Designing Radial Basis Function Networks for Classification Using Differential Evolution

Bryan O'Hora, Jerome Perera and Anthony Brabazon

Abstract— **The construction of a quality RBF network for a specific application can be a time-consuming process as the modeller must select both a suitable set of inputs and a suitable RBF network structure. Evolutionary methodologies offer the potential to automate all or part of these steps. This study illustrates how a hybrid RBFN-DE system can be constructed, and applies the system to a number of datasets. The utility of the resulting RBFNs on these classification problems is assessed and the results from the RFBN-DE hybrids are shown to be competitive against the best performance on these datasets using alternative classification methodologies.**

I. Introduction

The objective of this study is to illustrate the potential for automating the construction of a radial basis function network (RBFN) using differential evolution (DE) and to assess the utility of the resulting classification systems on a variety of test datasets. A RBFN generally consists of a three-layer feedforward network. Just as for the canonical multi-layer perceptron (MLP), a RBFN can be used for prediction and classification purposes, and both the MLP and RBFN are universal approximators. However, RBFNs differ from MLPs in that the activation functions of the hidden layer nodes are radial basis functions.

A. RBFN Construction

The training of RBFNs typically consists of a combination of unsupervised and supervised learning. Initially, a number hidden layer nodes (or centres) must be positioned in the input data space. When each input vector is presented to the network a value is calculated at each centre using a radial basis function, for example a Gaussian function. In the case of a Gaussian function, this value represents a measure of the quality of the match between the input vector and the location of that centre in the input space. Each hidden node therefore, can be considered as a local detector in the input data space.

The second phase of the model construction process for a RBFN is the determination of the value of the weights on the connections between the hidden layer and the output layer. In training these weights, the output value for each input vector are known, as are the activation values for that input vector at each hidden layer node, so a supervised learning method can be used. The simplest transfer function for the node(s) in the output layer is a linear function where the network's output is a linearly weighted sum of the outputs from the hidden nodes. In this case, the weights on the arcs to the output node(s) can be found using linear regression, with the weight values being the regression coefficients. Sometimes it may be preferred to implement a non-linear transfer function at the output node(s). The basic algorithm for the canonical RBFN is as follows:

i. Select the initial number of centres (m).
ii. Select the initial location of each of the centres.
iii. For each input data vector/centre pairing calculate the activation value $\phi(||x-y||)$, where ϕ is a radial basis function and $||...||$ is a distance measure between input vector x and a centre y in the data space. As an example, let $d = ||x-y||$. The value of a Gaussian RBF is then given by $y = \exp(\frac{-d^2}{2\sigma^2})$, where σ is a modeller selected parameter which determines the bandwidth of the centre.
iv. Once all the activation values for each input vector have been obtained, calculate the weights for the connections between the hidden and output layers using linear regression.
v. Go to step (iii) and repeat the above steps until a stopping condition is reached.
vi. Improve the fit of the RBFN to the training data by adjusting some or all of the following: the number of centres, their location, or the bandwidth of the radial basis functions.

As can be seen from the above steps, substantial modeller involvement is required in order to construct a quality RBFN. Another issue which requires modeller involvement is the selection of a quality set of model inputs. Both of these steps represent a combinatorial problem and the solution of this problem can be largely or partly automated through the application of an evolutionary algorithm such as DE. Although the application of evolutionary algorithms for the purposes of constructing neural networks has given rise to a large literature (see [1] for a good review), much of the early literature predates the use of DE and concentrates on the application of GA-based algorithms. The development of DE and its good performance across a range of studies suggest that it could be usefully hybridised with neural network models.

A relatively small number of recent papers have applied DE for the purposes of constructing various forms of MLPs, typically reporting good results [2], [3], [4], [5]. However, apart from [6] and [7] there are very few papers describing an application of DE for the purposes of constructing a RBFN. This paper seeks to address this gap.

Bryan O'Hora, Jerome Perera and Anthony Brabazon are members of the Natural Computing Research and Applications Group (http://ncra.ucd.ie/), University College Dublin, Dublin, Ireland (phone: + 353 1 7164705; fax: + 353 1 7164767; email: anthony.brabazon@ucd.ie).

B. Structure of Paper

The rest of this paper is organised as follows. The next section provides a concise overview of DE. We then outline the experimental methodology adopted. The remaining sections provide the results of these experiments followed by a number of conclusions.

