The Irish Accounting Review (2002), Vol. 9, No. 1, 39–56

NEURAL NETWORK DESIGN USING AN EVOLUTIONARY ALGORITHM: A FINANCIAL EXAMPLE

Tony Brabazon

University College Dublin

ABSTRACT

Applications of fully connected, feedforward, neural networks in a business and finance domain are generally developed through a trial and error approach, guided by heuristics. This process is time consuming and there is no guarantee that the final network structure is in any sense optimal. Utilising financial time series data drawn from the FTSE 100 index, this paper demonstrates how an evolutionary algorithm, the genetic algorithm, can be applied in order to automate the process of determining network topology. The results from the evolved network are contrasted with those generated by a standard neural network modelling process. It is found that the hybrid genetic algorithm/neural network outperforms the fully connected neural network developed using a standard modelling process both in terms of minimising predictive error and in terms of conserving degrees of freedom. This suggests that the application of fully connected, feedforward neural networks is not always appropriate and may result in network "bloat".

INTRODUCTION

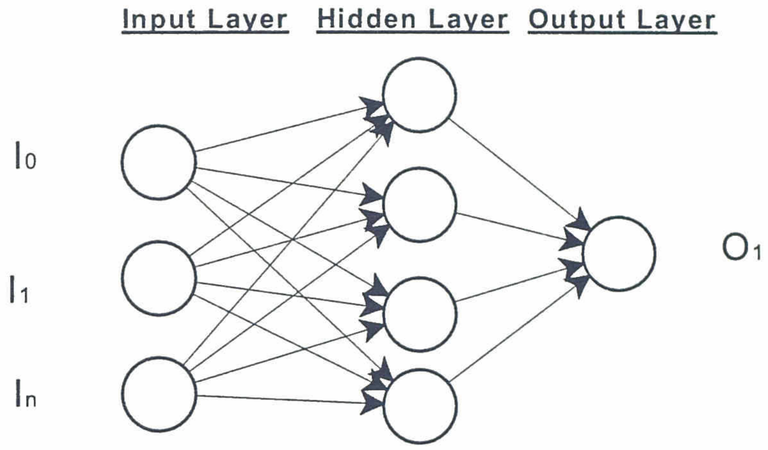
The objective of this paper is to demonstrate, by means of a case study, how an evolutionary algorithm, the genetic algorithm (GA), can be utilised to develop both the connection structure and connection weights in a feedforward, neural network (NN) model. The paper examines the predictive quality of the resulting hybrid GA/NN model by comparing it with a benchmark fully connected, feedforward NN model developed using the backpropagation training algorithm. A series of fundamental and technical data drawn from the FTSE 100 index is utilised as a test bed and the constructed models attempt to predict the five day percentage change in the value of this index.

While numerous studies have applied a NN methodology in order to predict market index values, there is a limited academic literature (Deboeck, 1994; Neely, Weller and Dittmar, 1997; Varetto, 1998; Allen and Karjalainen, 1999; Arifovic and Gencay, 2001) in the finance or general business domain that utilises evolutionary modelling approaches either on a stand-alone basis or in combination with NN methodologies. The most notable study applying evolutionary modelling to index prediction was undertaken by Allen and Karjalainen (1999). This applied an evolutionary approach to uncover technical trading rules for the S&P 500 but did not combine NN and GA methodologies.

Artificial Neural Networks

Artificial neural networks have been applied to many problems in both finance and general business domains (Wong, Lai and Lam, 2000). Artificial neural networks are a mathematical modelling methodology whose inspiration was drawn from an examination of the workings of biological neurons. NNs are generally used either to construct a model linking a set of inputs and outputs or to cluster data. A wide variety of NN architectures and training algorithms exist. The most commonly applied NN structures consist of a multi-layer, feedforward architecture and are developed using a backpropagation training algorithm. Basic multi-layer feedforward networks usually consist of three layers of nodes (Gurney, 1997). The first, or "input", layer serves as a holding layer for the data inputs to the model. This layer is connected to one or more "hidden" (internal) layers and nodes in this layer are connected in turn to an "output" layer, which represents the output from the model (**Figure 1**). The architecture is described as feedforward as the node-arc structure flows in one direction only, from input to output node(s). Each of the connections (arcs) between the nodes has an associated weight. The value passing along an arc is modified by multiplying it by this weight before reaching the next node. The weight serves to amplify or dampen the strength of a signal (value) passing along an arc. The nodes in the hidden and output layers each perform a simple computing task. Typically, the "processing" carried out at each of these nodes consists of passing the sum of the weighed inputs to that node through a non-linear (or "activation") function. Intuitively, feedforward NNs represent a non-linear, semi-parametric, regression model (Brown, Goetzmann and Kumar, 1998).

