PERGAMON

AVAILABLE AT
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Neural Networks 16 (2003) 507–517

Neural Networks

www.elsevier.com/locate/neunet

2003 Special Issue

Evolving neural networks to identify bent-double galaxies in the FIRST survey

Erick Cantú-Paz*, Chandrika Kamath

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 7000 East Avenue, L-561 Livermore, CA 94551, USA

Abstract

The FIRST (Faint Images of the Radio Sky at Twenty-cm) survey is an ambitious project scheduled to cover 10,000 square degrees of the northern and southern galactic caps. Until recently, astronomers associated with FIRST identified radio-emitting galaxies with a bent-double morphology through a visual inspection of images. Besides being subjective, prone to error and tedious, this manual approach is becoming increasingly infeasible: upon completion, FIRST will include almost a million galaxies. This paper describes the application of six methods of evolving neural networks (NNs) with genetic algorithms (GAs) to the identification of bent-double galaxies. The objective is to demonstrate that GAs can successfully address some common problems in the application of NNs to classification problems, such as training the networks, choosing appropriate network topologies, and selecting relevant features. We measured the overall accuracy of the networks using the arithmetic and geometric means of the accuracies on bent and non-bent galaxies. Most of the combinations of GAs and NNs perform equally well on our data, but using GAs to select feature subsets produces the best results, reaching accuracies of 90% using the arithmetic mean and 87% with the geometric mean. The networks found by the GAs were more accurate than hand-designed networks and decision trees.

Published by Elsevier Science Ltd.

Keywords: Astronomical surveys; Radio-emitting galaxies; Network design; Genetic algorithms; Feature extraction; Feature selection; Data mining

1. Introduction

The Faint Images of the Radio Sky at Twenty-cm (FIRST) survey (Becker, White, & Helfand, 1995) started in 1993 with the goal of producing the radio equivalent of the Palomar Observatory Sky Survey. Using the Very Large Array (VLA) at the National Radio Astronomy Observatory, FIRST is scheduled to cover more than 10,000 square degrees of the northern and southern galactic caps, to a flux density limit of 1.0 mJy (milli-Jansky). At present, with the data from the 1993 through 2000 observations, FIRST has covered about 8,000 square degrees, producing more than 32,000 images, each with two-million pixels. At a threshold of 1 mJy, there are approximately 90 radio-emitting galaxies, or radio sources, in a typical square degree.

Radio sources exhibit a wide range of morphological types that provide clues to the source class, emission mechanism, and properties of the surrounding medium. Sources with a bent-double morphology are of particular interest as they indicate the presence of clusters of galaxies, a key project within the FIRST survey. FIRST scientists currently identify the bent-double galaxies by visual inspection, which–besides being subjective, prone to error and tedious–is becoming increasingly infeasible as the survey grows in size.

Our goal is to bring automation to the classification of galaxies using techniques from data mining, such as neural networks (NNs). NNs have been used successfully to classify objects in many astronomical applications (Adams & Woolley, 1994; Odewahn & Nielsen, 1994; Odewahn, Stockwell, Pennington, Humphreys, & Zumach, 1992; Storrie-Lombardi, Lahav, Sodre, & Storrie-Lombardi, 1992). However, the success of NNs largely depends on their architecture, their training algorithm, and the choice of features used in training. Unfortunately, determining the architecture of a neural network is a trial-and-error process; the learning algorithms must be carefully tuned to the data; and the relevance of features to the classification problem may not be known a priori. Our objective is to demonstrate that genetic algorithms (GAs) can successfully address the topology selection, training, and feature selection problems, resulting in accurate networks with good generalization abilities. This paper describes the application of six

* Corresponding author. Tel.: +1-925-424-2467; fax: +1-925-423-2993.
E-mail addresses: cantupaz@llnl.gov (E. Cantú-Paz), kamath2@llnl.gov (C. Kamath).

0893-6080/03/$ - see front matter Published by Elsevier Science Ltd.
doi:10.1016/S0893-6080(03)00020-0

combinations of GAs and NNs to the identification of bent-double galaxies.

This study is one of a handful that compares different methods of evolving neural nets on the same domain (Grönross, 1998; Roberts & Turenga, 1995; Siddiqi & Lucas, 1998). In contrast with other studies that limit their scope to two or three methods, we use six combinations of GAs and NNs and two measures of classification accuracy to compare the results of the evolved networks against hand-designed networks. Most of the methods we tried performed equally well on our data, but using GAs to select feature subsets yielded the best results. The experiments also suggest that most of the GA and NN combinations produce significantly more accurate classifiers than hand-designed networks and decision trees.

This paper is organized as follows: Section 2 outlines the problem of detecting bent-double radio-emitting galaxies in the FIRST data, and provides details on the approach we have taken to derive meaningful features from the data available. Section 3 describes several combinations of GAs and NNs that have appeared previously in the literature. Section 4 presents our experiments and reports the results. The paper concludes with our observations and plans for future work.

2. Identification of bent-double galaxies

Fig. 1 includes several examples of radio sources from the FIRST survey. While some bent-double galaxies are relatively simple in shape (examples (a) and (b)), others, such as the ones in examples (e) and (f), can be rather complex. The task of automating the detection of bent-doubles can be quite difficult as illustrated by the similarity between the bent-double in example (a) and the non-bent-double in example (c).

Raw and postprocessed data from FIRST are available on the FIRST web site (sundog.stsci.edu). There are two forms of data available for use: image maps and a catalog. Fig. 2 shows an image map containing examples of two bent-doubles. These large image maps are mostly composed of background noise. They are obtained by processing the raw data collected by the VLA telescopes. Each image map covers an area of approximately 0.45 square degrees, with pixels that are 1.8 arc seconds wide.

In addition to the image maps, the FIRST survey also provides a source catalog (White, Becker, Helfand, & Gregg, 1997). The catalog is obtained by fitting two-dimensional elliptic Gaussians to each radio source on an image map. For example, the lower bent-double in Fig. 2 is approximated by more than seven Gaussians while the upper one is approximated by three Gaussians. Each entry in the catalog corresponds to a single Gaussian. The catalog entries include information such as the right ascension (RA, analogous to longitude) and declination (Dec, analogous to latitude) for the center of the Gaussian, the major and minor axes of the ellipse, the peak flux, and the position angle of the major axis (degrees counterclockwise from North). Note that we differentiate between catalog entries and radio sources, with a radio source being composed of one or more catalog entries. The results in this paper are based on the 2000 version of the catalog, which includes data from 1993 to 2000.