II. Differential Evolution

Differential evolution (DE) [8], [9], [10], [11] is a population-based search algorithm which typically operates on real-valued solution encodings. DE bears some similarity with the genetic algorithm (GA) [12] in that both algorithms maintain a population of potential solution encodings which are then perturbed in an effort to uncover yet better solutions to a problem of interest. In the GA, the key steps are fitness-driven selection, crossover and mutation. In DE, individual encodings are typically represented as real-valued vectors, and the perturbation of solution vectors is based on the scaled difference of two randomly chosen members of the current population. One advantage of this approach is that the resulting 'step' size and orientation during the perturbation process automatically adapts to the objective function landscape.

A. Canonical DE Algorithm

Although several DE algorithms exist, in the interests of brevity, we primarily describe one version of the algorithm based on the *DE/rand/1/bin* scheme [9]. The different variants of the DE algorithm are described using the shorthand DE/x/y/z, where x specifies how the base vector (of real values) is chosen (*rand* if it is randomly selected, or *best* if the best individual in the population is selected), y is the number of difference vectors used, and z denotes the crossover scheme (*bin* for crossover based on independent binomial experiments, and *exp* for exponential crossover).

At the start of the algorithm, a population of N, d-dimensional vectors $X_j = (x_{i1}, x_{i2}, \ldots, x_{id})$, $j = 1, \ldots, N$, each of which encode a solution, is randomly initialised and evaluated using a fitness function f. During the search process, each individual (j) is iteratively refined. The modification process has three steps:

i. Create a *variant vector* which encodes a solution, using randomly selected members of the population (mutation step).
ii. Create a *trial vector*, by combining the variant vector with j (crossover step).
iii. Perform a *selection process* to determine whether the newly-created trial vector replaces j in the population.

Under the mutation operator, for each vector $X_j(t)$ a variant vector $V_j(t+1)$ is obtained:

$$V_j(t+1) = X_m(t) + F(X_k(t) - X_l(t)) \qquad (1)$$

where $k, l, m \in 1, \ldots, N$ are mutually distinct, randomly selected indices, and all the indices $\neq j$ (X_m is referred to as the base vector and $X_k(t) - X_l(t)$ is referred to as a difference vector). Selecting the three indices randomly implies that all members of the current population have the same chance of being selected, and therefore influencing the creation of the difference vector. The difference between vectors X_k and X_l is multiplied by a scaling parameter F (typically $F \in (0, 2]$). The scaling factor controls the amplification of the difference between X_k and X_l and is used to avoid stagnation of the search process. There are several alternative versions of the above process for creating a variant vector (see [9] for details of these).

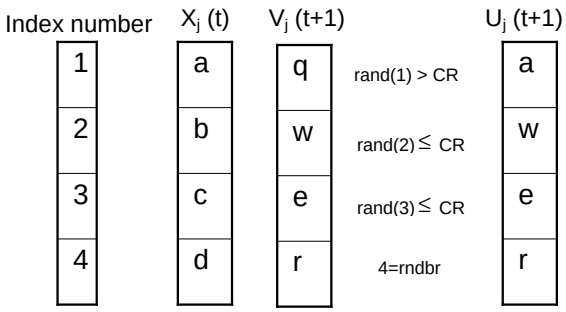

Fig. 1. An example of crossover in DE

A notable attribute of the mutation step in DE is that it is self-scaling. The size/rate of mutation along each dimension stems solely from the location of the individuals in the current population. The mutation step self-adapts as the population converges leading to a finer-grained search. In contrast, the mutation process in the canonical GA is typically based on draws from a fixed probability density function.

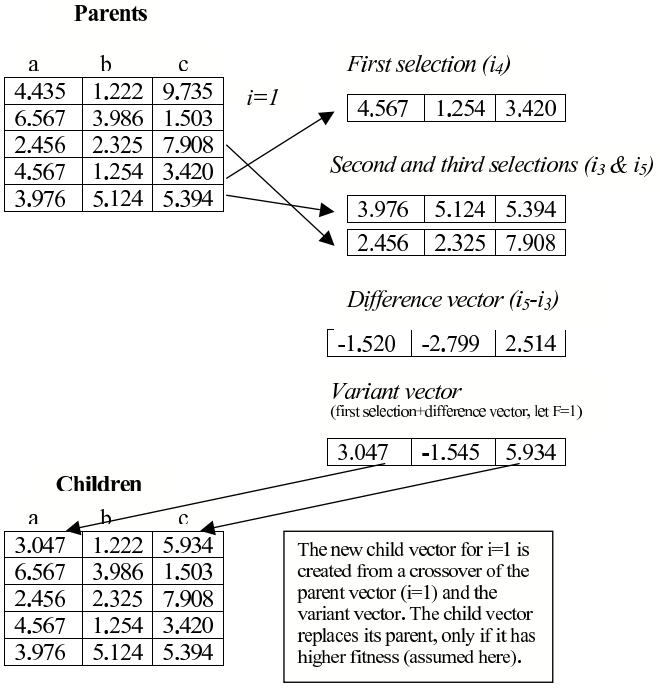

Fig. 2. Numerical example of the canonical DE algorithm

Following the creation of the variant vector, a trial vector $U_j(t+1) = (u_{j1}, u_{j2}, \ldots, u_{jd})$ is obtained:

