Discovering Gene Networks with a Neural-Genetic Hybrid

Edward Keedwell and Ajit Narayanan

Abstract—Recent advances in biology (namely, DNA arrays) allow an unprecedented view of the biochemical mechanisms contained within a cell. However, this technology raises new challenges for computer scientists and biologists alike, as the data created by these arrays is often highly complex. One of the challenges is the elucidation of the regulatory connections and interactions between genes, proteins and other gene products. In this paper, a novel method is described for determining gene interactions in temporal gene expression data using genetic algorithms combined with a neural network component. Experiments conducted on real-world temporal gene expression data sets confirm that the approach is capable of finding gene networks that fit the data. A further repeated approach shows that those genes significantly involved in interaction with other genes can be highlighted and hypothetical gene networks and circuits proposed for further laboratory testing.

Index Terms—Gene expression analysis, neural networks, genetic algorithms, reverse-engineering, gene interactions.

1 Introduction

The advent of DNA microarrays and gene chips has allowed biologists an unprecedented view of the genetic behavior of organisms. The latest Affymetrix gene chips contain more than one million unique oligonucleotide features covering more than 39,000 transcript variants that, in turn, represent more than 33,000 well-substantiated human genes. DNA arrays are capable of capturing the expression level of these genes at one particular timepoint and, while they have been criticized for a variety of reasons (e.g., noise in measurement and, more importantly, lack of knowledge of how much mRNA actually makes it through to translation), they currently represent the best opportunity for scientists to examine the qualitative and quantitative expression of most genes in the human genome. The resulting data generated by DNA arrays, however, has proven somewhat of a challenge for traditional analytical techniques due largely to the size of the arrays themselves and the inherent combinatorial problems that arise from analysing a data set with up to 33,000 variables. This paper describes a novel computational technique using the repeated application of genetic algorithms to elucidate gene interactions from gene expression profile data. We show that the approach, previously used for classification purposes, can also be modified for use as a tool for determining the interactions between genes over time. Experiments on artificial data and also two real-world gene expression data sets show the efficacy of this approach.

Generally speaking, DNA array profile experiments fall into two categories, temporal and classification based approaches. Classification DNA array data is obtained by sampling just once a number of individual organisms or tissue samples which differ in some respect from each other. Classification problems are most often seen in cancer studies, where individuals are sampled and separated into classes according to either an independent diagnosis using traditional methods or a class being determined by the pathology of the individuals involved. The task for this analysis technique is typically to differentiate between those individuals diagnosed with cancer and those without, based solely on the gene expression values of each of those individuals at one time-point. Such genetic differences are not necessarily causes of cancer: They can equally be caused by the cancer and, therefore, more extensive studies that include some temporal element (repeated measurement) are becoming popular. The neural-genetic approach described here has been shown to successfully discover sets of genes from DNA array data that classify static, nontemporal data accurately [12].

Temporal DNA array data, on the other hand, is created by measuring gene expression values over a number of timesteps, often while subjecting the organism to some stimulus, so that the behavior of genes in response to certain experimental conditions can be observed and measured to determine possible cause-effect relationships. The measurements gained from these studies often span thousands of genes (fields, attributes) and only a few timesteps (records), which makes it relatively unusual in structure and not as amenable to traditional analysis as other biological databases. The goal of analyzing such data is to identify possible direct excitatory and inhibitory connections between genes, gene products and proteins if the timesteps are close enough (a few seconds or minutes) or indirect connections if not (several minutes or hours). Following a gene regulatory network model can be considered as a matrix $T_{i,j}$ of positive, negative or zero connections by which one transcription factor can enhance or repress another. When a particular matrix entry is nonzero, there is a regulatory connection from gene product j to gene i. The regulation is enhancing if the entry is positive and repressing if it is negative. A zero entry signifies no connection between gene product j and gene i. The aim of this paper is not to derive full gene regulatory networks involving regulatory ele-

• *The authors are with the School of Engineering, Computer Science and Mathematics, Harrison Building, North Park Road, University of Exeter, Exeter, UK, EX4 4QF. E-mail: {E.C.Keedwell, A.Narayanan}@ex.ac.uk.*

Manuscript received 1 Sept. 2004; revised 3 Dec. 2004; accepted 14 Apr. 2005; published online 31 Aug. 2005.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBBSI-0113-0904.

ments, but to identify connections and interactions between genes that fit the gene expression profile data and that can lead to the subsequent analysis of regulatory and transcription factors that help in the construction of T.

Most current work in gene expression analysis deals with these two types of data set (DNA arrays can also be used to measure subcellular localization of gene products and detect mutations, for example). The common difficulty here is that of "underdeterminism," i.e., there are very many variables in comparison with the number of records available. While this imbalance may be reduced as DNA arrays become cheaper and more accessible, underdeterminism will exist in some form because the number of variables measured will also increase as the technology develops. Another common problem is that often combinations of gene are required to correctly determine the class of the sample or the activation of the regulated gene. That is, one gene is unlikely by itself to classify a sample or affect another gene. In such circumstances, the search space grows to a much larger potential search space of countless combinations. It is this "curse of dimensionality" that causes problems for traditional statistical and algorithmic approaches. Recently, "intelligent" approaches based on machine learning techniques in artificial intelligence have been applied to the problems of gene expression data analysis (e.g., [17], [18]). The next section first details some approaches that have been used for temporal gene expression analysis and, second, focuses on machine learning methods.

1.1 Recent Approaches to Temporal Gene Expression Analysis

A review of recent methods for extracting gene regulatory networks from temporal gene expression data [5], [9] reveals that the main problem is how to deal with the large number of different models that display broadly the same behavior (the "curse of dimensionality" described above). This is caused mainly by underdeterminism and can be counteracted in a number of ways. The majority of the methods make use of automated or human-derived methods for determining functional groups of genes within the data. By using clustering, or incorporating currently known biological information, these approaches are able to discover hypothetical regulatory connections between groups of genes. These approaches can also provide biologically plausible networks, provided that additional information about the organism is available. However, while the volume of additional information is increasing, such as that provided by Gene Ontology [20] and by transcription factors [4], there remain a large number of gene expression data sets for which this extra information is not available. The neural-genetic approach to be described below mitigates underdeterminism in two significant ways that do not require extra data to be used. First, each gene has a restriction on the number of genes that can affect it in the network. For example, gene networks have been obtained from prokaryotic (*E. coli*) gene expression data where the number of connections allowed between genes is limited to between one and five and between two and three on average [25]. This restriction is in line with ideas developed from chaos theory [10], which proposed that the behavior of one gene is regulated by only a handful of other genes in the network. The maximum connectivity in our experiments has been set as five genes. Second, the approach can be repeated a number of times and only those genes that are present in a majority of the runs are allowed to be included in the final network. These restrictions are designed to allow the algorithm to discover robust connection networks.

1.2 Machine Learning Approaches to Temporal Gene Expression Analysis

A number of methods have been used to generate gene regulatory and gene connectivity networks from (often artificial) data, including: Bayesian Networks [23], clustering [6], [16], traditional statistics [15], [26], unsupervised neural networks [24], evolutionary algorithms [1], and supervised learning algorithms [8]. A succession of papers [1], [2], [3] describe an evolutionary approach using genetic algorithms (GAs) to extract the gene networks from gene expression data. Mostly, they use the approach described by Weaver et al. [27] as the model for forward activation and reverse engineering their gene networks. The GA is applied to the problem in a number of ways, and the most relevant to this paper is the weight matrix method. In this approach, the chromosome of the GA is an encoded matrix of floating point values that correspond to the weight matrix between gene timesteps. Each individual, therefore, is a representation of the weight matrix as described in Weaver et al. [27]. The weight matrix is a set of connections from -1 to +1 that are summed and propagated through the sigmoid function to give the activation levels for each of the genes (which have previously been normalized). The fitness function for the GA is calculated as the sum, over all timesteps, of the differences between the predicted and actual levels of activation for the gene expression. The GA is then allowed to use its operators (crossover and mutation) to optimize this matrix of weights. Another objective factored into the fitness function is that sparse matrices are required in this problem, to remove the possibility of all genes having an effect on all other genes. Therefore, a measure of the number of zero weights (signifying no effect) in the chromosome is used so that the fitness of individuals is partly based on the number of zeroes in the solution.