We decided that, initially, we would identify the radio sources and extract the features using only the catalog. The astronomers expected that the catalog was a good

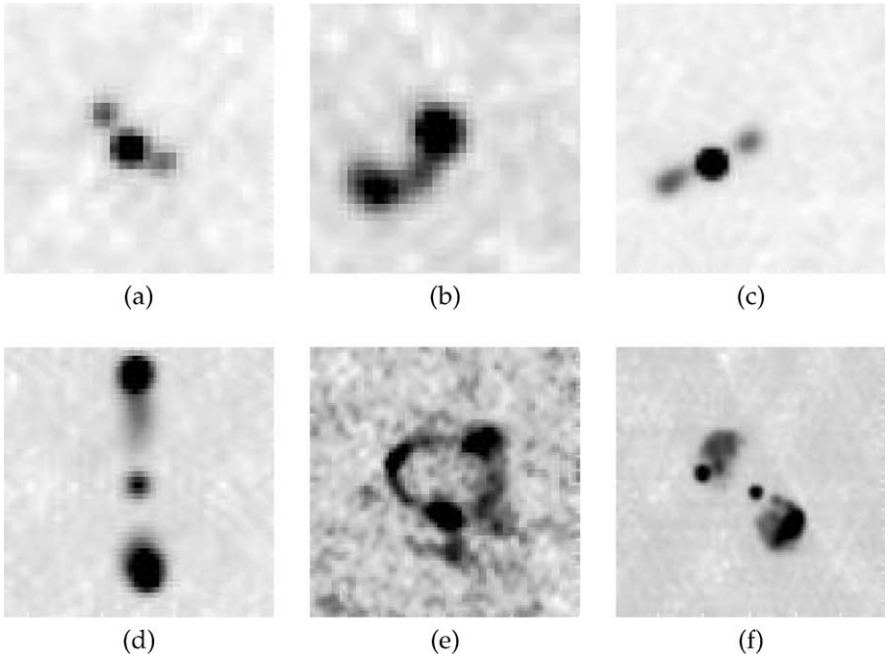
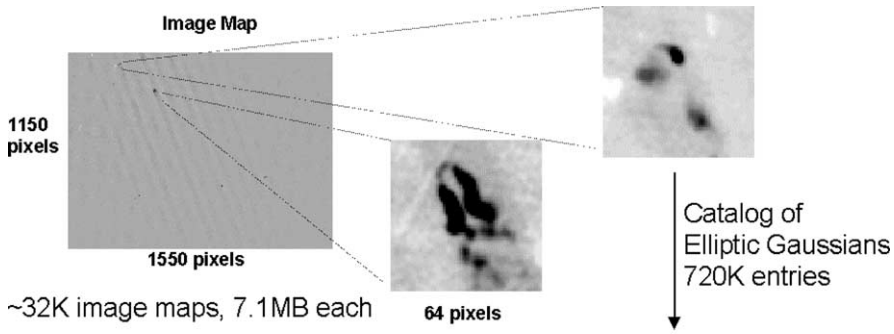


Fig. 1. Example radio sources from FIRST: (a)–(b) Bent-doubles, (c)–(d) Non-bent doubles, and (e)–(f) Complex sources.



Catalog entry

Radio Galaxy

RA	DEC	Peak Flux (mJy/bm)	Major Axis (arcsec)	Minor Axis (arcsec)	Position Angle (degrees)
00 56 25	-01 15 43	25.38	7.39	2.23	37.9
00 56 26	-01 15 57	5.50	18.30	14.29	94.2
00 56 24	-01 16 31	6.44	19.34	10.19	39.8

Fig. 2. FIRST data: images maps and catalog entries.

approximation to all but the most complex of radio sources, and several of the features they thought were important in identifying bent-doubles were easily calculated from the catalog.

Our first task in classifying the bent-doubles was to group the catalog entries (i.e., the elliptic Gaussians) into radio sources. Our algorithm starts with an entry in the catalog, searches for other entries within a region of interest of 0.96 arc minutes or restarts the search from each newly found entry, and repeats until no more new catalog entries are found within the region of interest. All catalog entries found in this search are collected to form a radio source. Next, the algorithm repeats the entire grouping procedure starting from the next available catalog entry, excluding any entries that are part of already existing radio sources.

After grouping the catalog entries into complex radio sources, we separated the data depending on the number of catalog entries that make up the sources. There is a data set each for the 1-entry sources, the 2-entry sources, the 3-entry sources, and the 3-plus-entry sources. This separation by the number of catalog entries was done for several reasons. First, using features from only the catalog, there were unlikely to be any 'bent-doubles' in the single-catalog-entry sources. Further, there are relatively few 3-plus-entry sources, all of which are 'interesting' to the astronomers, regardless of whether they are bent-doubles or not. So, we simply flag them and report them to the scientists.

Having removed the 1-entry and the 3-plus-entry radio sources from consideration, we further split the sources into two- and three-entry sources. This was done because the number of features extracted depends on the number of catalog entries, and we wanted a feature vector with a uniform length. However, this also meant that the size of the training set for the detection of bent-doubles was now divided into smaller training sets.

For the 2000 catalog, the number of radio sources as a function of the number of catalog entries they are composed of, is as follows:

Catalog entries	Radio sources
1	514637
2	66571
3	15059
3 +	6333

Once the radio sources (including the training set) were separated based on the number of catalog entries in the galaxy, we derived the features described below.

2.1. Features for bent-doubles

This section describes the features we are using to discriminate galaxies with bent-double morphology. Our focus is on features that are scale, rotation and translation invariant, as the bent-double pattern we are looking for has these properties. We are also interested in features that are not sensitive to small changes in the data (White, 1999). Of course, the features we select must be relevant to the problem.

We identified the features for the bent-double problem through extensive conversations with FIRST astronomers. When they justified their decisions classifying a radio source as a bent-double, they placed great importance on spatial features such as distances and angles. Frequently, the astronomers would characterize a bent-double as a radio-emitting 'core' with one or more additional components at various angles, which were usually side-wakes left by the core as it moved relative to the Earth.

We have concentrated our work on the 3-entry instances, because we have more labeled examples of this type. Our previous experience with this data suggested that the best accuracies are usually achieved using features extracted considering triplets of catalog entries. Therefore, in the remainder of this paper we focus on these features. A full list of features is described by Fodor, Cantú-Paz, Kamath, and Tang (2000).

