \tag{20}$$

The optimization process stops if: (1) the GL_5 criterion defined in Proben1 [35] 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 iterations.

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 $E(P_v)$, which is calculated according to Equation 8. Thus, denoting by $V(k)$ the classification error $E(P_v)$ at iteration $i = kI_T$, $k = 1, 2, ..., \frac{I_{max}}{I_T}$, 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%.

$$GL(k) \equiv \left(\frac{V(k)}{\min_{j \le k} V(j)} - 1 \right) \tag{21}$$

IV. Experiments

Four classification problems were used in experiments: (1) The odor recognition problem in artificial noses [8]; (2) Diabetes diagnoses in Pima indians [4]; (3) Fisher's Iris data set [2]; (4) Thyroid dysfunction data set [36]; aAnd one prediction problem: (1) Mackey-Glass time series [26].

A. Artificial nose data set

In this problem, the aim is to classify odors from three different vintages (1995, 1996 and 1997) of the same wine (Almadn, Brazil). A prototype of an artificial nose was used to acquire the data. The data set has 6 inputs, 3 outputs and 1800 examples. Further details on the data set and the artificial nose prototype can be found in [8].

B. Diabetes data set

This data set contains diabetes diagnoses in Pima indians based on personal data and results from medical examinations. The data set has 8 inputs, 2 outputs and 768 examples. There are no absent values, but there are non-representative values. The data set 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]. For each species, there are 50 observations regarding sepal length, sepal width, petal length and petal width in centimeters. This dataset was obtained from [4].

D. Thyroid data set

This data set contains 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 7200 cases in the data set, with 21 attributes used to determine to which of the three classes the patient belongs. This dataset was obtained from [4].

E. Mackey Glass data set

In experiments, a neural network was used to predict points of the time series that result from the Mackey-Glass equation integration [26], given by:

$$\frac{dx}{dt} \equiv -bx(t) + a\frac{x(t-\tau)}{1 + x^{10}(t-\tau)} \tag{22}$$

This 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, the fourth order Runge-Kutta method was applied to generate 1000 data points. The time step used assumes the values $x(0) = 1.2$, $\tau = 17$ and $x(t) = 0$ for $t < 0$. The neural network training was performed with 500 data points ($t = 118$ to 618), using 250 data points for 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 testing 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 [35]. All network units implemented the hyperbolic tangent activation function. The patterns were normalized to the range [-1, +1].

In order 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 using the same training parameters in order to perform a fine-tuned local search, as performed in the GaTSa method.

V. Results and Discussion

A. MLP Experiments

Apart from the GaTSa method, the other investigated optimization techniques require a good initial network topology (maximum topology) to obtain success in neural network architecture optimization. To define this topology, experiments were performed with different architecture topologies on each one of the data sets. For all data set experiments in 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 the test set obtained in the training of a fully connected MLP Neural Network by using a gradient descent with momentum backpropagation. The learning rate was set at 0.001 and the momentum term at 0.7.

TABLE I
Results for MLP Neural Networks

	Artificial Nose	Iris	Thyroid	Diabetes	Mackey Glass
N2	Mean test classification error (%)				SEP
02	33.6296	19.0598	10.2000	-	4.2146
03	-	18.2051	-	-	-
04	17.8123	7.9487	9.2704	27.8819	1.4357
05	-	6.8376	-	-	-
06	14.1185	10.6838	-	30.2951	1.8273
07	-	8.9744	-	-	-
08	11.1136	-	13.1519	28.4201	1.9045
10	6.3086	-	7.3800	27.0833	1.5804
12	8.8667	-	7.3804	27.3264	2.3831
14	11.9704	-	7.4824	28.4549	2.7860
16	-	-	10.2537	-	-

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

B. Optimization Methodologies Experiments

1) GaTSa Fixed Architecture Experiments: Table II 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 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 the training, validation and test sets; (2) algorithm iteration number; (3) artificial neural network connection number; and (4) the temperature value. The following table displays the average results of 10 simulations. Each simulation contained 30 different runs of the algorithm.

The technique that combines the heuristics of tabu search, simulated annealing and genetic algorithms obtained the best performance. This technique was better without using the local search heuristic to optimize the artificial neural network connection values. The average classification error was 2.87%, with an average of 8 connections from 36 possible connections in a fully connected neural network. Using a fully connected network, the local optimization technique backpropagation obtained an average error of 6.78%.

