A Memetic Pareto Evolutionary Approach to Artificial Neural Networks

H.A. Abbass

University of New South Wales, School of Computer Science, ADFA Campus, Canberra, ACT 2600, Australia, h.abbass@adfa.edu.au.

Abstract. *Evolutionary Artificial Neural Networks* (EANN) have been a focus of research in the areas of *Evolutionary Algorithms* (EA) and *Artificial Neural Networks* (ANN) for the last decade. In this paper, we present an EANN approach based on pareto multi-objective optimization and differential evolution augmented with local search. We call the approach *Memetic Pareto Artificial Neural Networks* (MPANN). We show empirically that MPANN is capable to overcome the slow training of traditional EANN with equivalent or better generalization.

Keywords: neural networks, genetic algorithms

1 Introduction

Evolutionary Artificial Neural Networks (EANNs) have been a key research area for the last decade. On the one hand, methods and techniques have been developed to find better approaches for evolving *Artificial Neural Networks* and more precisely - for the sake of our paper – Multi–layer feed–forward *Artificial Neural Networks* (ANNs). On the other hand, finding a good ANNs' architecture has been an issue as well in the field of ANNs. Methods for network growing (such as Cascade Correlation [4]) and for network pruning (such as Optimal Brain Damage [14]) have been used to overcome the long process for determining a good network architecture. However, all these methods still suffer from their slow convergence and long training time. In addition, they are based on gradient–based techniques and therefore can easily stuck in a local minimum. EANNs provide a better platform for optimizing both the network performance and architecture simultaneously. Unfortunately, all of the research undertaken in the EANN literature ignores the fact that there is always a trade–off between the architecture and the generalization ability of the network. A network with more hidden units may perform better on the training set, but may not generalize well on the test set. This trade–off is a well known problem in *Optimization* known as the Multi-objective Optimization Problem (MOP).

With the trade–off between the network architecture – taken in this paper to be the number of hidden units – and the generalization error, the EANN problem is in effect a MOP. It is, therefore, natural to raise the question of why not applying a multi–objective approach to EANN.

The objective of this paper is to present a *Memetic* (*ie. evolutionary algorithms* (EAs) augmented with local search [18]) Pareto Artificial Neural Networks (MPANN). The rest of the paper is organized as follows: In Section 2, background materials are covered followed by an explanation of the methods in Section 3. Results are discussed in Section 4 and conclusions are drawn in Section 5.

2 Background Materials

In this section, we introduce necessary background materials for Multi-objective Optimization, ANNs, *Differential Evolution* (DEs), Evolutionary Multi-objective, and EANN.

2.1 Multi-objective Optimization

Consider a *Multi-Objective Optimization Problem* (MOP) model as presented below:-

$$\begin{gathered}\text{Optimize } F(\boldsymbol{x}) \\ \text{subject to: } \Omega = \{\boldsymbol{x} \in R^n | G(\boldsymbol{x}) \leq 0\}\end{gathered}$$

Where $\boldsymbol{x}$ is a vector of decision variables $(x_1, \ldots, x_n)$ and $F(\boldsymbol{x})$ is a vector of objective functions $(f_1(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x}))$. Here $f_1(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})$, are functions on R^n and Ω is a nonempty set in R^n. The vector $G(\boldsymbol{x})$ represents a set of constraints.

In MOPs, the aim is to find the optimal solution $\boldsymbol{x}^* \in \Omega$ which optimize $F(\boldsymbol{x})$. Each objective function, $f_i(\boldsymbol{x})$, is either maximization or minimization. Without any loss of generality, we assume that all objectives are to be minimized for clarity purposes. We may note that any maximization objective can be transformed to a minimization one by multiplying the former by -1.

To define the concept of non-dominated solutions in MOPs, we need to define two operators, $\not\geqq$ and $\precnsim$ and then assume two vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$. We define the first operator as $\boldsymbol{x} \not\geqq \boldsymbol{y}$ iff $\exists\ x_i \in \boldsymbol{x}$ and $y_i \in \boldsymbol{y}$ such that $x_i \neq y_i$. And, $\boldsymbol{x} \precnsim \boldsymbol{y}$ iff $\forall\ x_i \in \boldsymbol{x}$ and $y_i \in \boldsymbol{y}, x_i \leq y_i$, and $\boldsymbol{x} \not\geqq \boldsymbol{y}$. The operators $\not\geqq$ and $\precnsim$ can be seen as the "not equal to" and "less than or equal to" operators respectively, between two vectors. We can now define the concepts of local and global optimality in MOPs.