There are different ways of ordering the entries in a 3-entry source, and the order affects the extraction of features. First, we need to identify the 'core' of the galaxy. If we consider the triangle formed by the centers of the three Gaussians, the core is the entry opposite to the longest side. In the following, assume that A is the core. Fig. 3 depicts a possible arrangement of the three catalog entries which is used to calculate the following features:

1. *totArea*: the sum of the three areas of the elliptic Gaussians
2. *peakFlux*: the maximum of the three peak fluxes of the entries

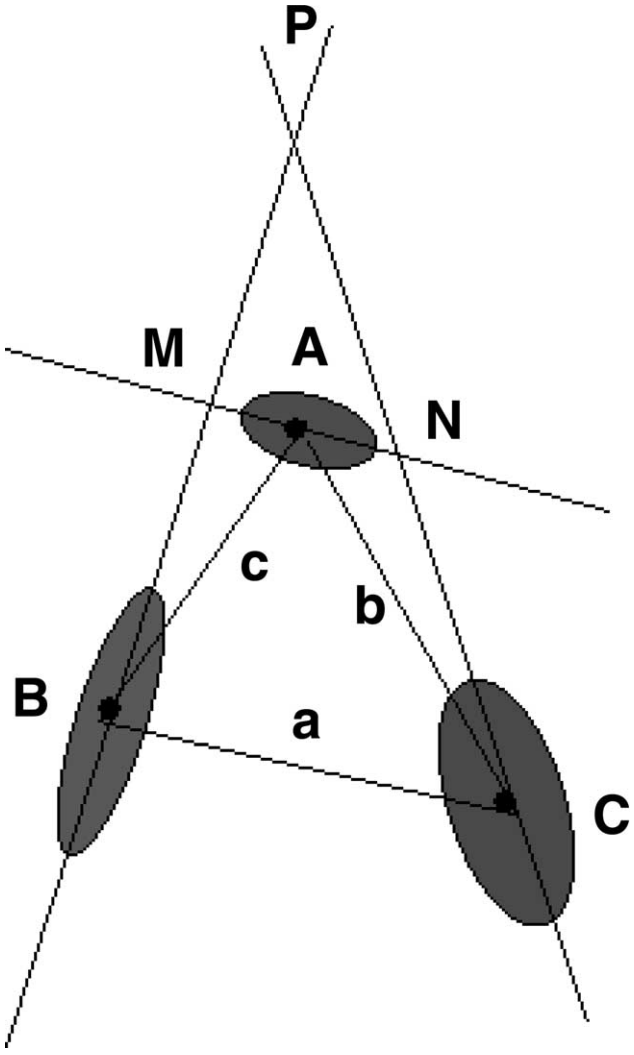


Fig. 3. An example of a 3-entry radio source.

3. *sumIntFlux*: the sum of the three integrated fluxes of the entries
4. *avgDiffusion*: the mean of the three diffusions
5. *totEllipt*: the sum of the three ellipticities
6. *maxFlux*: the maximum of the three integrated fluxes
7. *coreAngl*: the core angle, defined as the angle BAC in the triangle above
8. *angleAB*: angle ACB in the triangle above (between sides a and b)
9. *angleAC*: angle ABC in the triangle above (between sides a and c)
10. *totalBendGeom*: the total bentness of the source, equal to the sum of angles AMB and ANC.
11. *totalBendDiff*: the total bentness of the source, equal to the sum of $|\mathbf{APosAngle} - \mathbf{BPosAngle}|$ and $|\mathbf{APosAngle} - \mathbf{CPosAngle}|$, where **XPosAngle** denotes the angle of the major axis of entry **X**, measured counterclockwise from North.
12. ariAngl = acos BC/(AB + AC) : a measure of bentness (Lehàr, Buchalter, McMohan, Kochanek, & Muxlow, 2001) suggested by Ari Buchalter
13. *ABAnglSide*: the angle formed by the major axis of B with the AB segment, angle ABM
14. *ACAnglSide*: the angle formed by the major axis of C with the AC segment, angle ACN
15. *sumComDist*: the sum of the three pairwise distances between the centers of entries
16. *sumRelDist*: the sum of the three pairwise relative distances, calculated as

$$\frac{4\mathbf{XYComDist}}{\mathbf{XMaj} + \mathbf{XMin} + \mathbf{YMaj} + \mathbf{YMin}},$$

where for a pair of entries **X** and **Y**, **XYComDist** is the distance between their centers, and **Xmaj**, **XMin** denote the major and minor axis of entry **X**

17. *axialSym*: a symmetry measure given by the ratio of the ellipticities of entries B and C
18. arisym = AC/AB : a symmetry measure (Lehàr et al., 2001) suggested by Ari Buchalter
19. anotherSym = (AB + AC)/(AB + BC + AC) : another symmetry measure
20. consDemote: {0/1} flag, 1 if one of the non-core entries is far from the core, and 0 otherwise (B is considered far if $AB > 2 \times \text{const} \times (\mathbf{AMaj} + \mathbf{BMaj})$, where const is currently set to 3 arc seconds; similarly for C).

Unfortunately, our training set is relatively small, containing 195 examples for the three-catalog entry sources. Since FIRST scientists must manually label the bent- and non-bent-doubles, putting together an adequate training set is non-trivial. Moreover, scientists are usually subjective in their labeling of galaxies, and they often disagree in the hard-to-classify cases. There is also no ground truth we can use to verify our results. These issues imply that the training set itself is not very accurate, and there is a limit to the accuracy we can obtain with semi-automated techniques.

Among the 195 labeled examples of 3-entry sources, 28 are non-bent and 167 are bent-double galaxies. This unbalanced distribution of classes in the training set presents some problems in estimating the accuracy of the NNs, which are discussed in Section 4.

3. Genetic neural networks

GAs and NNs have been combined in two major ways. First, GAs have been used to train or to aid in the training of NNs. In particular, GAs have been used to search for the weights of the network, or to reduce the size of the training set by selecting the most relevant features. The second major type of collaboration is to use GAs to design the structure of the network. The structure largely determines the efficiency of the network and the classes of problems that it can solve. It is well known that to solve non-linearly separable problems, the network must have at least one hidden layer between the inputs and outputs; but determining the number and the size of the hidden layers is mostly a matter of trial and error. GAs have been used to search for these parameters, as well as for the pattern of connections and for developmental instructions to generate a network. The interested reader may consult the reviews by Branke (1995), Schaffer (1994) and Yao (1999).