Figure 1 presentes graphs comparing the performance of the investigated techniques. The proposed technique obtained the best results regarding the classification error, final network connection number and the number of iterations needed for architecture optimization.

TABLE II
Optimization technique performance

Technique	Training		Validation		Test		Iterat.	Connec.
	SEP	Class	SEP	Class	SEP	Class		
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

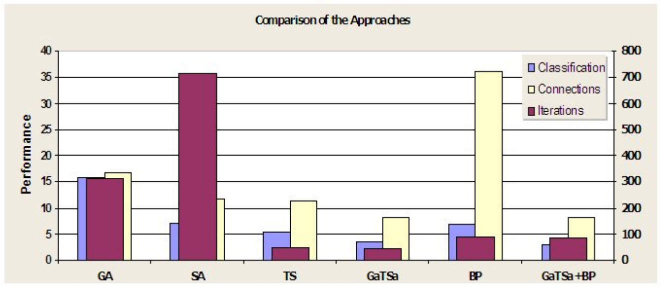

Fig. 1. Topology optimization of the optimization techniques.

2) GaTSa Variable Architecture Experiments: For SA, TS, GA and TSa, the maximal topology in the 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 the 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 the Diabetes data set, the maximal topology contains $N1 = 8$, $N2 = 10$, $N3 = 2$ and $N_{max} = 100$. In the Mackey-Glass experiments, the maximal topology contains $N1 = 4$, $N2 = 4$, $N3 = 1$ and $N_{max} = 50$. In all Neural Network topologies, $N1$ and $N3$ values are problem-dependent and $N2$ was obtained in experiments from the previous section. For GaTSa, the same values for $N1$ and $N3$ are used, but the value of $N2$ is optimized, together with the network weights and connections, in a constructive manner.

TABLE III
OPTIMIZATION TECHNIQUE PERFORMANCE

		SA	TS	GA	TSa	GaTSa
Artificial Nose	Class. (%)	3.3689	3.2015	13.3884	1.4244	0.7914
	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
	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
	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
	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
Mackey Glass	SEP Test	2.0172	0.8670	0.6542	0.6847	0.72164
	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

Table III displays the average performance of each optimization technique investigated. 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 the test set; (2) Mean number of input processing units; (3) Mean number of hidden processing units; and (4) Percentage of network connections. Table III displays the average results of 10 simulations. Each simulation contains 30 different runs of the algorithms.

For all data sets, the neural networks obtain a lower classification error than 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 the Artificial Nose data set, the classification error was around 0.79% (the fully connected MLP obtained a classification error of 6.30%) and the mean percentage number of connections obtained 36% of the maximum number allowed. In the Iris data set, the best classification error was of 5.26% (6.84% in a fully connected MLP) and the mean percentage of connections was 31.85%. In the Thyroid data set, the mean classification error was 7.15% (7.38% in a fully connected MLP) and the mean percentage of connections was 12.64% of a fully connected network. For the Diabetes data set, the mean classification error was 27.06% (the fully connected MLP obtained an error of 27.08%) and the mean percentage of connections was 9.10%. For the prediction problem of the Mackey-Glass data set, the obtained squared error percentage was 0.72 (1.43 in a fully connected neural network), using an 11.49% mean percentage of connections in a fully connected MLP.

Genetic algorithms, tabu search and simulated annealing methods incorporate domain specific knowledge in their search heuristics. They also tolerate some elements of non-determinism, which helps the search escape from local minima. They rely on the use of a suitable cost function that provides feedback to the algorithm as the search progresses. The proposed integration combines these advantages in order to use a larger amount of information in the problem domain and apply this information to practically all search phases. The initial solution is coded with a minimum valid network topology and hidden nodes are inserted in the network topology during algorithm execution. This process is similar to constructive neural network training and allows better topology selection. Moreover, the proposed methodology has two well-defined stages: a global search phase, which makes use of the capacity for generating new solutions from the genetic algorithms, the cooling process and cost function of the simulated annealing as well as the memory characteristics of the tabu search technique; and a local search phase, which makes use of characteristics such as gradient descending for a more precise solution adjustment. These characteristics can obtain better solutions to the problems investigated, with a short search time, low computacional cost and minimal investigated search space.

The better search space exploration of the proposed methodology can be verified in the experiment analysis. The proposed methodology generated solutions with a low-complexity topology. In order to find suitable solutions, the remaining search techniques require the definition of a good initial neural network topology. They then eliminate network connections through pruning. The proposed methodology is able to optimize the network size in the search automatically.