Definition 1: Neighborhood or open ball The open ball (*ie.* a neighborhood centered on $\boldsymbol{x}^*$ and defined by the Euclidean distance) $B_\delta(\boldsymbol{x}^*) = \{\boldsymbol{x} \in R^n |\ ||\boldsymbol{x} - \boldsymbol{x}^*|| < \delta\}$.

Definition 2: Local efficient (non-inferior/ pareto-optimal) solution A vector $\boldsymbol{x}^* \in \Omega$ is said to be a local efficient solution of MOP iff $\nexists\ \boldsymbol{x} \in (B_\delta(\boldsymbol{x}^*) \cap \Omega)$ such that $F(\boldsymbol{x}) \precnsim F(\boldsymbol{x}^*)$ for some positive δ.

Definition 3: Global efficient (non-inferior/ pareto-optimal) solution A vector $\boldsymbol{x}^* \in \Omega$ is said to be a global efficient solution of MOP iff $\nexists\ \boldsymbol{x} \in \Omega$ such that $F(\boldsymbol{x}) \precnsim F(\boldsymbol{x}^*)$.

Definition 4: Local non-dominated solution A vector $\boldsymbol{y}^* \in F(\boldsymbol{x})$ is said to be local non-dominated solution of MOP iff its projection onto the decision space, $\boldsymbol{x}^*$, is a local efficient solution of MOP.

Definition 5: Global non-dominated solution A vector $\boldsymbol{y}^* \in F(\boldsymbol{x})$ is said to be global non-dominated solution of MOP iff its projection onto the decision space, $\boldsymbol{x}^*$, is a global efficient solution of MOP.

In this paper, the term "non-dominated solution" is used as a shortcut for the term "local non-dominated solution".

2.2 Artificial Neural Networks

We may define an ANN by a graph: $G(N, A, \psi)$, where N is a set of neurons (also called nodes), A denotes the connections (also called arcs or synapses) between the neurons, and ψ represents the learning rule whereby neurons are able to adjust the strengths of their interconnections. A neuron receives its inputs (also called activation) from an external source or from other neurons in the network. It then undertakes some processing on this input and sends the result as an output. The underlying function of a neuron is called the activation function. The activation, a, is calculated as a weighted sum of the inputs to the node in addition to a constant value called the bias. The bias can be easily augmented to the input set and considered as a constant input. From herein, the following notations will be used for a single hidden layer MLP:

- I and H are the number of input and hidden units respectively.
- $\mathbf{X}^p \in \mathbf{X} = (x_1^p, x_2^p, \ldots, x_I^p), p = 1, \ldots P$, is the pth pattern in the input feature space $\mathbf{X}$ of dimension I, and P is the total number of patterns.
- Without any loss of generality, $\mathbf{Y}_o^p \in \mathbf{Y_o}$ is the corresponding scalar of pattern $\mathbf{X}^p$ in the hypothesis space $\mathbf{Y_o}$.
- w_{ih} and w_{ho}, are the weights connecting input unit i, $i = 1 \ldots I$, to hidden unit h, $h = 1 \ldots H$, and hidden unit h to the output unit o (where o is assumed to be 1 in this paper) respectively.
- $\Theta_h(\mathbf{X}^p) = \sigma(a_h)$; $a_h = \sum_{i=0}^{I} w_{ih} x_i^p$, $h = 1 \ldots H$, is the h^{th} hidden unit's output corresponding to the input pattern $\mathbf{X}^p$, where a_h is the activation of hidden unit h, and $\sigma(.)$ is the activation function that is taken in this paper to be the logistic function $\sigma(z) = \frac{1}{1+e^{-Dz}}$, with D the function's sharpness or steepness and is taken to be 1 unless it is mentioned otherwise.
- $\hat{Y}_o^p = \sigma(a_o)$; $a_o = \sum_{h=0}^{H} w_{ho} \Theta_h(\mathbf{X}^p)$ is the network output and a_o is the activation of output unit o corresponding to the input pattern $\mathbf{X}^p$.