Figure 1: Three Layer, Feedforward NN



During model development the NN is constructed (trained) using pre-existing input/output data vectors and the object is to determine the arc weights that produce the closest mapping between the model's output and the actual known outputs for the data set. The error between the network's predicted outputs and desired (correct) outputs is propagated backwards (the backpropagation algorithm) through the network, altering the values of connection weights. Intuitively, the connection weights are similar in nature to coefficients in a regression model. This step is performed iteratively until the chosen error measure associated with a network structure reaches a stable minimum. The behaviour of neural networks (how they map inputs to output(s)) is influenced primarily by the form of processing that takes place at hidden and output layer nodes, how the nodes are interconnected and the weights associated with these interconnections.

Structure of Paper

This contribution is organised as follows. The next section provides an overview of the GA and discusses how it can be combined with a NN methodology. Following this, the model development process adopted is described and the results of the hybrid GA/NN model are compared with those from a standard NN modelling process. Finally, a number of conclusions are drawn. An appendix is included which provides a discussion of landscape theory. Landscape theory is relevant in any modelling scenario as it provides a conceptual framework for discussing and evaluating search (modelling) heuristics.

GENETIC ALGORITHM

Although the development of the GA dates from the 1960s they were first brought to the attention of a wide audience by Holland (1975). GAs represent a mathematical search heuristic, whose inspiration is drawn from biological evolution. A GA applies a pseudo-Darwinian process to evolve good solutions to problems. In biology, an organism's genotype contains the results (memory) of past evolutionary learning and the benefits of this learning are disseminated through a population by means of natural (differential) selection. The source of variation upon which natural selection acts arises through mutations and other genetic mechanisms that operate during reproduction.

A GA commences by randomly creating a population of possible solutions to a given problem. These solutions may be as diverse as a set of rules or a series of coefficient values. Although many variants of the GA exist (Goldberg, 1989), implementations typically represent individual components of solutions as a series of bits (0,1), which are combined to provide a binary string representation of the solution. This representation can be recast in a genetic metaphor whereby the individual components of a solution represent genes and the overall, encoded, solution represents a chromosome. Once the initial population of proposed solutions has been (perhaps randomly) formed and evaluated using a problem-specific fitness (objective) function, a reproductive process is applied in which better quality solutions have a higher chance of being selected for propagation of their components/bits/genes into the next generation of candidate solutions. Over a series of generations, components of the better solutions, in terms of the defined fitness function, tend to flourish and components of the poorer solutions tend to disappear. Differential selection and the reproductive process provide the engine for this search, biasing the search towards high-quality existing solutions. In a basic implementation of a GA, the reproductive process is governed by two genetic operators, crossover and mutation (Mitchell, 1996). The crossover operator takes two members of the population of proto-solutions and swaps component parts of each in order to create two new members of the population. As an example, suppose two parent proto-solutions are represented by the following binary strings, (0001 1101) and (1100 1100). If a single point crossover is applied after the fourth bit, the resulting child proto-solutions could be (0001 1100) and (1100 1101). The mutation operator causes small random changes in one or more of the genes of a child solution. In a binary representation of a proto-solution, a 0 may mutate to a 1 or a 1 to a 0.

An outline of a standard GA is as follows:

1. Construct an initial population, possibly randomly, of ***n*** proto-solutions (candidate solutions to a problem)
2. Calculate the fitness of each proto-solution in this population
3. Select, with replacement, a pair of proto-solutions from the current population, where the probability of selection is an increasing function of the fitness of an individual proto-solution
4. With a probability p_{cross}, perform a crossover process on the selected "parent" proto-solutions to produce two "child" proto-solutions
5. Apply a mutation process, with probability p_{mut}, to each gene locus of the two child proto-solutions
6. Store the child proto-solutions
7. Repeat steps 3–6 until ***n*** child proto-solutions have been created. Then replace the old population with the new population. This constitutes a "generation"
8. Go to step 2.