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 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 a small number of connections and high classification performance. In the search process, irrelevant connections are eliminated from the network topology through pruning. The integration of SA and TS have the same characteristics, but the use of GA operators incorporate more domain-specific knowledge in the search process.

All the approaches implemented in the present work are able to eliminate input units in MLP topologies. Therefore, it is important to verify which input features are discarded and which are more relevant in the neural network 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 greatest 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. Figure 2 displays the mean results of the MLP topology optimization in the

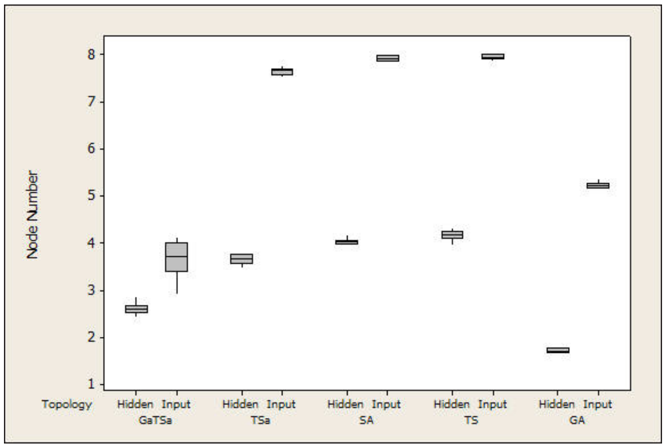

Fig. 2. Topology optimization of the optimization techniques.

five data sets by all the techniques investigated.

A paired-difference t test with a 95% confidence level [30] was applied in order to confirm the statistical significance of these conclusions. Statistically, the GaTSa method achieves better optimization of the architecture input nodes. Despite its finding topologies 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 neural networks was statistically equivalent for the Thyroid, Diabetes and Mackey Glass data sets. The GaTSa method obtained better results in the Artificial Nose data set, whereas GA had the best performance in the Iris data set.

3) The Cost Function Influence: Table IV displays the average results from 10 simulations. Each simulation contains 30 different runs of the algorithms. In this table, the cost functions evaluated are: the Average Method (Average), Weight-decay (WD), Multi-objective (MO) and Multi-objective plus Weight-decay (MO+WD). The parameters evaluated were: (1) Squared Error Percentage (SEP) and the classification error (Class) of the test set; (2) Mean number of input processing units; (3) Mean number of hidden processing units; and (4) Percentage of network connections.

In the Artificial Nose data set, the best classification results were obtained by the multi-objective approach and the best architecture optimization was found by the weight-decay method. In the Iris data set, the combination of weight-decay and genetic operators using multi-objective optimization presented the best performance. The weight-decay cost function presented the best optimization performance in the Thyroid, Diabetes and Mackey-Glass data sets.

The better performance of weight-decay demonstrates the capacity of this method for 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 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.

TABLE IV

Experiment Results

		Average	WD	MO	MO+WD
Artificial Nose	Class. (%)	11.8595	7.5462	7.0407	12.6237
	Input	5.9967	5.9233	5.9967	5.9867
	Hidden	9.1533	7.5600	8.4833	8.6200
	Connec.	50.2400	33.2433	49.6500	42.0467
Iris	Class. (%)	6.1197	6.9316	4.2735	3.9829
	Input	3.5667	3.8167	3.8567	3.1467
	Hidden	3.7967	4.4367	3.5500	3.0767
	Connec.	13.3867	18.1167	15.8600	9.8000
Thyroid	Class. (%)	7.1024	6.8196	6.9270	6.8609
	Input	20.6733	20.7800	20.9800	20.9767
	Hidden	7.1833	7.8067	8.5233	8.5167
	Connec.	83.8800	91.6700	114.6667	115.0933
Diabetes	Class. (%)	28.4583	25.7552	28.2639	25.8542
	Input	7.7367	7.7767	7.9700	7.9667
	Hidden	5.6100	5.2000	4.4133	6.3100
	Connec.	31.8433	31.7433	43.2700	42.0367
Mackey Glass	SEP Test	0.6215	0.2721	0.5745	0.6293
	Input	1.1067	1.0533	2.0933	1.7867
	Hidden	1.0000	1.0067	1.0100	1.1800
	Connec.	2.1533	2.0833	3.6300	3.3233

The use of multi-objective optimization in genetic operators presented interesting results in some data sets, but exhibited poor performance in most. The main problem with this approach is the construction of the Pareto ranking. There is no efficient algorithm for checking non-dominance in a set of feasible solutions. Traditional algorithms have serious performance degradation as the size of the population and the number of objectives increases [6]. The multi-objective approach presented some outlier results in experiments. These values hampered performance, and such problems also hampered the performance of the combination of weight-decay and multi-objective optimization. A paired-difference t test with a 95% confidence level [30] was applied in order to confirm the statistical significance of these conclusions.