This section first reviews some basic concepts from GAs. Then, we describe how to use GAs to train NNs, to select features, and to determine the topology of the network.

3.1. Genetic algorithms

Genetic algorithms are randomized search procedures inspired by the mechanics of genetics and natural selection (Goldberg, 1989). GAs operate on a population of individuals that represent possible solutions to a problem. The representation of a solution is defined by the user and may be as simple as a string of zeroes and ones, which is the representation used in this paper. The initial population may be created entirely at random or using some domain knowledge (in the form of previously known solutions, for example). The algorithm evaluates the individuals to determine how well they solve the problem at hand with an objective function, which is unique to each problem and must be supplied by the user. The individuals with better performance are selected into a mating pool to serve as parents of the next generation of individuals. GAs create new individuals using simple randomized operators that are inspired by sexual recombination (crossover) and mutation in natural organisms. The new solutions are evaluated, and the cycle of selection and creation of new individuals is repeated until a satisfactory solution is found or a predetermined time limit elapses.

There are numerous methods to select promising solutions into the mating pool. This paper uses binary (pairwise) tournaments, which is one of the simplest selection methods.

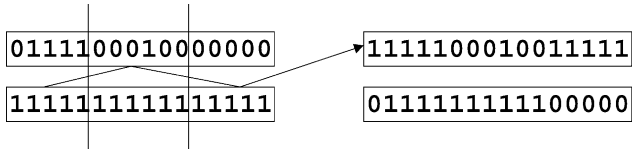


Fig. 4. Example of two-point crossover.

This selection randomly selects two individuals without replacement from the current population, and the most fit individual (according to the objective function) is incorporated into the mating pool. The random pairing of individuals is repeated twice to obtain a mating pool of the same size as the original population.

The crossover operator used in the experiments is multi-point crossover. This operator randomly chooses a pair of previously selected individuals from the mating pool and a number of crossover points along their chromosomes. Crossover exchanges segments of the two chromosomes delimited by the crossover points. Fig. 4 shows an example of two-point crossover.

In GAs, mutation occurs with a low frequency. It consists of flipping one randomly chosen bit from zero to one or vice versa. Some theoretical studies support the use of a mutation rate of $1/l$, where l is the length of the chromosome (Bäck, 1996; Mühlenbein, 1992). Although our application may not satisfy the assumptions made in those studies, this choice of mutation rate has been successful in several practical situations, and we adopt it for our experiments.

3.2. Training networks with genetic algorithms

Training a neural net is an optimization task with the goal of finding a set of weights that minimizes some error measure. The search space is high dimensional and, depending on the error measure, it may contain numerous local optima. Some traditional network training algorithms, such as backpropagation (BP), use some form of gradient search, and may get trapped in local optima. In contrast, GAs do not use any gradient information, and are likely to avoid getting trapped in a local optimum by sampling simultaneously multiple regions of the space.

A straightforward combination of GAs and NNs is to use the GA to search for weights that make the network perform as desired. The user prior to the experiment fixes the architecture of the network. In this approach, each individual in the GA represents a vector with all the weights of the network. This method has three variants:

- Start from scratch and use the weights found by the GA in the network without any further refinement (Caudell & Dolan, 1989; Montana & Davis, 1989; Whitley & Hanson, 1989). This is particularly useful when the activation function of the neurons is non-differentiable.
- Use BP or other methods to refine the weights found by the GA (Kitano, 1990b; Skinner & Broughton, 1995). The motivation of this approach is that GAs quickly

identify promising regions of the search space, but they do not fine-tune parameters very fast. So, GAs are used to find a promising set of initial weights from which a gradient-based method can quickly reach an optimum. This approach extends the processing time per individual, but sometimes the overall training time can be reduced because fewer individuals may need to be considered before reaching an acceptable solution.

- Use the GA to refine weights found by a traditional NN learning algorithm (Kadaba & Nygard, 1990). In general, seeding the initial population is an effective way to bias the GA toward good solutions.

These approaches are straightforward and have produced good results, but suffer from several problems. First, since adjacent layers in a network are usually fully connected, the total number of weights is $O(n^2)$, where n is the number of units. Longer individuals usually require larger populations, which in turn result in higher computational costs. For small networks, the GA can be used to search for good weights efficiently, but this method may not scale up to larger domains.Another drawback is the so-called permutations problem (Radcliffe, 1990). The problem is that by permuting the hidden nodes of a network, the representation of the weights in the chromosome would change, although the network is functionally the same. Some permutations may not be suitable for GAs because crossover might easily disrupt favorable combinations of weights. To ameliorate this problem, Thierens, Suykens, Vandewalle, and Moor (1991) suggested placing incoming and outgoing weights of a hidden node next to each other. An analysis by Hancock (1992) suggested that the permutation problem is not as difficult as it is often presented, and Thierens (1995) presented an encoding that avoids the permutations problem altogether.

3.3. Feature selection

Besides searching for weights, GAs may be used to select the features that are input to the NNs. The training examples may contain features that are irrelevant or redundant, but it is generally unknown a priori which features are relevant. Avoiding irrelevant and redundant features is desirable not only because they increase the size of the network and the training time, but also because they may reduce the accuracy of the network.

Applying GAs to the feature selection problem is straightforward, using what is referred to as the wrapper approach: the chromosome of the individuals contains one bit for each feature, and the value of the bit determines whether the feature will be used in the classification (Brill, Brown, & Martin, 1990; Brotherton & Simpson, 1995). The individuals are evaluated by training the networks (that have a predetermined structure) with the feature subset indicated by the chromosome. The resulting accuracy is used to calculate the fitness.

3.4. Using GAs to design the topology

As mentioned before, the topology of a network is crucial to its performance. If a network has too few nodes and connections, it may not be able to learn the required concept. On the other hand, if a network has too many nodes and connections, it may overfit the training data and have poor generalization. There are two basic approaches to use a GA to design the topology of a NN: use a direct encoding to specify every connection of the network or evolve an indirect specification of the connectivity.

The key idea behind direct encodings is that a neural network may be regarded as a directed graph where each node represents a neuron and each edge is a connection. A common method of representing directed graphs is with a binary connectivity matrix: the i, j-th element of the matrix is one if there is an edge between nodes i and j, and zero otherwise. The connectivity matrix can be represented in a GA simply by concatenating its rows or columns (Belew, McInerney, & Schraudolph, 1991; Miller, Todd, & Hegde, 1989). Using this method, Whitley, Starkweather, and Bogart (1990) showed that the GA can find topologies that learn faster than the typical fully connected feedforward network. The GA can be explicitly biased to favor smaller or sparsely connected networks, which can be trained faster. However, since each connection is explicitly coded, the length of the individuals is $O(n^2)$ (where n is the number of neurons), and the algorithm is not scalable to large problems.