The process is iterated until stopped, either after a set time or number of generations or when the fitness of proto-solutions attains a required level.

The GA provides a number of benefits when applied in an optimisation setting. The computational power of the GA results from its explicit and implicit parallel processing capabilities. The explicit parallelisation stems from the maintenance of a population of potential solutions, rather than a single potential solution at each stage of the algorithm. The implicit parallel processing capabilities are proposed to arise due to the Schema Theorem (Holland, 1975) which demonstrates that under general conditions, in the presence of differential selection, crossover and mutation, compact clusters of genes that provide above-average fitness will increase their density in the population between each generation. In contrast to the parallel processing capabilities of a GA, hill climbing and gradient descent optimisation methods such as commonly applied in backpropagation are local search methods. The parallel nature of a GA search process makes it less vulnerable to local optima. Despite the good properties of the GA they, like all non-linear optimisation techniques, are subject to limitations. There is no guarantee that an optimal solution will be found in finite search time (Fogel, 2000). Progress towards better solutions may be intermittent rather than gradual.

Combining GA and NN Methodologies

GA and NN methodologies naturally complement each other when applied to modelling. NNs develop a model (or hypothesis) linking input and output data, whereas a GA can be used to determine high-quality parameters for a given model structure. Hence, a hybrid GA/NN methodology contains both hypothesis generation and hypothesis optimisation components. There are several ways that a GA can be combined with a NN methodology. The first possibility is to use the GA to uncover good quality model inputs. In a time series model, this could correspond to uncovering good quality lag periods. A second use of the GA is to evolve the internal structure of a network. In a multi-layer feedforward network, this could correspond to determining the number of hidden nodes, appropriate transfer functions at each node, and the connection structure between the input and the hidden node layers. A GA can also be used to evolve the connection weights of a NN. This approach removes restrictions on error functions, such as the requirement under the backpropagation algorithm that the error function has a continuous derivative. When using the GA to evolve NN structures, each of the ***n*** binary strings in the population represents a NN structure. For example, if a GA is utilised solely to optimise the ***m*** weights in a pre-determined NN structure, every binary string will contain ***m*** components, each corresponding to a real-valued individual weight. To evaluate the "fitness" of a given member of the population (binary string), the string is decoded into a discrete NN structure using these weights, and the prediction error of that NN is determined by passing the training data through the network. This prediction error provides a measure of the NN's fitness and thereby determines the likelihood that a portion of the underlying binary string will be passed on to the next generation. Over successive generations, the population of binary strings corresponding to the weight vectors will tend to improve in quality.

To date, few studies combining GAs and NNs in a finance or general business domain have been reported. However, a small number of studies that have compared the efficiency of a GA versus the backpropagation algorithm as a means of training a multi-layer feedforward network have been undertaken. Sexton and Gupta (2000) found that a GA outperformed the backpropagation algorithm for training NNs on chaotic time series data. Sexton and Dorsey (2000) reported that a GA-trained multi-layer feedforward network outperformed a backpropagation-trained network on a collection of ten test data sets used for classification purposes. This paper contributes to extant literature by providing a case study demonstrating how these methodologies can be combined in a financial setting and by contrasting the performance of a GA trained NN with that of a NN trained using the backpropagation algorithm.

METHODOLOGY

In the construction of the various models, this paper utilises a combination of technical and fundamental data, drawn from the period 1/1/92 to 31/12/96, which may have information content for predicting the percentage five-day change in the value of the FTSE 100 market index. Although the performance of a NN or any other model depends critically on selecting the most appropriate input variables, the focus of this paper is not on developing an optimal input set but rather on comparing alternative model construction methodologies. This paper does not explicitly attempt to optimise model inputs but rather uses them as a test bed for the modelling processes. Despite this, the inputs have been subject to an extensive selection procedure. The rationale for the choice of inputs and a more detailed description of the selection procedure is provided in Brabazon (2001). A brief description follows.