MLPs are in essence non-parametric regression methods which approximate underlying functionality in data by minimizing a risk function. The data are presented to the network and the risk function is approximated empirically R_{emp} by summing over all data instances as follows:

$$R_{emp}(\alpha) = \sum_{p=1}^{P} (Y_o^p - \hat{Y}_o^p)^2 \tag{1}$$

The *Back-propagation algorithm* (BP), developed initially by Werbos [25] and then independently by Rumelhart group [21], is commonly used for training the network. BP uses the gradient of the empirical risk function to alter the parameter set α until the empirical risk is minimum. BP in its simple form uses a single parameter, η representing the learning rate. For a complete description for the derivations of this algorithm, see for example [8]. The algorithm can be described in the following steps:-

1. Until termination conditions are satisfied, do
   (a) for each input-output pairs, $(\mathbf{X}^p, Y_o^p)$, in the training set, apply the following steps
      i. Inject the input pattern $\mathbf{X}^p$ into the network
      ii. Calculate the output, $\Theta_h(\mathbf{X}^p)$, for each hidden unit h.
      iii. Calculate the output, $\hat{Y_o}^p$, for each output unit o.
      iv. for the output unit o, calculate $r_o = \hat{Y}_o^p(1-\hat{Y}_o^p)(Y_o^p - \hat{Y}_o^p)$ where r_o is the rate of change in the error of the output unit o.
      v. for each hidden unit h, $r_h = \Theta_h^p(1-\Theta_h^p)w_{ho}r_o$ where r_h is the rate of change in the error of hidden unit h.
      vi. update each weight in the network using the learning rate η as follows:

$$w_{ih} \leftarrow w_{ih} + \Delta w_{ih}, \qquad \Delta w_{ih} = \eta r_j a_{ih} \tag{2}$$

$$w_{ho} \leftarrow w_{ho} + \Delta w_{ho}, \qquad \Delta w_{ho} = \eta r_k a_{ho} \tag{3}$$

2.3 Differential Evolution

Evolutionary algorithms [5] is a kind of global optimization techniques that use selection and recombination as their primary operators to tackle optimization problems. *Differential evolution* (DE) is a branch of evolutionary algorithms developed by Rainer Storn and Kenneth Price [24] for optimization problems over continuous domains. In DE, each variable is represented in the chromosome by a real number. The approach works as follows:-

1. Create an initial population of potential solutions at random, where it is guaranteed, by some repair rules, that variables' values are within their boundaries.
2. Until termination conditions are satisfied
   (a) Select at random a trail individual for replacement, an individual as the main parent, and two individuals as supporting parents.
   (b) With some probability, called the *crossover probability*, each variable in the main parent is perturbed by adding to it a ratio, F, of the difference between the two values of this variable in the other two supporting parents. At least one variable must be changed. This process represents the crossover operator in DE.
   (c) If the resultant vector is better than the trial solution, it replaces it; otherwise the trial solution is retained in the population.
   (d) go to 2 above.

2.4 Evolutionary Multi-objective

EAs for MOPs [3] can be categorized into one of three categories: plain aggregating, population–based non–Pareto and Pareto–based approaches. The plain aggregating approach combines all the objectives into one using linear combination (such as in the weighted sum method, goal programming, and goal attainment). Therefore, each run results in a single solution and many runs are needed to generate the pareto frontier. In addition, the quantification of the importance of each objective (*eg.* by setting numerical weights) is needed, which is very difficult for most practical situations. Meanwhile, optimizing all the objectives simultaneously and generating a set of alternative solutions as in population-based approaches, offers more flexibility.

There has been a number of methods in the literature for population–based non–pareto [23] and pareto [9, 32, 13] approaches to MOPs. More recently, we developed the *Pareto Differential Evolution* (PDE) method using *Differential Evolution* (DE) for MOPs [1]. The PDE method outperformed all previous methods on five benchmark problems.

2.5 Evolutionary Artificial Neural Networks

Over the last two decades, research into EANN has witnessed a flourish period [28, 27]. Yao [29] presents a thorough review to the field with over 300 references just in the area of EANN. This may indicate that there is an extensive need for finding better ways to evolve ANN.