A simple method to avoid specifying all the connections is to commit to a particular topology (feedforward, recurrent, etc.) and a particular learning algorithm, and then use the GA to find the parameter values that complete the network specification. For example, with a fully connected feedforward topology the GA may search for the number of layers and the number of neurons per layer. Another example would be to code the parameters of a particular learning algorithm, such as the momentum and the learning rate of BP (Belew et al., 1991; Marshall & Harrison, 1991). By specifying only the parameters for a given topology, the coding is very compact and well suited for a genetic algorithm. However, this method is constrained by the initial choice of topology and learning algorithm.

A more sophisticated approach to indirect representations is to use a grammar to encode rules that govern the development of a network. Kitano (1990a) introduced the earliest grammar-based approach. He used a connectivity matrix to represent the network, but instead of encoding the matrix directly in the chromosome, he used a graph-rewriting grammar to generate the matrix. The chromosomes contain rules that rewrite scalar matrix elements into 2×2 matrices.

In this grammar, there are 16 terminal symbols that are 2×2 binary matrices. There are 16 non-terminal symbols, and the rules have the form $n \rightarrow m$, where n is one of the scalar non-terminals, and m is a 2×2 matrix of non-terminals.

There is an arbitrarily designated start symbol, and the number of rewriting steps is fixed by the user.

Only the 16 right-hand sides of the rules are contained in the chromosome, the left side is implicit in the position of the rule. To evaluate the fitness of individuals, the rules are decoded and the connectivity matrix is developed by applying all the rules that match non-terminal symbols. Then, the connectivity matrix is interpreted and the network is constructed and trained with BP.

Perhaps the major drawback in this approach is that the number of units must be 2^i (where i is any non-negative integer), because after each rewriting step the size of the matrix doubles in each dimension.

Other examples of grammar-based developmental systems are the work of Boers and Kuiper (1992) with Lindenmayer systems, Gruau's cellular encoding method (Gruau, 1992), and the system of Nolfi, Elman, and Parisi (1994) that simulates cell growth, migration, and differentiation.

4. Experiments

This section details the experimental methods and the results that we obtained with six combinations of NNs and GAs.

The programs were written in C++ and compiled with g++ version 2.96. The experiments were executed on a single processor of a Linux (Red Hat 7.1) workstation with dual 1.5 GHz Intel Xeon processors and 512 Mb of memory. The programs used a Mersenne Twister random number generator.

All the GAs used a population of 50 individuals. As mentioned in Section 3.1, we used a simple GA with binary encoding, pairwise tournament selection, and multi-point crossover. The number of crossover points was varied in each experiment according to the length of the chromosomes, l. In all cases, the probability of crossover was 1, and the probability of mutation was set to $1/l$. The initial population was initialized uniformly at random.

The experiments used feedforward networks with one hidden layer. All neurons are connected to a 'bias' unit with constant output of 1.0. Unless specified otherwise, the output units are connected to all the hidden units, which in turn are connected to all the inputs. In feedforward operation, the units compute their net activation as

$$\text{net} = \sum_{i=1}^{d} x_i w_i + w_0, \tag{1}$$

where d is the number of inputs to the neuron, x_i is an input, w_i is the corresponding weight, and w_0 is the weight corresponding to the 'bias' unit. Each unit emits an output according to

$$f(\text{net}) = \tanh(\beta \text{net}), \tag{2}$$

where β is a user-specified coefficient. Simple BP was used in some of the experiments. The weights from the hidden to the output layer were updated using

$$\Delta w_{kj} = \eta \delta_k y_j = \eta (t_k - z_k) f'(\text{net}_k) y_j, \tag{3}$$

where η denotes the learning rate, k indexes the output units, t_k the desired output, z_k the actual output, f', is the derivative of f, and y_j is the output of the j-th hidden unit. The weights from the i-th input to the hidden layer were updated using

$$\Delta w_{ji} = \eta \left[\sum_{k=1}^{c} w_{kj} \delta_k \right] f'(\text{net}_j) x_i. \tag{4}$$

In all experiments, each feature in the data was linearly normalized to the interval $[-1,1]$. The type of galaxy was encoded in one output value (-1 for bent and 1 for non-bent). When back-propagation was used, the examples were presented in random order for 20 epochs. All the results reported are averages over 10 runs of the algorithms. Comparisons are made using standard paired t-tests with 95% confidence.

4.1. Fitness calculation

One of the crucial design decisions for the application of GAs is the calculation of fitness values for each member of the population. Since we are interested in networks that predict accurately the type of galaxies that were not used in training, the fitness calculation must include an estimate of the generalization ability of the networks.

There are multiple ways to estimate generalization that we could use for our experiments with GAs. One approach is to partition the data into training and testing sets, use the training set as input to BP, and use the accuracy on the testing set to calculate the fitness. If enough data are available, the generalization may be better estimated by dividing the training data into training and validation sets. The training set is used to adjust the weights with back propagation, and the accuracy of the trained network on the validation set is used to calculate the fitness. The final network that results from the GA execution is tested on an independent testing set that has not used in any way during the execution of the GA. Ideally, the experiment should be repeated with different partitions of training, validation, and testing sets.

Since we do not have much training data, the procedure above is not practical in our case. An alternative way to estimate the generalization of the network is to use crossvalidation experiments. In this method, the data D is divided into k, non-overlapping sets, $D_1, \ldots D_k$. At each iteration i (from 1 to k), the network is trained with $D \backslash D_i$ and tested on D_i. In our experiments, we used the accuracy estimated by a single five-fold crossvalidation as the fitness. A better estimate of accuracy would be to use an average of multiple crossvalidation experiments, but we found the cost excessive.

Following our earlier work with this data using decision trees (Fodor et al., 2000), we first calculated the accuracy as the fraction of instances classified correctly. This is the most common measure of accuracy, but in our case it may give overly optimistic results, because our labeled data is unbalanced with far more examples of bent-double than non-bent-double galaxies. To correct for this unbalance, we also present results where we calculate the accuracy as the geometric mean of the accuracies for each class of galaxy (bent and non-bent) (Kubat & Matwin, 1997).