The inputs used in the model were selected from a range of technical, fundamental and intermarket data suggested in prior literature (Lakonishok and LeBaron, 1992; Chan, Jegadeesh and Lakonishok, 1996; Dissanaike, 1997; Brock and Cheng, 1998; Murphy, 1999). Initially, a range of technical indicators were selected. These included a variety of moving averages, relative strength indicators, oscillators and lagged index changes. In order to select the subset of technical indicators included in the final data set, several data selection tools were employed, including correlation analysis, regression models and the construction of preliminary neural network models. A similar selection process was applied to select the final intermarket and fundamental indicators. After these steps, the following 10 inputs were selected:

- 5-day lagged percentage change in the value of the FTSE 100 index
- 20-day lagged percentage change in the value of the FTSE 100 index
- Ratio of the 10 v 5-day moving average of the value of the FTSE 100 index
- Ratio of the 20 v 10-day moving average of the value of the FTSE 100 index
- Bank of England sterling index
- S&P $500_{(t-1)}$ composite index
- LIBOR 1-month deposit rate
- LIBOR 1-year deposit rate
- Aluminium ($ per tonne)
- Oil ($ per barrel).

Model Construction

In constructing the NN models for the purposes of this study, several factors were held constant across all models. The same data set was used in each case for training and out-of-sample validation. The network structure was fixed as a three layer, feedforward network with a hyperbolic tangent activation function. This reduced the benchmark NN development process to a decision regarding the number of nodes in the hidden layer. In developing the benchmark model, the SPSS – *Neural Connections* software package was used.

In the hybrid GA/NN model, both the number of connections between the input/hidden and the hidden/output layers, as well as the values of connection weights, were determined using the evolutionary algorithm. Implicitly, by allowing the GA to determine which connections to use, the process also determines the number of hidden nodes in the resulting NN and which inputs the model uses. As both the connection and weight values are evolved, the bit-strings representing each individual

NN structure in the population contain the information required to represent both the weights attached to each connection in a network and a binary indicator to indicate whether an individual connection is utilised. Initially, a population of 150 bit-strings was randomly generated. An iterative evolutionary process was then applied to this starting population. In order to select pairs of model structures for reproduction, a rank order procedure was adopted. This method ranks all candidate solutions such that $Rank_i > Rank_j$ for $f(i) > f(j)$, where f represents fitness, as measured by a root mean square error (RMSE) criterion. A selection function, derived from a negative exponential function, was used to calculate $p(i)$, which represents the probability that solution (bit-string) i is selected for reproduction at the end of a generation. Crossover determines degree of linkage between parent solutions and their offspring. In this study a uniform crossover operator is applied. In uniform crossover, each gene of the offspring is selected randomly from the corresponding genes (components) of the parent solutions. Unlike single or double point crossover, uniform crossover can generate any schema possible from the parents (Bauer 1994, p. 95). The mutation operator determines the likelihood that the value of an individual gene that comprises part of a problem solution (bit-string) will be replaced with a value randomly chosen from the parameter space. The mutation rate can vary between 0 (no mutation) and 1. A value of 1 effectively destroys the impact of the crossover operator. In this study, the mutation rate is varied dynamically as the genetic algorithm unfolds. If the population of solutions converges to a limited range of solutions, the crossover operator reduces in search effectiveness. To prevent premature convergence of solutions, the mutation operator increases in value as convergence is detected. The evolutionary process was continued until manually terminated. The hybrid GA/NN models were constructed using the *Evolver Development Kit*, a software development tool authored by Palisade Corporation.

RESULTS AND DISCUSSION

This section describes the results from each model and then provides a discussion of these results.

Fully Connected, Feedforward NN

The final NN utilised had an 11-7-1 structure, comprising the 10 model inputs and a bias node in both the input and hidden layers, resulting in a model with 73 weights. The results from the model are shown in **Table 1**.