The best problem search space exploration was with the use of the weight-decay cost function in the experiment analysis. This method generated solutions with low-complexity topology and a low number of errors. The superiority of the method was statistically verified in the Mackey-Glass, Diabetes and Artificial Nose data sets.

VI. Conclusions

The present work introduced a methodology that integrates the heuristics of tabu search, simulated annealing, genetic algorithms and backpropagation. This method uses concepts from search optimization, pruning and constructive training. The proposed methodology combines strategies of global and local searches, presenting excellent results regarding the investigated solution space, computacional cost and search time. It is important to remember that the problem investigated involves a critical subject, the stability versus plasticity relation in the training of artificial neural networks.

In the technique proposed, four cost functions were evaluated: the average of the classification error and the percentage of connections used by the network; the weight-decay mechanism; multi-objective optimization strategy; and the combination of the weight-decay mechanism and multi-objective optimization strategy. The best performance was obtained by the weight-decay approach in the benchmarks investigated.

In some of the literature, the multi-objective approach clearly outperforms a pure random search strategy that randomly generates new points in the search space without exploiting similarities between solutions. The use of this approach in the proposed methodology needs a more accurate investigation in future work.

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

Future investigations should consider other fitness functions and mechanisms for inserting new hidden nodes in the neural network architecture during the search process.

REFERENCES

[1] T. B. Ludermir A. Yamazaki, M. C. P. de Souto. Optimization of neural network weights and architectures for odor recognition using simulated annealing. In *Proc. Int. Joint Conf. on Neural Networks*, pages 547–552, 2002.

[2] E. Anderson. The irises of the gasp peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1953.

[3] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proc. of Int. Conf. on Genetic Algorithms and their application*, pages 14–21. Lawrence Erlbaum Associates, 1987.

[4] C. L. Blake and C. J. Merz. Uci repository of machine learning databases. Technical report, University of California, Irvine, CA, 1998.

[5] B. Cannas, A. Fanni, M. Marchesi, and F. Pilo. A tabu search algorithm for optimal sizing of locally recurrent neural networks. In *Proc. of Int. Symp. on Theoretical Electrical Engineering*, pages 267–272, 1999.

[6] C. A. C. Coello. An empirical study of evolutionary techniques for multiobjective optimization in engineering design. Ph.d. tesis, Tulane University, 1996.

[7] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Trans. on Math. Soft.*, 13:262–280, 1987.

[8] J. E. G. de Souza, B. B. Neto, F. L. dos Santos, M. S. Santos C. P. de Melo, and T. B. Ludermir. Polypyrrole based aroma sensor. *Synthetic Metals*, 102:1296–1299, 1999.

[9] C. M. Fonseca and Fleming P. J. Multi-objetive optimization and multiple constraint handling with evolutionary algorithms - part i: A unified formulation. *IEEE Trans. on Sistems, Man and Cybernetics - Part A*, 28(1):26–37, 1998.

[10] N. Garca-Pedrajas, D. Ortiz-Boyer, and C. Hervs-Martnez. An alternative approach for neural network evolution with a genetic algorithm: Crossover by combinatorial optimization. *Neural Networks*, 19:514528, 2006.

[11] S. Gepperth, A. e Roth. Applications of multi-objective structure optimization. *Neurocomputing*, 69:701713, 2006.

[12] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operation Research*, 13:533–549, 1986.

[13] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publ. Inc., 1989.

[14] L. Hamm, B.W. Brorsen, and M. T. Hagan. Global optimization of neural network weights. In *Proc. of Int. Joint Conf. on Neural Networks*, volume 2, pages 1228–1233, 2002.

[15] S. A. Harp, T. Samad, and A. Guha. Towards the genetic synthesis of neural networks. In *Proc. of Int. Conf. on Genetic Algorithms and Their Applications*, pages 360–369, 1989.

[16] G. E. Hinton. Learning distributed representations of concepts. In *Annual Conf. of Cognitive Science Society*, pages 1–12, 1986.