The matrix configuration of the genetic algorithm, however, suggests that the complexity of the algorithm is unmanageable on real-world problems where the number of genes is large. The matrix by definition must be of size g^2, where g is the number of genes, which for a network of 6,000 genes yields a chromosome size of 36,000,000 genes. This is certainly too many for most GA programs and machines to handle and, even if this is not the case, it will take time to evaluate a population of just 10 of these chromosomes over several generations.

The most recent evolutionary approaches to this problem use extended evolutionary algorithms for evolving network topology as well as mathematical models (e.g., [22]) and a multiobjective GA with the simplex method [13]. The approach of Spieth et al. [22] is to use a GA for generating populations of structures of possible networks and S-Systems (a power-law formalism consisting of nonlinear differential equations) for optimizing the parameters of gene expression concentrations over time, where the change in each mRNA level depends on all or only on some mRNA concentrations at the previous time step. A problem with this approach, as pointed out by the authors, is that the S-

System formalism requires a large number of parameters to be estimated. The total number of parameters in S-Systems is 2N(N + 1), with N representing the number of genes. An increase in the number of genes results in a quadratic increase in the number of parameters to infer. While this approach may work well with gene expression data sets consisting of small numbers of genes, for the real-world examples discussed below the implication is that almost 25,000 parameters have to be estimated for the 112-gene rat spinal cord data and almost 12.5 million parameters for the 2,500-gene yeast data. Spieth et al. [22] analyze artificial data sets consisting of at most 60 genes and refer to the need to try their approach on real-world data sets that describe "... large-scale systems with a minimum of 100 genes." However, large-scale systems can be expected to consist of significantly more than 100 genes and, therefore, the generalization of their S-system approach to real-world data is not clear. Koduru et al. [13] adopt a complex hybrid method for deriving model parameter estimates involving a multiobjective GA with the simplex algorithm of Nelder and Mead [19] as well as a novel "fuzzy dominance" technique. They report results on estimating 10 parameters using their approach. Again, it is not clear how their method can be scaled to more complex networks involving hundreds and possibly thousands of genes. In summary, while these approaches are promising, they rely solely on the use of artificial data involving only a small number of genes. These experiments are useful, but the ability of these algorithms as well as of standard genetic algorithms to generalize to the huge search spaces of real-world gene expression data is not known.

Overall, machine learning approaches and, in particular, extended and hybrid evolutionary approaches are among the most popular machine learning methods for deriving connectivity networks from gene expression data. The remainder of this paper comprises a section describing our neural-genetic approach in detail, followed by a results section which describes a set of experiments on, first, artificial data to determine accuracy on known interactions and, second, two real-world data sets. Conclusions and discussions follow these sections.

2 Method

The neural-genetic method adopted in this paper combines a GA with a supervized single-layer artificial neural network (ANN) to form a "hybrid" system, where one chromosome of the GA represents a small number of genes taken from the full set of genes and the ANN is used to check how well the expression values of these genes (input to the ANN) at one timepoint affect another gene's expression values (output from the ANN) at the subsequent timepoint, over a number of temporal datapoints. The GA consists of a population of such chromosomes, and each chromosome is evaluated by the ANN for its effect on every gene in the database as part of one generation. By choosing appropriate GA mutation and crossover operators to alter the genes making up chromosomes, and by selecting "good" chromosomes (as determined by the ANN) at each generation, chromosomes in future generations should determine sets of genes that have the greatest effect in the data. The method is described in more detail below.

The data used in the following experiments first comprises artificial data, constructed from Boolean values and using logic functions to determine the expression levels of genes at time t1 based on the activation of genes at time t0 [14]. Interactions can only occur between any two successive timepoints. The data is therefore comprised of the input data (expression values at time t0) and output data (expression values at time t1) and these pairs of data are used to train the neural-genetic hybrid. A similar approach has been used for the real-world data, in that a sequence of DNA array experiments is used to create a set of temporal gene expression values which are separated into pairs of input and output data. This approach inevitably ignores the fact that often the time between experiments is not necessarily regular and that the interactions between genes will be distributed over different time intervals. However, this is a factor of the experiments that are actually conducted. In many temporal gene expression experiments, measurements are taken frequently during the early part of the experiment (e.g., 1 hour, 2 hours, 3 hours, 4 hours) before larger gaps are introduced in the latter part (6 hours, 12 hours, 24 hours, 48 hours, etc.).

The aim is to derive a gene connectivity model that fits the data. The hypothesis is that the expression level of genes at a given time point is assumed to be a function of the expression levels at the previous time sample. The task for a "reverse engineering" algorithm given this aim and hypothesis is to relate the expression of genes between pairs of timesteps. If there are nine timepoints measured, there will be eight pairs of timesteps to relate independently of each other. Currently, there is no universally accepted method for combining gene expression values to give rise to the expression values of the regulated gene. However, a well-known approach is that of Weaver et al. [27], which sums the activation of the regulating genes and passes them through the sigmoid function to give rise to the next set of expression values. This is the approach used in this paper, which has the added bonus that the sigmoid function is one of the most popular used in ANNs and which, as will be shown later, allows the neural component of the algorithm to function well. The sigmoid function itself will only yield values from zero to one and, therefore, the actual output of the function is multiplied by the maximal gene expression value seen in the data. This, combined with a set of weights that connect the input and output genes, allows a direct comparison between desired gene expression values and actual output data from the ANN, for error calculation purposes.

The difficulty with what has been discussed so far is that, from a theoretical point of view, any gene can affect any other across a pair of timepoints and, more importantly, that combinations of genes can be involved in affecting other genes. Therefore the algorithm must determine plausible combinations of gene connections in the network. This again is achieved by the approach discussed in the following section, where the search is restricted to a small number (as a compromize to both biology and chaos theory this is set to 5) of genes for each network.

2.1 The Neural Genetic Approach

Previous experiments [12] have shown that GAs by themselves struggle to determine the correct excitatory and inhibitory values that link genes in a network using

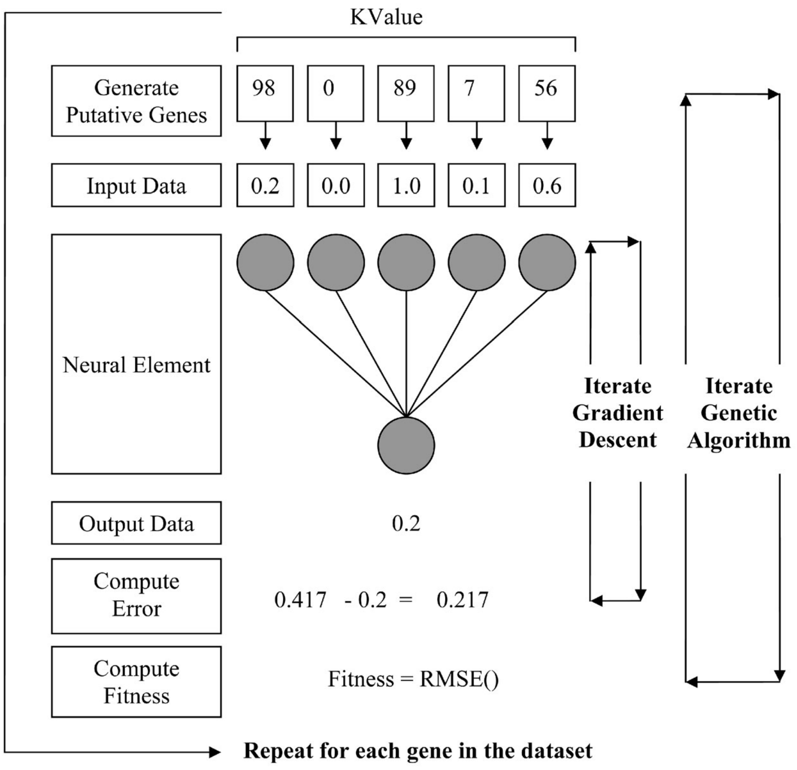

Fig. 1. Visual representation of the neural-genetic algorithm execution. Taken from *Intelligent Bioinformatics*, E. Keedwell and A. Narayanan (2005). Copyright John Wiley and Sons Limited. Reproduced with permission.

artificial data for which the connectivity is known. The neural-genetic approach described here uses a single-layer neural component using gradient descent to determine the weights of the ANN. This architecture is combined with an efficient GA for generating hypothetical links where the connectivity is restricted to a maximum of N. By combining the two methods of ANNs and GAs, the algorithm is able to make use of the connectivity restrictions imposed by the biology of the problem as well as the sigmoid activation function for the learning aspect of the problem.