$$U_{jk}(t+1) = \begin{cases} V_{jk}(t+1), \\ \text{if } (rand \leq CR) \text{ or } (j = rnbr(ind)) \text{ ;} \\ X_{jk}(t), \\ \text{if } (rand > CR) \text{ and } (j \neq rnbr(ind)). \end{cases} \quad (2)$$

where $k = 1, 2, \ldots, d$, $rand$ is a random number generated in the range (0,1), CR is the user-specified crossover constant from the range (0,1), and $rnbr(ind)$ is a randomly chosen index chosen from the range $(1, 2, \ldots, d)$. The random index is used to ensure that the trial solution vector differs by at least one element from $X_j(t)$. The resulting trial (child) solution replaces its parent if it has higher fitness (a form of selection), otherwise the parent survives unchanged into the next iteration of the algorithm.

Figure 1 provides an illustration of the crossover operator in DE, and Figure 2 illustrates a simple numerical example. In the numerical example, the parent vector is i=1. Three other vectors are randomly chosen to create the variant vector, and F=1 is assumed. When crossover is applied between the parent and the variant vector, the first and the third elements of the variant vector are assumed to combine with the second element of the parent vector to create the trial or child vector. Finally, it is assumed that the fitness of the trial vector exceeds that of its parent and it therefore replaces the parent (Eq. 3).

$$X_j(t+1) = \begin{cases} U_j(t+1), & \text{if } f(U_j(t+1)) > f(X_j(t)); \\ X_j(t), & \text{otherwise.} \end{cases} \quad (3)$$

Price and Storn [13] provide a comprehensive comparison of the performance of DE with a range of other optimisers, including the GA, and report that the results obtained by DE are consistently as good as the best obtained by other optimisers across a wide range of problem instances. They also report that DE is robust with respect to choice of parameter settings.

B. DE Variants

A wide variety of DE algorithms exist (see [13] for an up-to-date review). Two recent variants are briefly discussed. Das et al. [14] suggested that rather than holding the value of F constant during the optimisation run, it could be allowed to vary, either randomly in the range $(0.5 \rightarrow 1)$ (*DE with random scale factor*), or it could be decreased linearly during the optimisation run from an upper to a lower bound (*DE with time varying scale factor*). The first idea aims to reduce the chance of the search process stagnating at a local optimum, the second aims to encourage diverse searching early in the optimisation run, with a finer degree of search later in the optimisation run. Both approaches were found to outperform canonical DE across a range of test functions. In contrast, Norma and Iba [15] proposed *Fittest Individual Refinement* (FIR) wherein the canonical form of DE is supplemented by a crossover-based local search step (XLS), in order to assist in finding the optimum solution. In essence, this results in a memetic variant of DE. In this approach a local search is undertaken around the best individual after each iteration of the algorithm by selecting it as breeding stock, mating it with a number of newly created variant vectors, and then determining whether any of the child vectors generated have higher fitness.

III. Experimental Approach

A total of four test datasets were used in our investigations, two drawn from the UCI machine learning repository [16], and two financial datasets which have been used in prior studies [17], [18], [19], [20]. All of the datasets consist of a binary classification problem and have between eight and thirty input variables.

A. Financial Datasets

The financial datasets are drawn from the domains of corporate failure, and bond-rating prediction. Because of the commercial significance of developing high-quality classifiers for these domains, there has been substantial research developing a wide-range of classification models in each.

1) Corporate Failure: Corporate failure is a natural component of the market economy, facilitating the recycling of financial, human and physical resources into more productive organisations [21], [22]. Nonetheless, corporate bankruptcy can impose significant private costs on many parties including shareholders, providers of debt finance, employees, suppliers, customers, managers and auditors. All of these stakeholders have an interest in being able to identify whether a firm is on a trajectory which is tending towards corporate failure. Early identification of such a trajectory could facilitate successful intervention, to avert disaster. Most attempts to predict corporate failure implicitly assume that management decisions critically impact on firm performance [23]. Although management decisions are not directly observable, their consequent effect on the financial health of the firm can be observed through their impact on the firm's financial ratios. Typically when constructing corporate failure prediction models, explanatory variables are drawn from the financial statements of the firm, from financial markets, general macroeconomic variables, and non-financial, firm-specific information). In this study, attention is restricted to information drawn from financial statements.