Table 1: Fully Connected, Feedforward NN Model

	Training Data	*Test Set 1*	*Test Set 2*	*Test Set 3*
RMSE	0.1831	0.1113	0.1573	0.1585
R	0.2910	0.2937	0.2899	0.0945
R^2	0.0846	0.0862	0.0840	0.0089

The results exhibit a general consistency across each of the three "goodness of fit" measures calculated. The training/validation period consists of 918 data vectors drawn from the period 1/1/92 to 7/7/95. The out-of-sample data set is split into three divisions of similar size to determine whether predictive accuracy declines markedly as the length of time from the training/validation data set increases. The first out-of-sample data set consists of 125 data vectors drawn from the period 10/7/95 to 31/12/95. The second and third data sets are comprised of 130 and 132 data vectors respectively and are drawn from the periods 1/1/96 to 28/6/96 and 1/7/96 to 31/12/96.

Out-of-sample predictive performance in the first two test data sets is generally consistent with the model's performance on the training/validation data set, but declines in the third period, possibly indicating that the NN model requires retraining. Although care must be exercised in generalising from the results of any single sample, the results do not clearly support a hypothesis that a NN model is incapable of detecting structure underlying the behaviour of a stock market index.

Hybrid GA/NN Model

The GA/NN modelling approach does not construct a single network but rather constructs a population of network structures. An initial population of 150 networks was generated randomly, and the evolutionary algorithm was run for 1,500 generations. This resulted in the examination of 225,000 distinct NN structures. The search process was manually terminated at this point. An examination of the final population of evolved NNs showed that a substantial number had achieved a similar fitness level (minimisation of RMSE). The in-sample and out-of-sample results from the network with highest fitness are shown in **Table 2**.

Table 2: Best Hybrid GA/NN Model

	Training Data	*Test Set 1*	*Test Set 2*	*Test Set 3*
RMSE	0.1756	0.1086	0.1342	0.1331
R	0.4022	0.4453	0.3239	0.1805
R^2	0.1617	0.1983	0.1049	0.0326

The results show a similar consistency to those obtained from the fully connected, feedforward network. The predictive quality of the hybrid model degrades gracefully over the three out-of-sample test periods. The "goodness of fit" measures for the hybrid GA/NN are noticeably better than those obtained from the fully connected, feedforward NN.

One notable additional distinction between the models concerns the number of weights (connections) utilised in the hybrid GA/NN model. The best model in the final population utilises a structure containing 26 connections, less than half the number utilised in the fully connected model. **Table 3** shows the network structure, excluding the bias nodes, of the highest fitness member of the final population of GA/NN models. A 1 indicates that a connection is used, a 0 indicates that it is not. Connection pruning implicitly selects the number of hidden nodes and the number of inputs employed in the final network.

Table 3: Best Hybrid GA/NN Model-Structure

	Hidden Nodes					
	1	*2*	*3*	*4*	*5*	*6*
Input 1	0	0	1	0	1	1
Input 2	0	1	0	1	0	1
Input 3	0	1	1	1	0	0
Input 4	1	0	1	1	1	1
Input 5	0	1	1	1	1	1
Input 6	0	0	0	1	0	0
Input 7	0	1	1	1	1	1
Input 8	1	1	1	1	0	1
Input 9	1	0	0	0	0	1
Input 10	1	0	1	1	1	0
Output	0	1	1	0	0	1

In the final structure, hidden nodes 1, 4 and 5 are not connected to the output layer, effectively pruning these nodes from the network. Input 6 is not connected to any of the remaining hidden layer nodes and therefore has been ignored as an input by the hybrid GA/NN.

Despite utilising a sparser network structure, the predictive accuracy of the evolved GA/NN model both in- and out-of-sample, as measured by the metrics adopted in this study, is greater than that of the fully connected model. Although definitive conclusions cannot be drawn from a single experiment, the results obtained do not support a hypothesis that a NN structure evolved using a GA will be outperformed by a fully connected, feedforward NN. This suggests that, despite the prevalence of fully connected models in applications of NNs, these structures could actually impound a notable degree of redundancy, utilising a higher number of connections than required, leading to a "bloated" NN model. The ability of the GA to select which connections it wishes to use provides a generalisation of the typical fully connected NN model development process. The GA may choose to evolve a fully connected model, if in fact such a model provides the best fit of the training data.