A major advantage to the evolutionary approach over traditional learning algorithms such as *Back-propagation* (BP) is the ability to escape a local optima. More advantages include robustness and ability to adopt in a changing environment. In the literature, research into EANN has been taking one of three approaches; evolving the weights of the network, evolving the architecture, or evolving both simultaneously.

The EANN approach uses either binary representation to evolve the weight matrix [10, 11] or real [6, 7, 16, 19]. There is not an obvious advantage of binary encoding in EANN over the real. However, with real encoding, there are more advantages including compact and natural representation.

The key problem (other than being trapped in a local minimum) with BP and other traditional training algorithms is the choice of a correct architecture (number of hidden nodes and connections). This problem has been tackled by the evolutionary approach in many studies [12, 15, 20, 30, 31]. In some of these studies, weights and architectures were evolved simultaneously.

The major disadvantage to the EANN approach is it is computationally expensive, as the evolutionary approach is normally slow. To overcome the slow convergence of the evolutionary approach to ANN, hybrid techniques were used to speed up the convergence by augmenting evolutionary algorithms with a local search technique (*ie.* memetic approach), such as BP [26].

3 The MPANN Algorithm

3.1 Representation

In deciding on an appropriate representation, we tried to choose a representation that can be used for other architectures without further modifications. Our chromosome is a class that contains one matrix Ω and one vector ρ. The matrix Ω is of dimension $(I + O) \times (H + O)$. Each element $\omega_{ij} \in \Omega$, is the weight connecting unit i with unit j, where $i = 0, \ldots, (I - 1)$ is the input unit i; $i = I, \ldots, (I + O - 1)$ is the output unit $(i - I)$; $j = 0, \ldots, (H - 1)$ is the hidden unit j; and $j = H, \ldots, (H + O - 1)$ is the output unit $(j - H)$. This representation has the following two characteristics that we are not using in the current version but can easily be incorporated in the algorithm for future work:-

1. It allows direct connection from each input to each output units (we allow more than a single output unit in our representation).
2. It allows recurrent connections between the output units and themselves.

The vector ρ is of dimension H, where $\rho_h \in \rho$ is a binary value used to indicate if hidden unit h exists in the network or not; that is, it works as a switch to turn a hidden unit on or off. The sum, $\sum_{h=0}^{H} \rho_h$, represents the actual number of hidden units in a network, where H is the maximum number of hidden units. This representation allows simultaneous training of the weights in the network and selecting a subset of hidden units.

3.2 Methods

As the name indicates in our proposed method, we have a multi–objective problem with two objectives; one is to minimize the error and the other is to minimize the number of hidden units. The pareto–frontier of the tradeoff between the two objectives will have a set of networks with different number of hidden units (note the definition of pareto-optimal solutions). However, sometimes the algorithm will return two pareto-networks with the same number of hidden units. This will only take place when the actual number of pareto-optimal solutions in the population is less than 3. Because of the condition in DE of having at least 3 parents in each generation, if there are less than three parents, the pareto optimal solutions are removed from the population and the population is re-evaluated. For example, assume that we have only 1 pareto optimal solution in the population. In this case, we need another 2. The process simply starts by removing the pareto optimal solution from the population and finding the pareto optimal solutions in the remainder of the population. Those solutions dominating the rest of the population are added to the pareto list until the number of pareto solutions in the list is 3.

Our proposed method augments the original PDE [1, 22] algorithm with local search (*ie.* BP) to form the memetic approach. In initial investigations, the algorithm was quite slow and the use of local search improved its performance. MPANN consists of the following steps:

1. Create a random initial population of potential solutions. The elements of the weight matrix Ω are assigned random values according to a Gaussian distribution $N(0,1)$. The elements of the binary vector ρ are assigned the value 1 with probability 0.5 based on a randomly generated number according to a uniform distribution between $[0,1]$; otherwise 0.
2. Repeat
   (a) Evaluate the individuals in the population and label those who are non-dominated.
   (b) If the number of non-dominated individuals is less than 3 repeat the following until the number of non-dominated individuals is greater than or equal to 3:-
      i. Find a non-dominated solution among those who are not labelled.
      ii. Label the solution as non-dominated.
   (c) Delete all dominated solutions from the population.
   (d) Mark 20% of the training set as a validation set for BP.
   (e) Repeat
      i. Select at random an individual as the main parent α_1, and two individuals, α_2, α_3 as supporting parents.
      ii. With some crossover probability $Uniform(0,1)$, do