The algorithm works as follows:

For each gene in the network:

1. Start with gene 1 as the "output" gene, gene_j.
2. Use the GA to generate combinations of N genes ($\mathrm{gene}_1, \mathrm{gene}_2, \ldots, \mathrm{gene}_n$) to affect the "output" gene. Each combination is a "chromosome" and the total set of combinations forms the initial population of chromosomes.
3. For each chromosome, select pairs of expression values for $\mathrm{gene}_1, \mathrm{gene}_2, \ldots, \mathrm{gene}_n$ and gene_j from the database of temporal pairs to form a training set of input-output pairs. That is, each pair in the training set will consist of the expression values for $\mathrm{gene}_1, \mathrm{gene}_2, \ldots, \mathrm{gene}_n$ from the t0 side of the temporal pair to be the input values to the ANN, and the expression value for gene_j from the t1 side of the temporal pair to be the target output for the ANN, up to time tN-1 and tN, respectively.
4. Run the gradient descent algorithm on this training data via the ANN to determine the weights between the "input" genes and the "output" gene until some stopping criterion is met.
5. Return the final ANN output error as a fitness value for that particular combination of $\mathrm{gene}_1, \mathrm{gene}_2, \ldots,$ gene_n, i.e., chromosome. Repeat Steps 3 and 4 for each chromosome.
6. Repeat Steps 2-5 as a normal GA run, using standard crossover and mutation operators on all chromosomes to vary the input genes chosen (but not the output gene).
7. When the GA stops, due to some stopping criterion, save the best chromosome and the weights on the connections linking the genes to the output node to a network structure.
8. Repeat Steps 1-7 for each gene in the network. That is, update gene_j (increment j).

When all eight steps are completed, a network structure will be derived which consists of a set of incoming connections for each gene in the data set. This process can also be seen graphically in Fig. 1.

As can be seen in Fig. 1, the chromosomes of the GA can consist of a number of "null" genes that do not correspond to genes in the data. This is to allow the GA to use a number of genes less than the n specified by the length of the chromosome, where these null genes have no bearing on the output of the sigmoid function. A further restriction is that there must be at least one non-null regulating gene for each regulated gene, and solutions that do not conform to this are heavily penalized, but not removed altogether. Due to the inclusion of null genes, each gene in the network will have a maximum of n incoming connections and a minimum of one. There is no restriction on the connections that a gene can possess beyond this number, i.e., a gene can regulate itself, and the number of times that a gene can

TABLE 1
Genetic Algorithm Parameter Settings

Parameter	Value, (rate)
Genetic Algorithm	Steady state, single objective
Crossover	One Point, (0.9)
Mutation	Random, (0.1)
Selector	Roulette Wheel
Population Size	10-50
Iterations	1000-1500

TABLE 2
ANN Parameter Settings

Parameter	Value, (rate)
Gradient descent	Standard/Step
Momentum	0.9
Beta	0.9
Weight Update	Online or Batch
Max Weight Value	-10, +10
Epochs	20-150

regulate another is entirely unrestricted. Therefore, the only restrictions placed on the connectivity of the network are those imposed by the biology of the problem at hand.

The parameters for the GA and the neural element can be seen in Tables 1 and 2, respectively. The genetic algorithm uses a standard binary representation with a one point crossover and standard random mutation. Due to the binary representation, the GA can generate values greater than the number of genes in the experiment. In this case, if the generated value is within 10 percent above the actual number of genes then it is considered as a "noncontributing" gene, otherwise, the values are "wrapped" around to the beginning. So, for 2,500 genes, a binary string of order 12 would be used giving 4,096 (i.e., 2^{12}) possible genes when converted to decimal, and 10 percent of 2,500 is 250. The gene number 2,653 would therefore yield a "noncontributing" gene, since 2,653 is below 2,500 + 250 = 2,750, whereas the gene number 3,563 would activate gene 813 (3,563 - 2,750). A limit is set for the maximum weights that can be learned for the ANN and the reasons for this are twofold. If the weights become too large, an overflow can occur with the power operator element of the sigmoid function, but secondly and more importantly, a limit on the weights restricts the extent to which the neural component can compensate for bad gene selection by the GA.

This method is capable of extracting gene networks with numerically plausible connections from real-world temporal gene expression data, as will be shown in the following section. An important feature of the algorithm is that it is also computationally efficient with respect to the number of genes that are present in the data and, so, can be applied to large-scale gene expression data on a standard PC.

2.2 Repeated Genetic Algorithm with Neural Network (RGANN)

The stochastic nature of both algorithms (the GA and the ANN) means that the same results are not expected from each run of the algorithm. Also, even though the number of connections in the network is biologically plausible, there are often still too many to visualize as a network to yield information in a digestible format for biologists. The combination of both these factors has led to the creation of a "repeated method" (RGANN). This simple method repeats the GA run five times for each "output" gene. If a gene connection is present in a certain number of these runs (typically, three to four of them), the connection is added into an aggregate gene network with weight values equal to the average of those in the repeated runs. This method, as it will subsequently be shown, is useful for discovering the most "significant" connections in the network in terms of repeated occurrences.

3 EXPERIMENTATION

The neural-genetic approach is tested on three data sets, as follows:

First, an artificial data set is created to ascertain the ability of the algorithm to "rediscover" the underlying rules and recreate the data that it has been trained on. The data is Boolean in nature and is created by a program written by the authors based on work undertaken by Liang et al. [14] who created several such data sets with which to test their algorithm REVEAL. The data is constructed by defining a number of Boolean genes (typically 10-20). The data is generated in pairs, with the input data being one member of the full enumeration of the possible individuals and the output data the outcome of the artificial rules when given that input data. Two example rules are:

$$\text{IF geneX} = 1 \text{ at time t0 THEN geneY} = 1 \text{ at time t1}$$
$$\text{IF geneX} = 1 \text{ AND geneY} = 1 \text{ at time t0 THEN geneZ} = 1 \text{ at time t1.}$$

A gene value at time t1 is determined to be 0 unless a rule changes it to 1. Rules can only apply to those genes that have the value 1, i.e., negative interactions such as "if a gene is off at time t0, then another gene is on at time t1" are not used. This process gives rise to 2^n data pairs, where n is the number of genes. The task here is to determine whether the approach is capable of discovering known associations between genes in the data, given the current pair-wise format.

Second, the rat spinal cord data set [28] is used. This data set consists of 112 gene measurements (using the reverse transcription-polymerase chain reaction or RT-PCR, method) of the central nervous system of a rat during its development from embryo through to adulthood. In all, nine measurements are made, which yields eight pairs of datapoints for the neural-genetic method. This data set represents a modest number of genes (112), but the aim is to ascertain whether the method can fit real-world data and produce gene networks that may be useful in future for

TABLE 3
Accuracy Results from Six Separate Runs of the Hybrid System on Liang Type Networks

K=2-4	**1**	**2**	**3**
Time (s)	18	13	9
Accuracy	96.875%	99.375%	100%
K=3-9	**1**	**2**	**3**
Time (s)	10	19	9
Accuracy	99.29%	97.30%	99.375%

Also noted is the time taken to complete the run to this accuracy.