4.2. Training networks with GAs

We implemented the first of the methods described in Section 3.2: the GA was used to find the network's weights. The network had 20 inputs that correspond to each of the features in the data, 25 hidden nodes, and one output. Each weight was represented with 10 bits, and the range of possible weights was [−10,10].

For this experiment, the GA used a population of 50 individuals, each with a length of $l = 5510$ bits (there are 551 total weights). The number of crossover points was set at 25, and the mutation rate was 0.00018 ($\approx 1/l$). As in all experiments, pairwise tournament selection without replacement was used to select promising solutions.

The second training method described in Section 3.2 is to run BP using the weights represented by the individuals in the GA to initialize the network. We implemented this method and used the same network architecture and GA parameters as in the previous experiment. Each network was trained with 20 epochs of BP with a learning rate η of 0.1 and β of 0.4.

The entries WEIGHTS and WEIGHTS + BP in Tables 1 and 2 present the average accuracy of the best networks found in each run of the GA for these two sets of experiments. The results highlighted in bold in the tables are the best results and those not significantly worse than the best (according to the *t*-test, which may detect more differences than there actually exist).

Table 1
Mean accuracies on the bent and non-bent doubles and overall accuracy for different combinations of GAs and NNs using the *arithmetic* mean of class-wise accuracies as fitness. The numbers in parenthesis are the standard errors, and the results in bold are the best and those not significantly worse than the best

Method	Bent-Doubles	Non-Bent	Overall
WEIGHTS	97.15 (0.80)	30.36 (8.38)	85.68 (2.07)
WEIGHTS + BP	95.42 (0.43)	43.28 (3.49)	87.58 (0.40)
FEATURE SEL	95.53 (0.52)	**60.40** (**4.27**)	**90.00** (**0.56**)
PARAMETERS	96.02 (0.54)	35.12 (3.85)	87.02 (0.44)
MATRIX	95.66 (0.26)	41.29 (3.98)	88.00 (0.47)
GRAMMAR	95.36 (0.37)	40.78 (3.67)	87.07 (0.33)

Table 2
Mean accuracies on the bent and non-bent doubles and overall accuracy for different combinations of GAs and NNs using the *geometric* mean of class-wise accuracies as fitness. The numbers in parenthesis are the standard errors, and the results in bold are the best and those not significantly worse than the best

Method	Bent-doubles	Non-bent	Overall
WEIGHTS	86.34 (2.83)	**78.01** (**4.13**)	80.98 (2.41)
WEIGHTS + BP	**91.89** (**0.67**)	75.23 (0.87)	81.68 (0.53)
FEATURE SEL	**92.99** (**0.55**)	**83.65** (**1.41**)	**87.51** (**0.77**)
PARAMETERS	**92.35** (**0.89**)	69.13 (1.56)	78.76 (0.57)
MATRIX	**93.58** (**0.46**)	70.77 (1.34)	80.22 (0.69)
GRAMMAR	**92.84** (**0.69**)	73.73 (1.40)	81.78 (0.72)

4.3. Feature selection

The next combination of GAs and NNs is to use the GA to select a subset of features that will be used to train the networks, as described in Section 3.3. As in previous experiments, we set the number of hidden units to 25, the learning rate to 0.1 and β to 0.4. The networks were trained with 20 epochs of BP.

Our data has 20 features, and therefore the chromosomes in the GA are 20 bits long. The GA used one-point crossover and the same parameters as in previous experiments. The accuracy results are labeled FEATURE SEL and are significantly better than the other overall results in Tables 1 and 2.

The GAs consistently selected approximately half of the features. Table 3 and 4 present the features selected by each of the ten runs using both accuracy measurements. The GAs frequently selected features that appear to be relevant to the problem of identifying bent-double galaxies, such as sumRelDist, ariSym, angleAB, and peakFlux.

4.4. Using GAs to design the networks

For our first application of GAs to network design, the GA was used to find the number of hidden units, the parameters for BP, and the range of initial weights as described in Section 3.4. The learning rate was encoded with four bits and the range of possible values was [0,1]. The coefficient β for the activation function was also encoded with four bits and its range was [0,1]. The upper and lower ranges for the initial weights were encoded with five bits each and were allowed to vary in [−10,0] and [0,10], respectively. Finally, the number of hidden units was represented with seven bits and could take values in [0,127].

After extracting the parameters from a chromosome, a network was built and initialized according to the parameters and trained with 20 epochs of BP. As in all the experiments, the crossvalidated accuracy (arithmetic and geometric mean versions) was used as the fitness of the networks. There are no explicit biases in the fitness to prefer

Table 3
Features selected on 10 runs by the GA using the *arithmetic* mean of class-wise accuracies as fitness. The last row is the number of times a feature was selected, and the last column is the number of features selected on each run. The features are described in Section 2.1

Feature	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Selected
		1	1	1		1	1	1								1		1			8
		1				1	1	1	1			1				1	1	1	1	1	11
			1		1					1	1			1	1	1	1	1			9
	1	1						1				1				1		1	1		7
		1				1		1						1		1		1			6
	1	1		1				1				1				1		1			7
	1			1		1	1	1	1					1	1	1		1			10
		1	1	1		1		1			1			1		1		1			8
	1	1	1	1	1			1	1			1			1	1		1	1		12
		1			1									1		1			1		5
Total	4	8	4	5	3	5	3	8	3	1	2	4	0	5	3	10	2	9	4	1	

smaller networks, but there is an implicit bias toward networks that can learn quickly, since we are using only 20 epochs of BP. It is probable that small networks learn faster than larger ones, and so it is likely that the GA favors small networks. The GA used two-point crossover and the same parameters as in previous experiments. The accuracy results are labeled PARAMETERS in Tables 1 and 2. On average, the best learning rate found by the GA was 0.82 (with 0.06 std. error), which is higher than the usual recommendation of 0.1–0.2 (Duda, Hart, & Stork, 2001). Perhaps the learning rate is high because of the implicit bias for learning quickly. This bias may also explain the average number of hidden units being relatively small at 15.6 (std. error 2.8). The average β was 0.16 (0.01), and the range of initial weights was [−3.51,3.45] (both with std. errors of 0.4).