$$\omega_{ih}^{child} \leftarrow \omega_{ih}^{\alpha_1} + Gaussian(0,1)(\omega_{ih}^{\alpha_2} - \omega_{ih}^{\alpha_3}) \tag{4}$$

$$\rho_h^{child} \leftarrow \begin{cases} 1 \; if(\rho_h^{\alpha_1} + Gaussian(0,1)(\rho_h^{\alpha_2} - \rho_h^{\alpha_3})) \geq 0.5 \\ 0 \; otherwise \end{cases} \tag{5}$$

      otherwise

$$\omega_{ih}^{child} \leftarrow \omega_{ih}^{\alpha_1} \tag{6}$$

$$\rho_h^{child} \leftarrow \rho_h^{\alpha_1} \tag{7}$$

      and with some crossover probability $Uniform(0,1)$, do

$$\omega_{ho}^{child} \leftarrow \omega_{ho}^{\alpha_1} + Gaussian(0,1)(\omega_{ho}^{\alpha_2} - \omega_{ho}^{\alpha_3}) \tag{8}$$

      otherwise

$$\omega_{ho}^{child} \leftarrow \omega_{ho}^{\alpha_1} \tag{9}$$

      where each weight in the main parent is perturbed by adding to it a ratio, $F \in Gaussian(0,1)$, of the difference between the two values of this variable in the two supporting parents. At least one variable must be changed.
      iii. Apply BP to the child.
      iv. If the child dominates the main parent, place it into the population.
   (f) Until the population size is M
3. Until termination conditions are satisfied, go to 2 above.

One may note that before each generation starts, 20% of the instances in the training set are marked as a validation set for the use of BP; that is, BP will use 80% of the original training set for training and 20% for validation. Also, the termination condition in our experiments is the maximum number of epochs is reached; where one epoch is equivalent to one pass through the training set. Therefore, one iteration of BP is equivalent to one epoch since 80% of the training set is used for training and the other 20% for validation; that is, one complete pass through the original training set. After the network is trained, the chromosome changes to reflect the new weight sets.

4 Experiments

4.1 Data Sets

We have tested MPANN on two benchmark data sets; the Australian credit card assessment problem and the diabetes problem. Both data sets are available by anonymous ftp from ice.uci.edu [2]. The following is a brief description of each data set.

- **The Australian Credit Card Assessment Data Set**
  This data set contains 690 patterns with 14 attributes; 6 of them are numeric and 8 discrete (with 2 to 14 possible values). The predicted class is binary - 1 for awarding the credit and 0 for not. The problem is to assess applications for credit cards [17].
- **The Diabetes Data Set**
  This data set has 768 patterns; 500 belonging to the first class and 268 to the second. It contains 8 attributes. The objective is to test if a patient has a diabetes or not. The classification problem is difficult as the class value is a binarized form of another attribute that is highly indicative of a certain type of diabetes without having a one-to-one correspondence with the medical condition of being diabetic [17].

4.2 Experimental Setup

To be consistent with the literature [17], the Australian credit card assessment data set is divided into 10 folds and the Diabetes data set into 12 folds where class distribution is maintained in each fold. One–leave–out cross–validation is used where we run the algorithm with 9 (11) out of the 10 (12) folds for each data set then we test with the remaining one. We vary the crossover probability between 0 to 1 with an increment of 0.1. The maximum number of epochs is set to 2000, the population size 25, the learning rate for BP 0.003, the maximum number of hidden units is set to 10, and the number of epochs for BP is set to 5.

4.3 Results

The average of the pareto networks with the best generalization and the corresponding number of hidden units in each fold are being calculated along with the standard deviations as shown in Table 1. It is interesting to see the small standard deviations for the test error in both data sets, which indicates consistency and stability of the method.

Table 1. The average and standard deviations of the pareto network with the best generalization (smallest test error) in each run

.