TABLE 4
K = 2 Table Showing the Number Missing and Extra Connections between the Discovered Gene Network and that in the Artificial Network

K=2-4	**1**	**2**	**3**
Missing connections	1	0	1
Extra Connections	5	5	4

generating hypotheses for empirical testing. In addition to this, the small size of the data allows experimentation to be conducted repeatedly with various parameter settings.

The final data set is that of Spellman et al. [21], which consist of 2,468 genes of the yeast *Saccharomyces cerevisiae* measured at 7 minute intervals from 0 to 119 minutes, yielding 17 timepoints. The data was collected to determine those genes that are important in the cell cycle and it can be expected that many genes involved in the regulation of cell-cycle related activities are differentially expressed in this data set at different time-points. There are some missing values in this data, and interpolation was obtained by exploiting the expected smoothness of the time-courses as functions of time. If a value which required interpolation was at the end of the data, a line was drawn through the last two points and then a new value was added which this line predicts. The resulting data set represents a much sterner test for the neural-genetic approach as it has a large number of genes to deal with and the genes themselves have a large range of values.

The neural-genetic approach is tested with respect to the accuracy with which it can reproduce the data. For the artificial data, the evaluation is calculated as the number of bits correctly determined by the discovered network, and for the real-world data as the root mean squared error of the data produced by the network in comparison with the actual data recorded by DNA array experiments. Also for the artificial data, because the actual gene connectivity in the data is known, comparisons can be made between the actual connections used to create the data and those discovered by the algorithm. As the approach uses a different random seed for each gene, the performance of the algorithm on one run can be taken as an average of the performance over N such runs and repeated runs as used in many GA experiments are not required. In any case, the RGANN method uses a repeated approach which provides five distinct algorithm runs in any case.

4 Results

4.1 Artificial Data Experiments

Six data sets of 10 genes were created using logical rulesets. A full enumeration of 10 genes yields 1,024 individual records and the number of genes that can regulate a gene is determined to be between 2 and 4 for three of the data sets and 3 to 9 for the other three data sets. The ANN was run using the step function and for 20 epochs, while the GA was run for 100 generations with a population of 10. The step function was used because of the Boolean representation of the data.

The results in Table 3 were obtained by running the algorithm over these six data sets. The accuracy reported is the percentage of bits that are correct in the data set created by the gene network discovered by the algorithm. This measure was taken over the total number of 10,240 bits contained in the data set. The time in seconds is indicative of the time required to discover this network using a standard Pentium IV PC. As can be seen, the results show that the hybrid algorithm is capable of reproducing the data to a high level of accuracy. However, this result is to be expected, as there is no noise in the data set.

Table 4 shows the comparison between the first three data sets of Table 3 (top row). The artificial and discovered networks have been compared for the number of connections that they share. Any connections present in the artificial network, but not in the discovered one, are marked as missing connections, whereas any extraneous connections in the discovered network are marked as extra connections. The comparison has only been made for the data set with smaller k-value (2-4). This is because there are large numbers of connections in the more complex data set. This makes it difficult to make comparisons, since there are a number of permutations of these connections that can give rise to the same data. The testing of the algorithm on this artificial data set has shown that it is capable of discovering the correct interactions between genes when given temporal data that describes these interactions.

4.2 Rat Spinal Cord Data

The rat spinal cord data consists of 112 real-coded genes over nine time points, giving eight training examples. In the following experiments, the GA was run for 1,000 generations and a population of 50, and the ANN component was run using 100 epochs. These parameter settings are justified in subsequent experiments. As this is real-world data, the correct gene connectivity network is not available and, therefore, the accuracy of the network is determined by how well it fits the data. The data is reconstructed using the input values of the genes multiplied by the weight values and then passed through the normalized sigmoid function. Error is computed as the root mean squared error (RMSE) in the normal fashion and compared with the performance of a random network. The random network is generated by selecting at random, k (5) genes from the total set for each output gene and assigning to them random weights within the permitted range (-10 to +10). The network is constructed in this way and then executed on the relevant data set with the same transformation and normalization procedures as the discovered networks.

Fig. 2 shows a comparison of the results of running a random network with the same constraints as the neural-genetic algorithm on the problem, and three separate runs of the neural-genetic algorithm itself. Each run of the algorithm is conducted on a different training set, with a number of

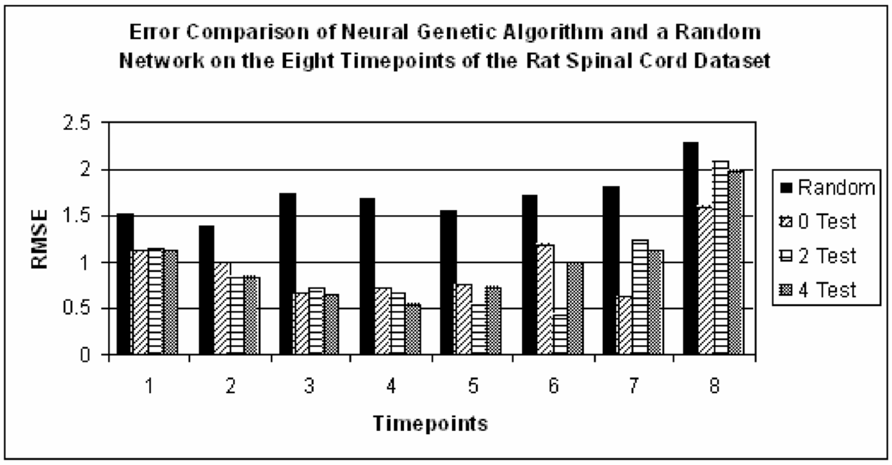

Fig. 2. Comparison of a random genetic network and three neural-genetic gene networks with a variety of testing timepoints.

timepoints held out for testing purposes. For instance, the "4 test" run was trained only on the first four timepoints of the data and tested on the final four. This approach shows that the neural-genetic approach derives a set of connections that fit the data with much higher accuracy than a random network with the same constraints (i.e., a sigmoid function activation normalized to the largest value of that gene). The error increases for later test timepoints, but not catastrophically. The error values are consistently below those of the random network value. An interesting point is that the final timepoint has the highest error for each of the neural-genetic methods and the random network. The data at the final timepoint relates to the expression levels when the rat is in a "mature" state as opposed to either the embryonic or postnatal stages. This appears to suggest that the expression levels (and perhaps underlying connectivity network) change markedly at this point.

The data used in the experiments was not normalized and is the raw data used in the original paper. While these experiments show the accuracy of the discovered models, which have at most 560 connections (or less than 5 percent of the total search space), they are still too large to visualize. The repeated approach delivers much smaller models that can be scrutinized by biologists for their biological plausibility. Repeated runs for the rat spinal cord data are seen in later sections.

The following experiments were undertaken to determine the sensitivity of the algorithm to the parameters of the GA and ANN. In these experiments, both the number of epochs and the number of generations of the algorithms were varied while recording the computation times and error rates for the training data (first four examples) and test data (the last four). The RMSE values in Figs. 3 and 4 should be compared with the average RMSE (1.57 for the first four examples, and 1.84 for the final four) for the random network.

Fig. 3 shows that the time complexity of the genetic algorithm increases linearly with respect to the number of generations, with 2,000 generations requiring some four times the computation of 500 generations. What is also interesting is that, generally speaking, the training error decreases with respect to the number of generations, whereas the testing data increases slightly. This indicates that a modest number of generations should give best performance, but with a linear increase in running time.