The findings of this study also cast light on the question of data sufficiency when constructing NN models. A wide variety of opinions exist on the minimum number of input/output data vectors required to train a NN of a given size adequately. Proffered rules of thumb range from three data vectors per network weight (Bigus, 1996) to ten (Trigueiros and Taffler, 1996). These differences may be reconciled by considering the sparseness of the underlying structure of the NN. A fully connected network may have a number of connections with weights close to zero, effectively possessing a sparse structure whilst appearing fully connected. In such cases, the amount of data required to train the network may be reduced in comparison with that required to train a non-sparse network of similar size.

Table 4 provides the weight matrix and weight contribution totals for the best GA/NN. In constructing this table, the incoming/outgoing weights associated with hidden layer nodes pruned from the network by the GA, as well as the weights of other connections not utilised by the network, are set to zero.

Table 4: Weight Structure and Input Node Contribution

	Hidden Nodes			*Contribution%*
	2	*3*	*6*	
Bias to Hidden	–0.2133	–0.8000	0.8000	14.08
Input 1	0.0000	–0.1859	–0.1140	2.33
Input 2	–0.7990	0.0000	–0.5958	10.82
Input 3	–0.8000	–0.7098	0.0000	11.72
Input 4	0.0000	–0.8000	0.3182	8.68
Input 5	–0.8000	0.8000	–0.7982	18.62
Input 6	0.0000	0.0000	0.0000	0.00
Input 7	0.6318	0.7971	–0.0750	11.67
Input 8	0.7990	–0.4000	0.6827	14.61
Input 9	0.0000	0.0000	0.8000	6.21
Input 10	0.0000	–0.1628	0.0000	1.26
Output Node	0.7997	–0.8000	–0.5332	100.00

In order to gain increased insight into the relative importance assigned by the hybrid GA/NN model to each of the data inputs, contribution values were calculated for each input. The contribution value is calculated as the ratio of the sum of the absolute values of the connection weights between each individual input and the hidden layer and the total sum of the absolute values of the connection weights between all input nodes and the hidden layer (Gately, 1996). Analysis of these values suggests that the model places particular emphasis on the inputs concerning the Bank of England index, LIBOR, the 20-day lagged change in the FTSE 100 index and the ratio of the 5/10-day moving average of the FTSE 100 index. However, undue reliance cannot be placed on these findings as prior work has demonstrated that interpretation of the weight matrix, either directly or by means of contribution matrices, may provide limited insight as a variety of weight structures may produce NNs with similar predictive quality (Brabazon, 2001).

CONCLUSIONS

The objective of this paper is to demonstrate, by means of a case study, how the global search properties of a GA could be combined with a NN modelling methodology in order to ascertain both a connection structure and associated weights for a three layer, feedforward NN. The paper also sought to determine the predictive quality of the resulting model by comparing it with a benchmark NN model. The benchmark model was developed using a backpropagation training algorithm, and had a three layer, fully connected, feedforward structure.

Utilising a series of fundamental and technical market data drawn from the FTSE 100 index as a test bed, both NN models attempted to predict the five-day percentage change in the value of the FTSE 100 index. The results of the study show that the GA/NN model utilised a sparse internal connection structure but, despite this, outperformed the fully-connected NN model in terms of predictive ability. This suggests that the application of fully-connected, feedforward neural network structures is not always appropriate and may result in network structures that contain substantial redundancy. This finding provides additional insight into the question of data sufficiency for training a NN, and suggests that there may be particular scope for utilising hybrid GA/NN models to determine whether sparse network structures could be beneficially employed in applications where there is a limited quantity of data available for model construction.

ACKNOWLEDGEMENTS

The author wishes to thank the editors and the anonymous referees for their constructive comments on earlier versions of this paper.

APPENDIX

This appendix provides a more technical overview of the process of developing a model, in terms of landscape theory. In particular, the section draws on the work of Jones (1995). Initially, a discussion of a search process is provided, leading to a definition of a landscape. The implications of this definition are then considered.