Data set	Error	Number of hidden units
Australian Credit Card	0.136 ± 0.045	5.000 ± 1.943
Diabetes	0.251 ± 0.062	6.6 ± 1.505

In Figure 1, the average test and training errors corresponding to the best generalized network in each fold is plotted against each of the eleventh crossover probabilities. In Figure 1 (left), with crossover 0.1 and upward, the test error is always smaller than the training error, which indicates better generalization. However, the degree of this generalization varied across the different crossover probabilities. The best performance occurs with crossover probability 0.3, which indicates that 30% of the weights, on the average, in each parent change. This is quite important as it entails that the building blocks in MPANN is effective; otherwise a better performance would have occurred with the maximum crossover probability. We may note here that crossover in DE is in effect a guided mutation operator. In Figure 1 (right), it is also apparent that an average crossover probability of 0.8 resulted in the best generalization ability. Very high or low crossover probabilities are not as good.

In summary, the best performances for the Australian credit card and Diabetes data sets are 0.136 ± 0.045 and 0.251 ± 0.062 respectively and occur with crossover probabilities 0.3 and 0.8 respectively.

4.4 Comparisons and Discussions

We compare our results against 23 algorithms tested by Michie *et al.* [17]. These algorithms can be categorized into decision trees (CART, IndCART, NewID, AC^2, Baytree, Cal5, and C4.5), rule–based methods (CN2, and ITrule), neural networks (Backprob, Kohonen, LVQ, RBF, and DIPOL92), and statistical algorithms (Discrim, Quadisc, Logdisc, SMART, ALLOC80, k-NN, CASTLE, NaiveBay, and Default). For a complete description of these algorithms, the reader may refer to [17].

In Tables 2 and 3, we find that MPANN is equivalent or better than BP and comparable to the others. However, we notice here that MPANN also optimized

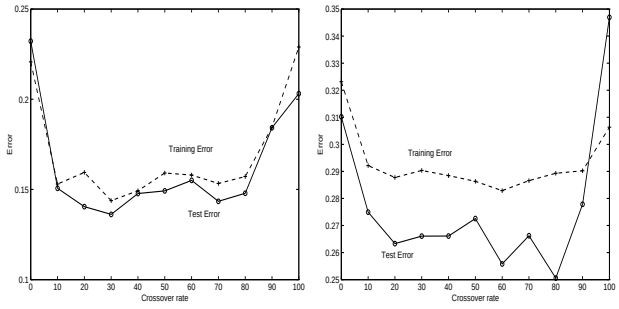

Fig. 1. *The average training and test error for the Australian Credit Card (on the left) and Diabetes data sets (on the right) obtained by each crossover probability.*

Table 2. Comparing MPANN against 23 traditional methods in terms of the average generalization error for the Australian Credit Card data set.

Algorithm	Error Rate	Algorithm	Error Rate	Algorithm	Error Rate	Algorithm	Error Rate
MPANN	0.136	CASTLE	0.148	NaiveBay	0.151	Default	0.440
CART	0.145	IndCART	0.152	NewID	0.181	AC^2	0.181
Baytree	0.171	Cal5	0.131	C4.5	0.155	CN2	0.204
ITrule	0.137	Backprob	0.154	Kohonen	Fail	LVQ	0.197
RBF	0.145	DIPOL92	0.141	Discrim	0.141	Quadisc	0.207
Logdisc	0.141	SMART	0.158	ALLOC80	0.201	k-NN	0.181

its architecture while optimizing its generalization ability. Therefore, in terms of the amount of computations, it is by far faster than BP as we simultaneously optimize the architecture and generalization error. In addition, the total number of epochs used is small compared to the corresponding number of epochs needed by BP.

5 Conclusion

In this paper, we presented a new evolutionary multi–objective approach to artificial neural networks. We showed empirically that the proposed approach outperformed traditional Back-propagation and had comparable results to 23 classification algorithms. For future work, we will evaluate the performance of the proposed method on regression problems and test the scalability of the evolutionary approach.

6 Acknowledgement

The author would like to thank Xin Yao, Bob Mckay, and Ruhul Sarker for their insightful comments while discussing an initial idea with them. This work

Table 3. Comparing MPANN against 23 traditional methods in terms of the average generalization error for the Diabetes data set.