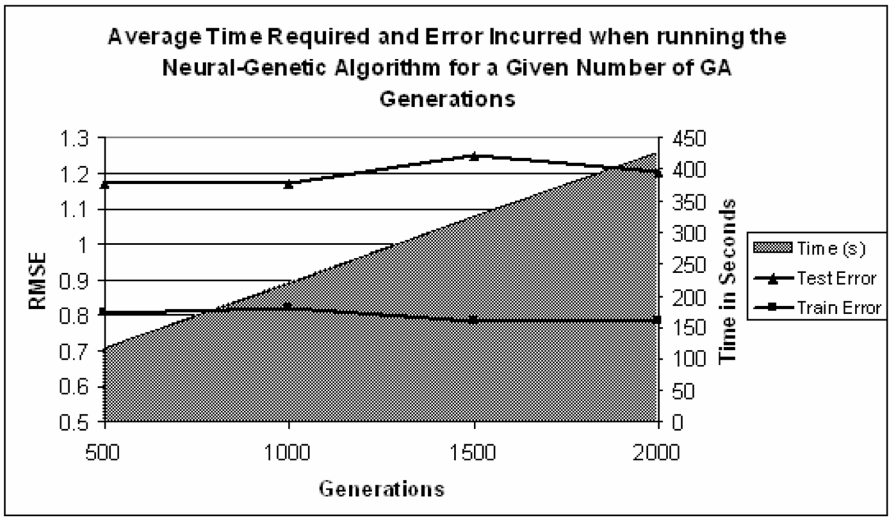

Fig. 3. Comparison of the average error incurred and time taken to compute an interaction network for the Rat Spinal Cord Data Set for a given number of GA generations.

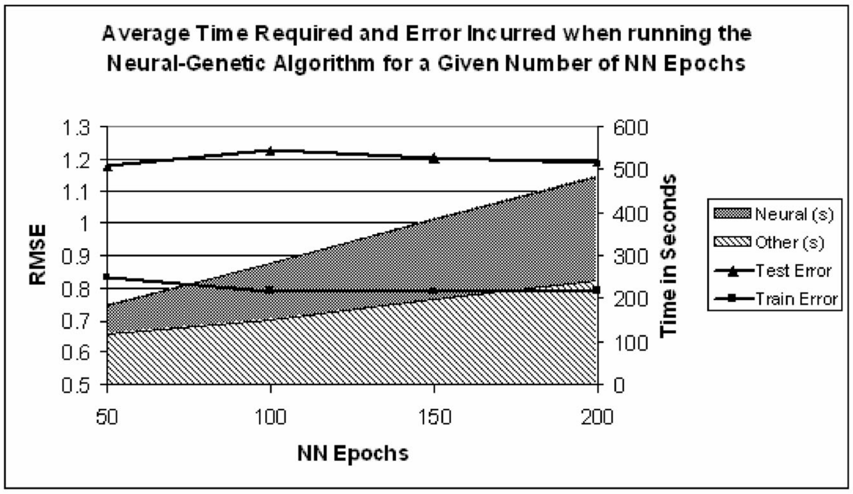

Fig. 4. Comparison of the average error incurred and time taken to compute a interaction network for the Rat Spinal Cord Data Set for a given number of NN epochs.

Fig. 4 shows a similar comparison to that portrayed in Fig. 3, but this time varying the number of neural network epochs used in the training procedure. The graph shows the total computation time split into the time required to run the neural component and that required to perform the remaining computation. The remaining computation may not be all due to the GA, since some CPU time will be required to update the screen and perform operating system procedures. The figures show that both components have a similar impact on the time required to run the algorithm, with around half of CPU time spent on the neural component, although this increases as expected with the number of epochs. This is accompanied by a commensurate decrease in the error seen on the training data, so it appears that a moderate set of values for the number of epochs is required.

These experiments therefore indicate that the fundamental parameter involved with the performance of the algorithm is the number of generations of the genetic algorithm. A subsequent experiment (not shown here) conducted using 1,000 generations of the genetic algorithm, but using random as opposed to neural-derived weights, showed the error to be 1.35, which is still some 0.31 higher than the combined algorithm. Therefore, it appears that while the number of generations specified for the genetic algorithm is the most important parameter, input from the neural component of the algorithm is required to achieve better error values.

4.3 Yeast Cell Cycle Data

The yeast cell cycle data set is much more complex than either of the two previous data sets, consisting of almost 2,500 genes and 17 pairs of data. The number of gene combinations increases significantly in comparison to the rat spinal cord data. The neural-genetic method was run on the data using 1,500 generations of the genetic algorithm and 150 epochs for the neural network. The increased computation is due to the vastly increased space of the potential network connections and incurs much longer running times.

Fig. 5 shows a comparison of the neural-genetic technique with a randomly generated network subjected to the same constraints as the algorithm itself. The performance is again compared between three runs of the algorithm while reserving an increasing amount of the data set for testing purposes. As can be seen, the algorithm comfortably outper-

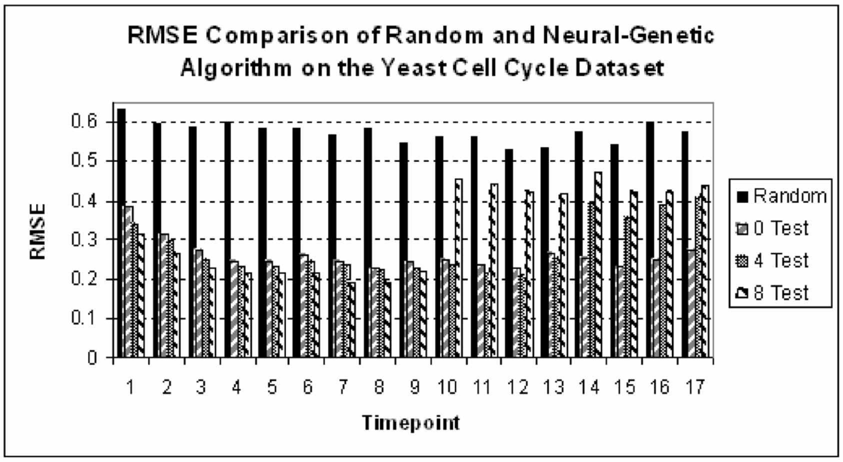

Fig. 5. Comparison of the neural-genetic technique with a random network on the yeast cell cycle data set.