In the next experiment, we used the GA to search for a connectivity matrix as described in Section 3.4. As in previous experiments, we fixed the number of hidden units to 25, the learning rate to 0.1 and β to 0.4. The neurons are numbered consecutively starting with the inputs and followed by the hidden units and outputs. The connectivity matrix is encoded by concatenating its rows. Since we allow direct connections between the inputs and the outputs, the string length is (hidden + outputs)inputs + hidden × outputs = (26 × 20) + (25 × 1) = 545 bits. For this longer string, we use 10 crossover points, and the same GA parameters as before. The results corresponding to this method are labeled MATRIX in Tables 1 and 2.

We also implemented Kitano's graph rewriting grammar method. We limited the number of rewriting steps to 6, resulting in networks with at most 64 units. Since the chromosomes encode four 2 × 2 binary matrices for each of the 16 rules, the string length is 256 bits. The GAs used five crossover points. The results obtained with this method are labeled GRAMMAR in Tables 1 and 2.

4.5. Comparison and discussion

The results using the standard accuracy (table 1) do not show major differences among the different methods that we tried. Only the feature selection method has a significantly better overall accuracy than any other method, as well as a better accuracy on detecting non-bent doubles.

Using the geometric mean of the accuracies of each type of galaxy (Table 2), the overall results are lower than with the standard accuracy measure. This reduction is expected,

Table 4
Features selected on 10 runs by the GA using the *geometric* mean of class-wise accuracies as fitness. The last row is the number of times a feature was selected, and the last column is the number of features selected on each run. The features are described in Section 2.1

Feature	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Selected
	1	1			1		1	1	1			1	1		1	1	1		1		12
			1	1	1	1		1				1	1		1	1	1		1	1	12
	1	1	1		1	1			1			1			1	1		1			10
	1	1						1	1			1				1		1			7
		1		1				1	1						1	1		1	1		8
			1			1	1	1	1							1		1	1		8
		1				1	1	1	1			1			1	1		1			9
	1			1	1	1	1	1		10		1				1			1		10
				1		1	1	1	1							1		1			7
		1					1	1								1	1	1	1		7
Total	4	6	3	4	4	6	6	9	7	1	0	6	2	0	5	10	3	7	6	1	

since our training data is unbalanced toward one of the two classes: with the standard accuracy measure, even if the networks make numerous mistakes in the minority class (the non-bent doubles), the overall accuracy is dominated by the high accuracy in the majority class. In contrast, the geometric mean gives equal weight to the accuracies on both types of galaxies in the overall performance. We can observe a notable increase in the accuracy for the non-bents in Table 2. Overall, the feature selection method seems to outperform the others.

We also performed numerous experiments with networks designed by hand. The best parameters that we could find for 20 epochs of BP were those used in the experiments with the GAs: $\beta = 0.1$, the learning rate was 0.4, and the number of hidden was 25. The average of ten, ten-fold cross-validation experiments resulted in an overall accuracy of 84.63% (with std. error of 0.4), which is significantly worse than all of the results in Table 1, except for WEIGHTS. However, the accuracy of the hand-designed network on the non-bents is only 16.4% (1.7) and on the bents is 99.69% (0.16). The overall accuracy estimated with the geometric mean is a disappointing 23.41% (2.02).

Increasing the number of training epochs to 100 raised the standard accuracy to 88.52% (0.44) and the geometric mean accuracy to 72.69% (0.32). The accuracy on the non-bents also improved to 56.7%, while the accuracy on the bents decreased slightly to 94.38%.

We also applied decision trees to this data. The decision trees used the Gini splitting criterion and pessimistic error pruning. The overall arithmetic mean accuracy of ten ten-fold crossvalidation experiments was 87% with a standard error of 0.42. This result is less accurate than all the combinations of GAs and NNs reported in Table 1, except for WEIGHTS.

Using the geometric mean and ten ten-fold crossvalidations, we estimated the overall accuracy of a single tree as 74.09% (with standard error of 1.3%), which is less accurate than all the GAs and NNs combinations in Table 2.

5. Conclusions

This paper presented a comparison of six combinations of GAs and NNs for the identification of bent-double galaxies in the FIRST survey. Our experiments suggest that, for this application, some combinations of GAs and NNs can produce accurate classifiers that are competitive with networks designed by hand. For our application, we found few differences among the GA and NN combinations that we tried. The only consistently best method was to use the GA to select the features used to train the networks, which suggests that some of the features in the training set are irrelevant or redundant.

There are several avenues to extend this work. The highly unbalanced training set presents some difficulties that could be avoided or ameliorated by including more examples of the minority class. Extending the training set is non-trivial, as we mentioned in Section 2.1, because the labeling is subjective and disagreements among the experts are common.

Other optimization techniques, evolutionary and traditional, can be used to train NNs. In this paper we used a simple genetic algorithm with a binary encoding, but other evolutionary algorithms operate on vectors of real numbers, which can be directly mapped to the network's weights or the BP parameters (but not to a connectivity matrix, a grammar, or a feature selection application). There are other combinations of GAs and NNs that we did not include in this study, but appear promising. For instance, since evolutionary algorithms use a population of networks, a natural extension of this work would be to use evolutionary algorithms to create ensembles that combine several NNs to improve the accuracy of classifications.

A big disadvantage in using GAs in combination with NNs is the long computation time required. This can be an obstacle to applying these techniques to larger data sets, but there are numerous alternatives to improve the performance of GAs. For instance, we could approximate the fitness evaluation using sampling or we can exploit the inherently parallel nature of GAs using multiple processors.

Acknowledgements

We gratefully acknowledge our FIRST collaborators Robert Becker, Michael Gregg, David Helfand, Sally Laurent-Muehleisen, and Richard White for their technical interest and support of this work. We would also like to thank Imola K. Fodor and Nu Ai Tang for useful discussions and computational help.UCRL-JC-146705. This work was performed under the auspices of the US Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

Adams, A., & Woolley, A. (1994). Hubble classification of galaxies using neural networks. *Vistas in Astronomy*, *38*, 273–280.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice*, New York: Oxford University Press.

Becker, R. H., White, R., & Helfand, D. (1995). The FIRST survey: Faint images of the radio sky at twenty-cm. *Astrophysical Journal*, *450*, 559–599.

Belew, R., McInerney, J., & Schraudolph, N. (1991). *Evolving networks: Using the genetic algorithm with connectionist learning. Proceedings of the Second Artificial Life Conference*, New York: Addison-Wesley, pp. 511–547.

Boers, J. W., & Kuiper, H (1992). *Biological metaphors and the design of modular artificial neural networks*. Master's thesis, Leiden University, The Netherlands.