Algorithm	Error Rate	Algorithm	Error Rate	Algorithm	Error Rate	Algorithm	Error Rate
MPANN	0.251	CASTLE	0.258	NaiveBay	0.262	Default	0.350
CART	0.255	IndCART	0.271	NewID	0.289	AC^2	0.276
Baytree	0.271	Cal5	0.250	C4.5	0.270	CN2	0.289
ITrule	0.245	Backprob	0.248	Kohonen	0.273	LVQ	0.272
RBF	0.243	DIPOL92	0.224	Discrim	0.225	Quadisc	0.262
Logdisc	0.223	SMART	0.232	ALLOC80	0.301	k-NN	0.324

is supported with ADFA Special Research Grants TERM6 2001 DOD02 ZCOM Z2844.

References

1. H.A. Abbass, R. Sarker, and C. Newton. A pareto differential evolution approach to vector optimisation problems. *Congress on Evolutionary Computation*, 2:971–978, 2001.
2. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/mlrepository.html. *University of California, Irvine, Dept. of Information and Computer Sciences*, 1998.
3. C.A. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999.
4. S. Fahlman and C. Lebiere. The cascade correlation learning architecture. Technical Report CMU-CW-90-100, Canegie Mellon University, Pittsburgh, PA, 1990.
5. D.B. Fogel. *Evolutionary Computation: towards a new philosophy of machine intelligence*. IEEE Press, New York, NY, 1995.
6. D.B. Fogel, E.C. Wasson, and E.M. Boughton. Evolving neural networks for detecting breast cancer. *Cancer letters*, 96(1):49–53, 1995.
7. D.B. Fogel, E.C. Wasson, and V.W. Porto. A step toward computer-assisted mammography using evolutionary programming and neural networks. *Cancer letters*, 119(1):93, 1997.
8. S. Haykin. *Neural networks - a comprehensive foundation*. Printice Hall, USA, 2 edition, 1999.
9. J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation*, 1:82–87, 1994.
10. D.J. Janson and J.F. Frenzel. Application of genetic algorithms to the training of higher order neural networks. *Systems Engineering*, 2:272–276, 1992.
11. D.J. Janson and J.F. Frenzel. Training product unit neural networks with genetic algorithms. *IEEE Expert*, 8(5):26–33, 1993.
12. H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4(4):461–476, 1990.
13. J. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

14. Y. LeCun, J.J. Denker, and S.A. Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1990.
15. V. Maniezzo. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks*, 5(1):39–53, 1994.
16. F. Menczer and D. Parisi. Evidence of hyperplanes in the genetic learning of neural networks. *Biological Cybernetics*, 66:283–289, 1992.
17. D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine learning, neural and statistical classification*. Ellis Horwood, 1994.
18. P. Moscato. Memetic algorithms: a short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New ideas in optimization*, pages 219–234. McGraw-Hill, 1999.
19. V.W. Porto, D.B. Fogel, and L.J. Fogel. Alternative neural network training methods. *IEEE Expert*, 10(3):16–22, 1995.
20. J.C.F. Pujol and R. Poli. Evolving the topology and the weights of neural networks using a dual representation. *Applied Intelligence*, 8(1):73–84, 1998.
21. D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In J.L. McClelland D.E. Rumelhart and the PDP Research Group Eds, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition., Foundations, 1, 318,*. MIT Press Cambridge, 1986.
22. R. Sarker, H.A. Abbass, and C. Newton. Solving multiobjective optimization problems using evolutionary algorithm. *The International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA'2001), Los Vegas, USA*, 2001.
23. J.D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100, 1985.
24. R. Storn and K. Price. Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, 1995.
25. P. Werbos. *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, 1974.
26. W. Yan, Z. Zhu, and R. Hu. Hybrid genetic/bp algorithm and its application for radar target classification. *Proceedings of the 1997 IEEE National Aerospace and Electronics Conference, NAECON*, pages 981–984, 1997.
27. X. Yao. Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4(5):203–222, 1993.
28. X. Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 8(4):529–567, 1993.
29. X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.
30. X. Yao and Y. Liu. Making use of population information in evolutionary artificial neural networks. *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28(3):417–425, 1998.
31. X. Yao and Y. Liu. Towards designing artificial neural networks by evolution. *Applied Mathematics and Computation*, 91(1):83–90, 1998.
32. E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.