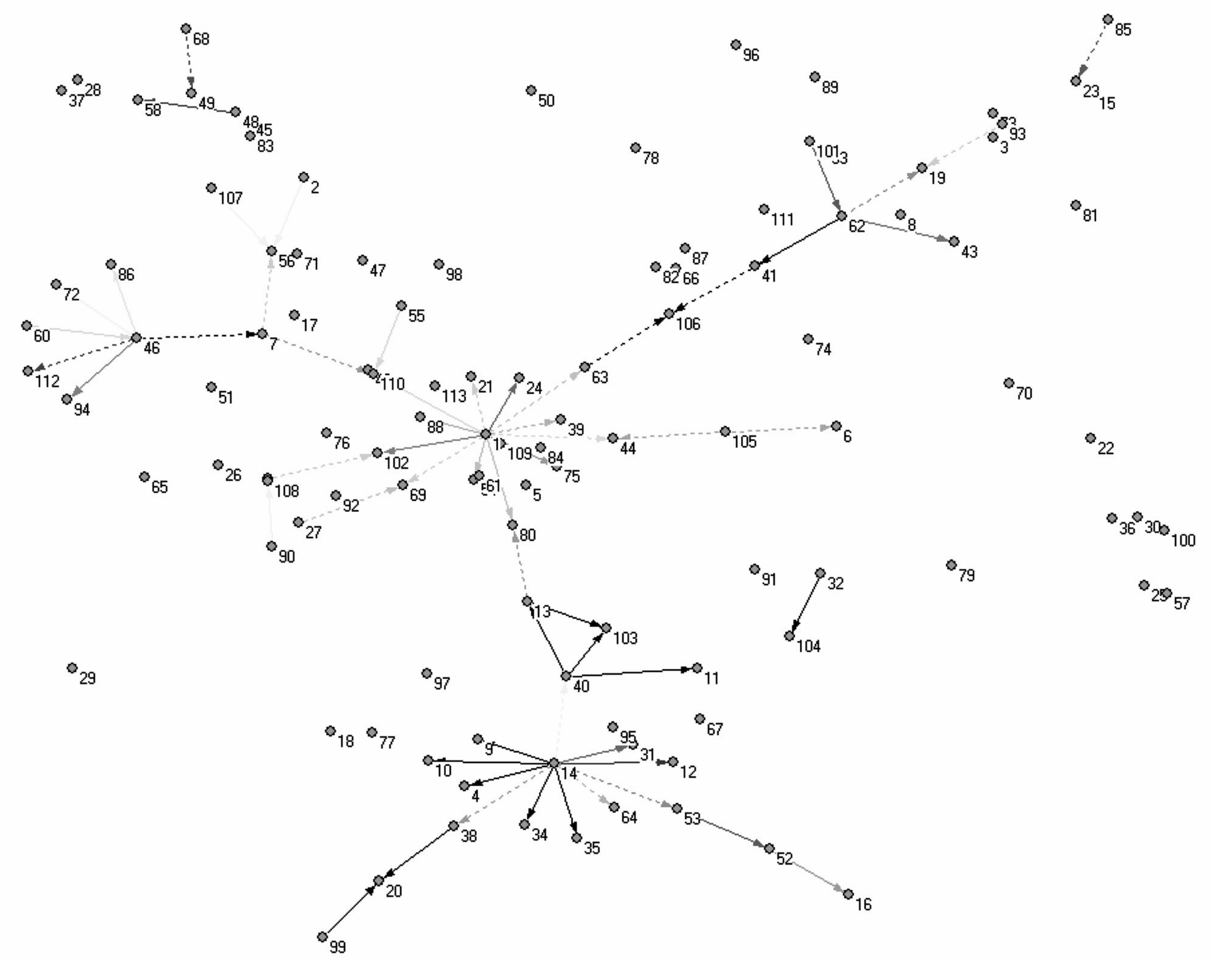

Fig. 6. Gene connectivity/interaction network for the rat spinal cord data with a threshold of 4.

forms the random network on the training data, but this advantage is considerably reduced when applied to the test data. This could be for a number of reasons. First, the eight-test data set performs badly here and this could be because almost half of the data is reserved for testing. Second, the datapoints at the end of the data may not be as representative of those at the start, given the temporal differences between successive time-points (equally spaced) that accumulate through the experiment. Finally, the algorithm could be overfitting the data (in the sense that it fits the noise as well as the patterns in the data through increased epochs, thereby producing poor test results on unseen data), which is a possibility considering the variables outnumber the records 156:1. Nevertheless, the test performance, which is consistently better than random even when the variable/record ratio is almost doubled, shows that this algorithm is capable of discovering potential underlying relationships on this much larger data set.

4.4 Repeated Neural-Genetic Experiments

The repeated algorithm was run on the rat spinal cord data using 1,000 genetic algorithm generations, 100 neural network epochs and repeated five times. These parameter settings were used as they were shown to give reasonable performance and running times in the earlier complexity experimentation. Figs. 6 and 7 show the networks that are discovered using this approach, with different thresholds for inclusion in the network. To allow easy visualization of the network, each gene is marked with a number and the corresponding gene names for these can be seen in Appendix 1 (which can be found on the CS Digital Library at http://csdl2.computer.org/persagen/DLPublication.jsp). The following figures are produced using a software package called Pajek.[1] Negative effect links are indicated by dashed lines, positive effect links by solid lines, and each line is shaded according to its weight, so minor interactions are shaded light grey and major interactions are coloured black. A legend is also provided in Table 5 to show the mapping between gene names and the numbers used in Figs. 6 and 7.

Fig. 6 shows the advantages of generating smaller gene networks, in that each of the paths in the network can be seen without being masked by other connections. Also, "hubs," "circuits," and pathways can be easily identified from a diagram such as this. An example of a hub is that of gene 14 (RATMOG-myelin/oligodendrocyte glycoprotein), which appears to regulate a whole host of other genes, and gene 1 (keratin_RNKER19), which also appears to have a large-scale effect. Laboratory experiments are required to

1. Go to http://vlado.fmf.uni-lj.si/pub/networks/pajek/ for more information.

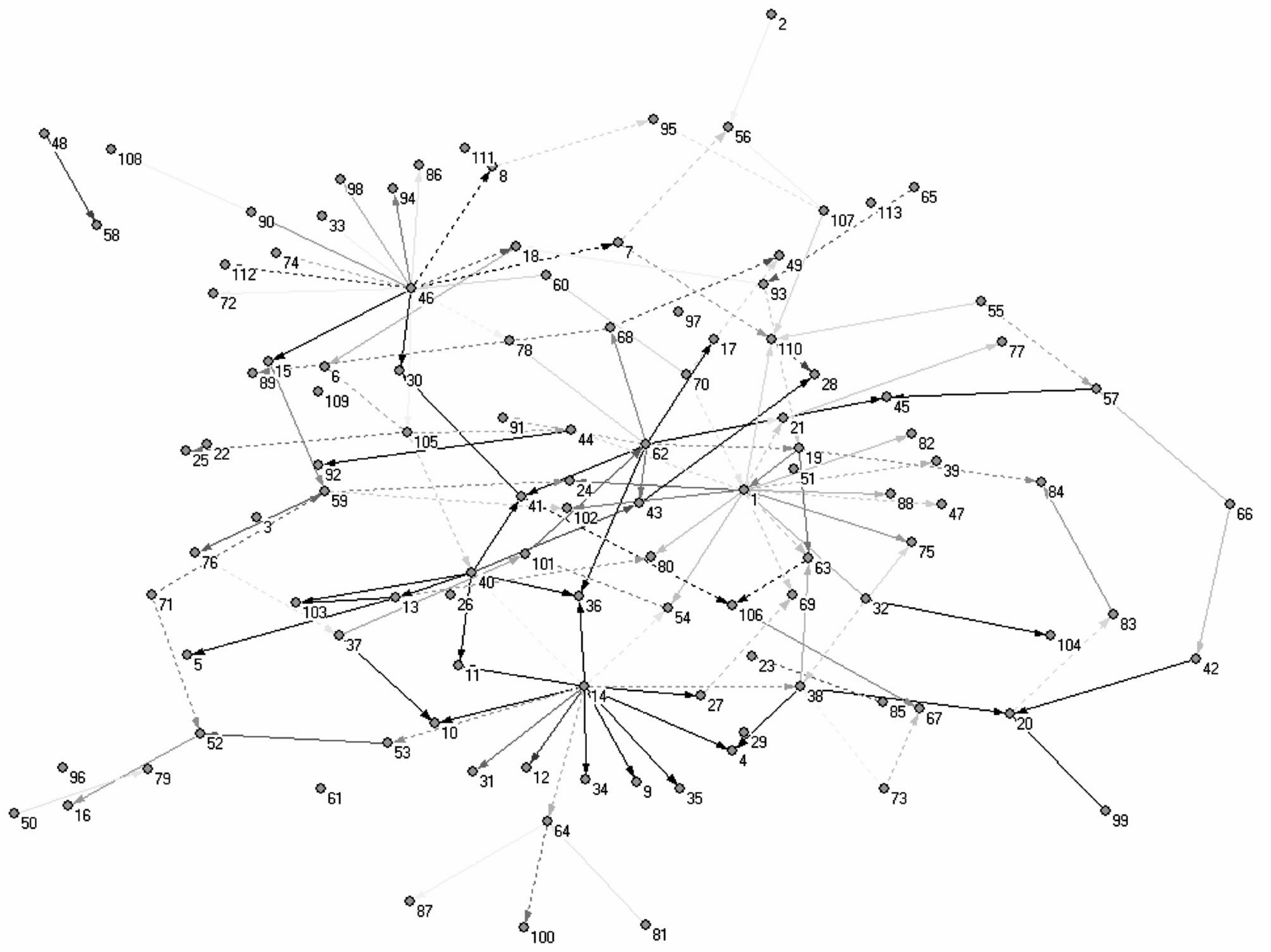

Fig. 7. Gene connectivity/interaction network for the rat spinal cord data with a threshold of 3.

test these *in silico* hypotheses. Fig. 7 shows the GA-NN process running on the same gene expression data but with the threshold of the link admittance being lowered from 4 to 3. It is easy to see how incomprehensible the model can become when there are so many interactions. This network has 138 connections and, while this is far fewer than the 560 that the normal algorithm will produce, the task of presenting and digesting this information is significantly more difficult.