Branke, J. (1995). Evolutionary algorithms in neural network design and training—a review. In J. T. Alander (Ed.), *Proceedings of the First*

Nordic Workshop on Genetic Algorithms and their Applications (pp. 145–163). Finland: Vaasa.

Brill, F. Z., Brown, D. E., & Martin, W. N., (1990). *Genetic algorithms for feature selection for counter-propogation networks* (Tech. Rep. No. IPC-TR-90-004). Charlottesville: University of Virginia, Institute of Parallel Computation.

Brotherton, T. W., & Simpson, P. K. (1995). Dynamic feature set training of neural nets for classification. In J. R. McDonnell, R. G. Reynolds, & D. B. Fogel (Eds.), *Evolutionary Programming IV* (pp. 83–94). MIT Press: Cambridge, MA.

Caudell, T. P., & Dolan, C. P. (1989). Parametric connectivity: training of constrained networks using genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 370–374). San Mateo, MA: Morgan Kaufmann.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification*, New York: Wiley.

Fodor, I. K., Cantú-Paz, E., Kamath, C., & Tang, N. (2000). Finding bent-double radio galaxies: A case study in data mining. *Interface: Computer Science and Statistics*, 33.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*, Reading, MA: Addison-Wesley.

Grönross, M. A (1998). *Evolutionary design of neural networks*. Master's thesis, University of Turku, Finland.

Gruau, F. (1992). Genetic synthesis of boolean neural networks with a cell rewritting developmental process. In D. Whitley, & J. D. Schaffer (Eds.), *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks* (pp. 55–74). Los Alamitos, CA: IEEE Computer Society Press.

Hancock, P. J. B. (1992). Recombination operators for the design of neural nets by genetic algorithm. In R. Männer, & B. Manderick (Eds.), *Parallel Problem Solving from Nature II* (pp. 441–450) Amsterdam: Elsevier Science.

Kadaba, N., & Nygard, K. E. (1990). Improving the performance of genetic algorithms in automated discovery of parameters. In B. Porter, & R. Mooney (Eds.), *Machine Learning: Proceedings of the Seventh International Conference* (pp. 140–148). Morgan Kaufmann: San Mateo, CA.

Kitano, H. (1990a). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, *4*(4), 461–476.

Kitano, H. (1990b). Empirical studies on the speed of convergence of neural network training using genetic algorithms. *Proceedings of the Eighth National Conference on Artificial Intelligence*, 789–795.

Kubat, M., & Matwin, S. (1997). *Addressing the curse of imbalanced training sets: One-sided selection*. *Proceedings of the 14th International Conference on Machine Learning*, San Francisco, CA: Morgan Kaufmann, pp. 179–186.

Lehàr, J., Buchalter, A., McMahon, R. G., Kochanek, C. S., & Muxlow, T. W. B. (2001). An efficient search for gravitationally lensed radio lobes. *Astrophysical Journal*, *547*, 60–76.

Marshall, S. J., & Harrison, R. F. (1991). *Optimization and training of feedforward neural networks by genetic algorithms*. *Proceedings on the Second International Conference on Artificial Neural Networks and Genetic Algorithms*, Berlin: Springer, pp. 39-43.

Miller, G. F., Todd, P. M., & Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 379–384). San Mateo, CA: Morgan Kaufmann.

Montana, D. J., & Davis, L. (1989). *Training feedforward neural networks using genetic algorithms*. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, pp. 762–767.

Mühlenbein, H. (1992). How genetic algorithms really work: I.Mutation and Hillclimbing. In R. Männer, & B. Manderick (Eds.), *Parallel Problem Solving from Nature II* (pp. 15–25) Amsterdam: Elsevier Science.

Nolfi, S., Elman, J. L., & Parisi, D. (1994). Learning and evolution in neural networks. *Adaptive Behavior*, *3*(1), 5–28.

Odewahn, S., & Nielsen, M. (1994). Star-galaxy separation using neural networks. *Vistas in Astronomy*, *38*, 281–286.

Odewahn, S., Stockwell, E., Pennington, R., Humphreys, R., & Zumach, W. (1992). Automated star/galaxy discrimination with neural networks. *The Astronomical Journal*, *103*(1), 318–331.

Radcliffe, N. J (1990). *Genetic neural networks on MIMD computers*. Unpublished doctoral dissertation, University of Edinburgh, Scotland.

Roberts, S. G., & Turenga, M. (1995). Evolving neural network structures. In D. Pearson, N. Steele, & R. Albrecht (Eds.), *International Conference on Genetic Algorithms and Neural Networks* (pp. 96–99). New York: Springer-Verlag.

Schaffer, J. D. (1994). Combinations of genetic algorithms with neural networks or fuzzy systems. In J. M. Zurada, R. J. Marks, II, & C. J. Robinson (Eds.), *Computational Intelligence Imitating Life* (pp. 371–382). New York: IEEE Press.

Siddiqi, A. A., & Lucas, S. M. (1998). A comparison of matrix rewriting versus direct encoding for evolving neural networks. *Proceedings of the 1998 International Conference on Evolutionary Computation*, Piscataway, NJ: IEEE Press, pp. 392–397.

Skinner, A., & Broughton, J. Q. (1995). Neural networks in computational material science: Training algorithms. *Modelling and Simulation in Material Science and Enginnering*, *3*, 371–390.

Storrie-Lombardi, M., Lahav, O., Sodre, L., & Storrie-Lombardi, L. (1992). Morphological classification of galaxies by artificial neural networks. *Monthly Notices of Royal Astronomical Society*, *259*, 8–12.

Thierens, D., 1995. Analysis and design of genetic algorithms. Doctoral dissertation, Katholieke Universiteit Leuven, Leuven, Belgium.

Thierens, D., Suykens, J., Vandewalle, J., & Moor, B. D. (1991). Genetic weight optimization of a feedforward neural network controller. *Proceedings of the Conference on Neural Nets and Genetic Algorithms*, Berlin: Springer, pp. 658–663.

White, R. L. (1999). *Private Communication*.

White, R. L., Becker, R., Helfand, D., & Gregg, M. (1997). A catalog of 1.4 GHz radio sources from the FIRST survey. *Astrophysical Journal*, *475*, 479.

Whitley, D., & Hanson, T. (1989). Optimizing neural networks using faster, more accurate genetic search. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 391–397). San Mateo, CA: Morgan Kaufmann.

Whitley, D., Starkweather, T., & Bogart, C. (1990). Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing*, *14*, 347–361.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, *87*(9), 1423–1447.