[17] J. H. Holland. *Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems.* IEEE Computer Society Press, 1986.

[18] T. Jiang, X. Li, and F. Kruggel. Global optimization approaches to meg source localization. In *Proc. of IEEE Int. Symp. on Bio-Informatics and Biomedical Engineering*, pages 223–230, 2000.

[19] D. Karaboga and A. Kalinli. Training recurrent neural networks for dynamic system identification using parallel tabu search algorithm. In *Proc. of IEEE Int. Symp. Intel. Control*, pages 1325–1330, 1997.

[20] P. Khn. Genetic encoding strategies for neural networks. *Knowledge-Based Systems*, 2:947–950, 1996.

[21] S. Kirkpatrick, C. D. Gellat Jr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[22] E. G. M. Lacerda, A. Carvalho, and T. B. Ludermir. Evolutionary optmization of rbf networks. In *Proc. of Brazilian Symp. on Neural Networks*, pages 219–224, 2000.

[23] W. D. Li, S. K. Ong, and A. Y. C. Nee. Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts. *Int. Jour. of Prod. Research*, 40(8):1899–1922, 2002.

[24] A.P.S. Lins and T. B. Ludermir. Hybrid optimization algorithm for the definition of mlp neural network architectures and weights. In *Proc. of Int. Conf. on Hybrid Intelligent Systems*, pages 149–154, 2005.

[25] Y. Liu, L. Ma, and J. Zhang. Ga/sa/ts hybrid algorithms for reactive power optimization. In *Proc. of IEEE Power Engineering Society Summer Meeting*, page 245249, 2000.

[26] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287–289, 1977.

[27] A. H. Mantawy, Y. L. Abdel-Magid, and S. Z. Selim. Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem. *IEEE Trans. Power Sys.*, 14(3):829–836, 1999.

[28] B. Metin N.G. adn Sahiner, H. Chan, L. Hadjiiski, and N. Petrick. Optimal selection of neural network architecture for cad using simulated annealing. In *Proc. of EMBS Int. Conf.*, pages 1325–1330, 2000.

[29] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. In *Journal of Chemical Physics*, volume 21, pages 1087–1092, 1953.

[30] T. M. Mitchell. *Machine Learning.*, chapter Evaluating hypotheses, pages 128–153. WCB/McGraw-Hill, 1997.

[31] P. P. Palmes, T. Hayasaka, and S. Usui. Mutation-based genetic neural network. *IEEE Trans. on Neural Networks*, 16(3):587–600, 2005.

[32] R. Parekh, J. Yang, and V. Honavar. Constructive neural network learning algorithms for multi-category real-valued pattern classification. Technical Report 21, Iowa State University, 1997.

[33] D. T. Pham and D. Karaboga. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer-Verlag New York, 1998.

[34] V. W. Porto, D. B. Fogel, and L. J. Fogel. Alternative neural network training methods. *IEEE Expert*, 10(3):16–22, 1995.

[35] L. Prehelt. Proben1 – a set of neural network benchmark problems and benchmarking rules. Technical Report 21, Karlsruhe, 1994.

[36] J. Quinlan. Simplifying decision trees. *Int. Journal of Man-Machine Studies*, 27:221234, 1987.

[37] S. M. Sait and H. Youssef. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*. IEEE Computer Society Press, 1999.

[38] R J. Sexton, B. Alidaee, R. E. Dorsey, and J.D Johnson. Global optimization for artificial neural networks: A tabu search application. In *European Journal of Operational Research*, volume 106, pages 570–584, 1998.

[39] R. S. Sexton, R. E. Dorsey, and J. D. Johnson. Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing. *European Journal of Operational Research*, 114:574–601, 1999.

[40] S. W. Stepniewski and A. J. Keane. Pruning back-propagation neural networks using modern stochastic optimization techniques. *Neural Computing and Applications*, 5:76–98, 1997.

[41] G. Sywerda. Uniform crossover in genetic algorithm. In *Proc. of Int. Conf. on Gen. Algo.*, pages 2–9. Morgan Kaufmann Pub. Inc., 1989.

[42] J. Tsai, J. Chou, and T. Liu. Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. *IEEE Trans. on Neural Networks*, 17(1):69–80, 2006.

[43] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Generalization by weight elimination with application to forecasting. In *Adv. in Neural Inf. Processing Systems III*, pages 875–882. San Mateo: Morgan Kaufmann, 1991.