In summary, this repeated method has been shown to discover interpretable gene connectivity structures from real-world gene expression data. There is a trade off between model size and accuracy, but the use of the repeated approach ensures that only "significant" connections are present in the final model. The sparse network shown in Fig. 6 shows the problem of reducing the number of connections so that the network can be easily viewed. Sixty-two connections are present in the network, but a number of genes have no connections. The network in Fig. 7 addresses this problem, with almost every gene affected to some extent, but the extra complexity makes interpreting this network more difficult.

5 Discussion

The discovery of gene connectivity/interaction networks from temporal expression data is one of the most pressing problems in computational biology. It is widely recognized that the causes of a variety of genetic-related phenomena, such as the cell cycle, rat spinal cord development or diseases such as cancer, are a product of complex interactions between genes over time rather than the activation of a handful of genes at one timepoint. The analysis of temporal gene expression data will become increasingly prevalent, an effect which will be compounded by cheaper and more accessible DNA array technology.

Currently, there are only a few algorithms that can extract putative regulatory networks from gene expression data alone. When assessing methods for gene network discovery, the size, type and number of sources of data must be taken into account. A number of approaches already exist which produce good results in highly restricted domains, such as with Boolean data and small numbers of genes. While these approaches can be theoretically interesting, the approach must be capable of handling noisy, floating point and high dimensional gene expression data if they are to be used by biologists. The neural-genetic approach described here has been shown to be effective on both artificial and real-world expression data on a per-gene basis and, most importantly, can discover networks in reasonable computational time without additional data. The fact that the GA is run for a fixed number of generations for each output gene means that its complexity increases linearly with each new output gene. However,

TABLE 5
Rat Spinal Cord Data Set Gene Legend

1	keratin_RNKER19	39	mGluR3_RATMGLURC	77	bFGF_RNFGFT
2	cellubrevin_s63830	40	mGluR4_RATMGLUR4B	78	aFGF_RNHBGF1
3	nestin_RATNESTIN	41	mGluR5_RATMGLUR	79	PDGFa_RNPDGFACP
4	MAP2_RATMAP2	42	mGluR6_RATMGLUR6.	80	PDGFb_RNPDGFBCP
5	GAP43_RATGAP43	43	mGluR7_RNU06832	81	EGFR_RATEGFR
6	L1_S55536	44	mGluR8_MMU17252	82	FGFR_RATFGFR1
7	NFL_RATNFL	45	NMDA1_RNMDA	83	PDGFR_RNPDGFRBE
8	NFM_RATNFM	46	NMDA2A_RATNMDA2A	84	TGFR_RATTGFBIIR
9	NFH_RATNFHPEP	47	NMDA2B_RATNMDA2B	85	Ins1_RNINS1
10	synaptophysin_RNSYN	48	NMDA2C_RATNMCA2C	86	Ins2_RNINS2
11	neno_RATENONS	49	NMDA2D_RNU08260	87	IGF_I_RATIGFIA
12	S100_beta_RATS100B	50	nAChRa2_RATNNAR	88	IGF_II_RATGFI2
13	GFAP_RNU03700	51	nAChRa3_RNACHRAR	89	InsR_RATINSAB
14	MOG_RATMOG	52	nAChRa4_RATNARAA	90	IGFR1_RATIGFI
15	GAD65_RATGAD65	53	nAChRa5_RATNACHRR	91	IGFR2_MMU04710
16	pre-GAD67_RATGAD67	54	nAChRa6_RATNARA6S	92	CRAF_RATRAFA
17	GAD67_RATGAD67	55	nAChRa7_RATNARAD	93	IP3R1_RATI145TR
18	G67I80/86_RATGAD67	56	nAChRd_RNZCRD1	94	IP3R2_RNITPR2R
19	G67I86_RATGAD67	57	nAChRe_RNACRE	95	IP3R3_RATIP3R3X
20	GAT1_RATGABAT	58	mAChR2_BOVMRM2SUB	96	cyclin_A_RATPCNA
21	ChAT_(*)	59	mAChR3_RATACHRMB	97	cyclin_B_RATCYCLNB
22	ACHE_S50879	60	mAChR4_RATACHRMD	98	H2AZ_RATHIS2AZ
23	ODC_RATODC	61	5HT1b_RAT5HT1BR	99	statin_RATPS1
24	TH_RATTOHA	62	5HT1c_RATSR1CA	100	cjun_RNRJG9
25	NOS_RRBNOS	63	5HT2_RATSR5HT2	101	cfos_RNCFOSR
26	GRa1_(#)	64	5HT3_MOUSE5HT3	102	Brm_(I_I)
27	GRa2_(Ý)	65	NGF_RNNGFB	103	TCP_(I_I)
28	GRa3_RNGABAA	66	NT3_RATHDNFNT	104	actin_RNAC01
29	GRa4_(§)	67	BDNF_rat_BDNF_D	105	SOD_RNSODR
30	GRa5_(#)	68	CNTF_RNCNTF	106	CCO1_RATMTCYTOC
31	GRb1_RATGARB1	69	trk_RATTRKPREC	107	CCO2_RATMTCYTOC
32	GRb2_RATGARB2	70	trkB_RATTRKB1	108	SC1_RNU19135
33	GRb3_RATGARB3	71	trkC_RATTRKCN3	109	SC2_RNU19136
34	GRg1_RNGABA	72	CNTFR_S54212	110	SC6_RNU19140
35	GRg2_(#)	73	MK2_MUSMK	111	SC7_RNU19141
36	GRg3_RATGABAA	74	PTN_RATHBGAM	112	DD63.2_(I_I)
37	mGluR1_RATGPCGR	75	GDNF_RATGDNF		
38	mGluR2_RATMGLURB	76	EGF_RATEPGF		

this does also increase the number of genes that the GA can select from, which in turn could require more GA generations to achieve similar accuracy to that shown on a smaller data set. The rat-spinal cord data experiments required approximately 2 minutes of computation each on a standard Pentium IV PC. The repeated runs required more time, approximately 45 minutes, and the yeast cell cycle runs required between 2 and 6 hours. These times give little indication of the actual complexity of the algorithm, but it does show that results on experimental data can be discovered quickly using modest hardware. In addition to this, if computation time is at a premium, small segments of the network can be generated according to the interest of the biologist. This "ondemand" discovery option would allow the user to select an output gene of interest and discover the connections associated solely with that gene. This approach would yield solutions quickly, and visualisation would not be a problem due to the small number of connections present in the resulting network.