A modelling (or search) process can be characterised as "find an object with the following properties". The search is considered to take place amongst a potentially infinite set of objects. The required properties will be problem-specific and could be defined as broadly as "find the rules that explain the process under examination". The first step in a search process is to select the set of objects. This selection provides a basic definition of the problem. Let $\boldsymbol{O}$ represent the object space. The second step in a search process is to select the representation of the objects in $\boldsymbol{O}$. This representation determines a set, $\boldsymbol{R}$, the representation space. The choice of $\boldsymbol{R}$ may represent the general model specification to be searched, for example, a class of NN models. Thus, $\boldsymbol{R}$ represents a specific mapping of $\boldsymbol{O}$. For $\boldsymbol{o} \in \boldsymbol{O}$ and $\boldsymbol{r} \in \boldsymbol{R}$, we can write $o \, \zeta \, r$ to indicate that $\boldsymbol{o}$ is represented by $\boldsymbol{r}$. In a broad class of search problems, the aim is to find as good an object as possible, within a time constraint. As it is generally impossible in large problems to search enumeratively even $\boldsymbol{o} \in \boldsymbol{O}$, a non-random search process will require a function $\boldsymbol{g}\colon \boldsymbol{O} \rightarrow \boldsymbol{G}$ for some set $\boldsymbol{G}$ and a partial order $>_G$, such that, for $\boldsymbol{o}_1, \boldsymbol{o}_2 \in \boldsymbol{O}$, if $\boldsymbol{g}(\boldsymbol{o}_1) >_G \boldsymbol{g}(\boldsymbol{o}_2)$, then $\boldsymbol{g}(\boldsymbol{o}_1)$ is considered a better answer to the search problem than $\boldsymbol{g}(\boldsymbol{o}_2)$.

Define $M(\boldsymbol{S})$ as a multiset defined on $\boldsymbol{S}$. A multiset is a set that is allowed to contain repeated elements. For example, {1}, {4,4,4,4} and {1,5} are multisets drawn from $M(1,4,5)$. Generalising, $\boldsymbol{S}$ may be defined as containing the set of valid parameters that can be used to define a specific instance of $\boldsymbol{R}$, for example, a vector that completely defines a NN structure. Taking a specific vector of these parameters, $\boldsymbol{v} \in M(\boldsymbol{R})$, a "neighbourhood" $N_\varphi(\mathbf{v})$ can be defined such that it represents the set of elements of $M(\boldsymbol{R})$ that can be reached through a single application of an operator, φ. An operator is a function φ: $M(R)$ X $M(R) \in [0..1]$. The value of $\varphi\ (v,w) = p$ for $v,w \in M(R)$ indicates the probability (p) that a single application of the operator transforms $v \rightarrow w$. Therefore, in a search context, an operator (for example, a mutation operator) usually generates a variant of a parameter vector such as v. Finally, a search algorithm requires the existence of $M(R) \rightarrow F$ for a set F (a fitness function) and a partial order $>_F$ over F such that, if v, $w \in M(R)$ and $f(v) >_F f(w)$, then the multiset v is considered to offer better search potential than the multiset w. An example of this would include the use of differential fitness selection. Thus, the algorithm is choosing between multiple elements of R rather than between single elements of R. The choice of this fitness function $f(v)$ lies in the hands of the modeller. From the above, a landscape may be defined as: $L = (R, \varphi, f, F ,>_F)$. The components are: the representation space, an operator, the function f: $M(R) \rightarrow F$ for some set F (the fitness space), and a partial order $>_F$ over F.

This definition has a number of important consequences. Firstly, in a modelling process, model definition selects a representation space. In determining "quality" parameters for this model, the search takes place in this representation space, not the object space. Thus, the notion of a global or local search is in respect of representation, not object, space. Secondly, the landscape being searched is a function of the operators of that search technique and each operator defines a specific landscape. Hence, in a GA methodology, separate landscapes may be defined by (for example) a crossover operator and a mutation operator. The implication of this is that there is no "single" landscape. The choice of operator also defines the concept of a neighbourhood, that is, the set of elements of $M(R)$ that can be reached from v through a single application of the operator. Search algorithms typically examine the "neighbourhood" of existing solutions for better solutions. Thus, a series of decisions are made both explicitly and implicitly by the modeller, which have the affect of defining the landscape to be traversed by the modelling process.