6 Conclusions

A neural-genetic approach to the discovery of gene networks from gene expression data has been described. On both artificial and real-world data, the algorithm has been shown to be able to accurately fit the data on which it was trained and, in the case of the artificial data, those connections that were present in the data. In addition to this, the algorithm has been shown to discover models from a number of training examples and reproduce test data examples with good accuracy. One of the problems of gene network discovery from real-world data is the lack of definitive network models to compare the approach against, but the fact that the approach is able to reproduce test data points is significant in this field. Also, it has been shown that a repeated approach to this problem can yield models that have a flexible number of connections, but that the most significant connections in the model are retained for the more sparse solutions. Finally, since this algorithm has been previously shown to also discover models from classification experiments, the neural-genetic approach can be considered a candidate multipurpose tool for gene expression data analysis.

REFERENCES

[1] S. Ando and H. Iba, "Identifying the Gene Regulatory Network by Real-Coded, Variable Length and Multiple-Stage GA," http://www.citeseer.nj.nec.com/333262.html, 2000.

[2] S. Ando and H. Iba, "Inference of Gene Regulatory Model by Genetic Algorithms," *Proc. IEEE Congress on Evolutionary Computation,* pp. 712-719, 2001.

[3] S. Ando and H. Iba, "The Matrix Modeling of Gene Regulatory Networks—Reverse Engineering by Genetic Algorithms," *Proc. Atlantic Symp. Computational Biology, and Genome Information Systems and Technology,* 2001.

[4] Z. Bar-Joseph, G. Gerber, T. Lee, N. Rinaldi, J. Yoo, F. Robert, B. Gordon, E. Fraenkel, T. Jaakkola, R. Young, and D. Gifford, "Computational Discovery of Gene Modules and Regulatory Networks," *Nature Biotechnology,* vol. 21, no. 11, pp. 1337-1342, 2003.

[5] Z. Bar-Joseph, "Analyzing Time Series Gene Expression Data," *Bioinformatics,* vol. 20, no. 16, pp. 2493-2503, 2004.

[6] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering Gene Expression Patterns," *J. Computational Biology,* vol. 6, pp. 281-297, 1999.

[7] T. Chen, V. Filkov, and S.S. Skiena, "Identifying Gene Regulatory Networks from Experimental Data," *Proc. ACM-SIGAT the Third Ann. Int'l Conf. Computational Molecular Biology (RECOMB99),* pp. 94-103, 1999.

[8] A. Califano, S. Stolovitzky, and Y. Tu, "Analysis of Gene Expression Microarrays for Phenotype Classification," http://citeseer.nj.nec.com/califano00analysis.html, 2000.

[9] P. D'haeseleer and X. Wen, "Mining the Gene Expression Matrix: Inferring Gene Relationships from Large Scale Gene Expression Data," *Bioinformatics,* vol. 16, no. 8, pp. 707-726, 2000.

[10] S. Kauffman, *At Home in the Universe: The Search for Laws of Self-Organization and Complexity.* Penguin Books, 1996.

[11] E.C. Keedwell, "Knowledge Discovery from Gene Expression Data Using Neural-Genetic Models," PhD thesis, Univ. of Exeter, UK, http://www.ex.ac.uk/eckeedwe, 2003.

[12] E.C. Keedwell and A. Narayanan, "Genetic Algorithms for Gene Expression Analysis," *Applications of Evolutionary Computing LNCS 2611,* 2003.

[13] P. Koduru, S. Das, S. Welch, and J.L. Row, "Fuzzy Dominance Based Multi-Objective GA-Simplex Hybrid Algorithms Applied to Gene Network Models," *Proc. Genetic and Evolutionary Computation Conf. (GECCO '04),* 2004.

[14] S. Liang, S. Fuhrman, and R. Somogyi, "REVEAL, A General Reverse Engineering Algorithm for Inference of Genetic Network Architectures," *Proc. Pacific Symp. Biocomputing 3,* pp. 18-29, 1998.

[15] Y. Maki, D. Tominaga, M. Okamoto, S. Watanabe, and Y. Eguchi, "Development of a System for the Inference of Large Scale Genetic Networks," *Proc. Pacific Symp. Biocomputing 6,* pp. 446-458, 2001.

[16] G.S. Michaels and D.B. Carr, "Cluster Analysis and Data Visualisation of Large-Scale Gene Expression Data," *Proc. Pacific Symp. Biocomputing 3,* pp. 42-53, 1998.

[17] A. Narayanan, E. Keedwell, and B. Olsson, "ArtificialIntelligence Techniques for Bioinformatics," *Applied Bioinformatics,* vol. 1, no. 4, pp. 191-222, 2003.

[18] A. Narayanan, E.C. Keedwell, J. Gamalielsson, and S. Tatineni, "Single Layer Artificial Neural Networks for Gene Expression Analysis," *Proc. Neurocomputing Conf. 61,* pp. 217-240, 2004.

[19] J.A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *Computer J.,* vol. 7, no. 4, pp. 308-313, 1965.

[20] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman, "Module Networks: Identifying Regulatory Modules and Their Condition-Specific Regulators from Gene Expression Data," *Nature Genetics,* vol. 34, pp. 166-176, 2003.

[21] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher, "Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization," *Molecular and Cell Biology,* vol. 9, pp. 3273-3297, 1998.

[22] C. Spieth, F. Streichert, N. Speer, and A. Zell, "Optimizing Topology and Parameters of Gene Regulatory Network Models from Time Series Experiments," *Proc. Genetic and Evolutionary Computation Conf. (GECCO '04),* 2004.

[23] P. Spirtes, C. Glymour, R. Scheines, S. Kauffmann, V. Aimale, and F. Wimberly, "Constructing Bayesian Network Models of Gene Expression Networks from Microarray Data," *Proc. Atlantic Symp. Computational Biology, Genome Information Systems and Technology,* 2000.

[24] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E.S. Lander, and T.R. Golub, "Interpreting Patterns of Gene Expression with Self-Organising Maps: Methods and Applications to Hematopoietic Differentiation," *Proc. Nat'l Academy of Sciences,* vol. 96, pp. 2907-2912, 1999.

[25] D. Thieffry, A.M. Huerta, E. Perez-Rueda, and J. Collado-Vides, "From Specific Gene Regulation to Genomic Networks: A Global Analysis of Transcriptional Regulation in Escherichia Coli," *BioEssays,* vol. 20, pp. 433-440, 1998.

[26] E.P. van Someren, L.F.A. Wessels, M.J.T. Reinders, and E. Backer, "Robust Genetic Network Modeling by Adding Noisy Data," *Proc. 2001 IEEE-EURASIP Workshop Nonlinear Signal and Image Processing (NSIP01),* June 2001.

[27] D.C. Weaver, C.T. Workman, and G.D. Stormo, "Modelling Regulatory Networks with Weight Matrices," *Proc. Pacific Symp. Biocomputing,* pp. 112-123, 1999.

[28] X. Wen, S. Furhmann, G.S. Michaels, D.B. Carr, S. Smith, J.L. Barker, and R. Somogyi, "Large-Scale Temporal Gene Expression Mapping of Central Nervous System Development," *Proc. Nat'l Academy of Sciences 95,* pp. 334-339, 1998.

Edward Keedwell received a PhD degree in computer science (bioinformatics) from the University of Exeter in 2003, having previously graduated with a degree in cognitive science (BSc with honors) in 1998. Dr. Keedwell's interests are in the application of artificial intelligence and evolutionary computation techniques to optimization problems in engineering as well as to bioinformatics problems. He has written one book, a number of journal articles, and conference publications in the fields of computer science, bioinformatics, and engineering. He is currently a research fellow at the University of Exeter.

Ajit Narayanan received a degree in communication science and linguistics from the University of Aston (1973), and the PhD degree in philosophy from the University of Exeter (1976). He took up a lectureship in computer science at the University of Exeter soon after the department was formed (1980). He is a professor of artificial intelligence at the University of Exeter, where he has been located for the last 25 years. His interests are in artificial intelligence, cognitive science, and mind/brain issues. He has written several books and numerous articles on artificial intelligence and, more recently, has become the Director of Bioinformatics at Exeter. His current research interests focus on the application of artificial intelligence techniques to problems in bioinformatics and systems biology.