A hybrid methodology entails the inclusion of multiple operators that also combine to define the fitness landscape. The choice of a combined GA/NN approach has implications for the implicit definitions of object and representation space for the problem. The choice of a NN class of structures results in a selection of $o \in O$. In this paper, the GA selects the connection structure and weight values of the NN. The alternative connection structures can be considered as different representation spaces and the weight selection comprises an optimisation within each representation space. Hence, the hybrid methodology initially conducts a series of searches on varying landscapes and implicitly attempts to identify "good landscapes" as the same fitness criterion is used to assess each. On the identification of one (or a small number) of these landscapes, the search process concentrates on identification of "good" weight combinations.

REFERENCES

Allen, F. and Karjalainen, R. (1999). 'Using Genetic Algorithms to Find Technical Trading Rules', *Journal of Financial Economics*, Vol. 51, pp. 245–271.

Arifovic, J. and Gencay, R. (2001). 'Using Genetic Algorithms to Select Architecture of a Feedforward Artificial Neural Network', *Physica A*, Vol. 289, pp. 574–594.

Bauer, R. (1994). *Genetic Algorithms and Investment Strategies*, New York: John Wiley and Sons.

Bigus, J.P. (1996). *Data Mining with Neural Networks*, New York: McGraw Hill.

Brabazon, T. (2001). 'Market Index Modelling Using Neural Networks', *Proceedings BAA Annual Conference*, University of Nottingham, 26–28 March.

Brock, W., Lakonishok, J. and LeBaron, B. (1992). 'Simple Technical Trading Rules and the Stochastic Properties of Stock Returns', *Journal of Finance*, Vol. 47, No. 5, pp. 1731–1764.

Brown, S., Goetzmann, W. and Kumar, A. (1998). 'The Dow Theory: William Peter Hamilton's Track Record Reconsidered', *Journal of Finance*, Vol. 53, No. 4, pp. 1311–1333.

Chan, L.K.C., Jegadeesh, N. and Lakonishok, J. (1996). 'Momentum Strategies', *Journal of Finance,* Vol. 51, No. 5, pp. 1681–1714.

Cheng, A.C.S. (1998). 'International Correlation Structure of Financial Market Movements – the Evidence from the United Kingdom and the US', *Applied Financial Economics*, Vol. 8, No. 1, pp. 1–13.

Deboeck, G. (1994). *Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets*, New York: John Wiley and Sons.

Dissanaike, G. (1997). 'Do Stock Market Investors Overreact?', *Journal of Business Finance and Accounting*, Vol. 24, No. 1, pp. 27–50.

Fogel, D. (2000). *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, New York: IEEE Press.

Gately, E. (1996). *Neural Networks for Financial Forecasting,* New York: John Wiley and Sons.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Boston: Addison, Wesley, Longman.

Gurney, K. (1997). *An Introduction to Neural Networks,* London: University of London Press.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*, Ann Arbor, USA: University of Michigan Press.

Jones, T. (1995). Evolutionary Algorithms, Fitness Landscapes and Search, *Unpublished PhD Thesis*, Department of Computer Science, University of New Mexico.

Neely, C., Weller, P. and Dittmar, R. (1997). 'Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach', *Journal of Financial and Quantitative Analysis*, Vol. 32, No. 4, pp. 405–428.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*, Cambridge, Massachusetts: MIT Press.

Murphy, John J. (1999). *Technical Analysis of the Financial Markets,* New York: New York Institute of Finance.

Sexton, R. and Dorsey, R. (2000). 'Reliable Classification using Neural Networks: A Genetic Algorithm and Backpropagation Comparison', *Decision Support Systems*, Vol. 30, pp. 11–22.

Sexton, R. and Gupta, J. (2000). 'Comparative Evaluation of Genetic Algorithm and Backpropagation for Training Neural Networks', *Information Sciences*, Vol. 129, pp. 45–59.

Trigueiros, D. and Taffler, R. J. (1996). 'Neural Networks and Empirical Research in Accounting', *Accounting and Business Research*, Vol. 26, No. 4, pp. 347–355.

Varetto, F. (1998). 'Genetic Algorithms in the Analysis of Insolvency Risk', *Journal of Banking and Finance*, Vol. 22, No. 10, pp. 1421–1439.

Wong, B., Lai, V. and Lam, J. (2000). 'A Bibliography of Neural Network Business Applications Research: 1994–1998', *Computers and Operations Research*, Vol. 27, pp. 1045–1076.