2) Bond Rating: When a company wants to issue traded debt (bonds), it must obtain a credit rating for the issue from at least one recognised rating agency (Standard & Poor's (S&P), Moody's, Fitches' or Dominion Bond Rating Service). The credit rating represents the rating agency's opinion at a specific date of the creditworthiness of a borrower in general (an issuer credit rating), or in respect of a specific debt issue (a bond credit rating). These ratings impact on the borrowing cost, and the marketability of issued bonds.

Several categories of individuals would be interested in a model that could produce accurate estimates of bond ratings. Such a model would be of interest to firms that are considering issuing debt as it would enable them to estimate the likely return investors would require if the debt was issued, thereby providing information for pricing the bonds. The model could also be used to assess the creditworthiness of firms that have not issued debt and hence do not already have a published bond rating. This information would be useful to bankers or other companies that are considering whether they should extend credit to that firm.

Most rated debt is publicly tradable on stock markets, and bond ratings are typically changed infrequently. An accurate bond-rating prediction model could indicate whether the current rating of a bond is still justified. To the extent that an individual investor could predict a bond re-rating before other investors foresee it, this may provide a trading edge.

3) Rationale for Adopting a RBF-DE Hybrid: There are a number of reasons to suppose that an evolutionary methodology, coupled with a RBF can prove fruitful in the prediction of both corporate failure and bond ratings. Both domains are characterised by a lack of a strong theoretical framework, with many plausible, competing explanatory variables. The selection of quality explanatory variables and model form represents a high-dimensional combinatorial problem, giving rise to potential for evolutionary methodologies. Combining these with the universial approximator qualities of a RBF produces a powerful modelling methodology.

B. Methodology

There are a multitude of ways that a RBFN-DE hybrid could be constructed, depending on what the modeller wishes to achieve. A major decision in creating any form of evolutionary artificial neural network (EANN) is deciding which elements of the system should be evolved and which should not. Although it is tempting to try to evolve as many parameters as possible, this can produce an unfeasibly large search space. In this study we allow DE to select:

- model inputs,
- location of centres,
- bandwidth of each RBF,
- weight on each centre's output, and the
- classification cut-off point.

The number of centres is not evolved in our approach. Rather a suitable number of centres is determined using a grid search process, whereby the number of centres is incremented between a lower and an upper bound, with a separate evolutionary process being repeated at each increment. It would also be possible to evolve the nature of radial basis function at each centre.

I	I*C*n	C	C	1

Fig. 4. Chromosome encoding choice of inputs and RBFN structure.

The chromosomes being evolved each consist of $I + (I * C*n)+2C+1$ real-valued elements (see Figure 4). The first I elements represent a flag which determines whether a specific input (from a set of I inputs) is used, the next $(I * C * n)$ elements represent the location of each of C centres in the n dimensional input space, the next $2C$ elements consist of the band-width for each centre followed by the weight on each centre's output. The last element of the chromosome is the classification cut-off point for the RBFN output. We implement the canonical form of DE (DE/rand/1/bin) but following [14], a time-varying scaling factor is incorporated, whereby F is linearly decreased from 1.5 to 0.5 during the DE run.

$$F = F_{max} - \frac{F_{max} - F_{min}}{max_{iter}} * curr_{iter} \tag{4}$$

In this study, a simulator was created to implement the RBFN-DE hybrid. The user interacts with the simulator by means of a GUI which allows the selection of a wide number of parameters, such as whether the system will evolve the choice of inputs to be used in the classification models, or whether the choice of inputs is in the hands of the modeller. The simulator also allows the user to choose the form of selection, number of difference vectors, the form of crossover to be applied and the form of scaling applied in the DE algorithm (see Figure 3).

IV. Results

The results from our experiments are now provided. In each case the datasets were randomised to produce five recuts, with the data being split approximately 80:20 between training and test in each recut. Each of the experiments is run for 50 generations, with CF=0.8, and a population size of 200. A grid search was carried out when choosing the appropriate number of centres, whereby the number of centres was incrementally increased from 10 to 50 with a step size of 2. The number of centres which produced highest in-sample classification accuracy was then used to calculate the out-of sample results. In selecting the above parameters, initial trial experiments were undertaken. The classification accuracies were not found to be highly sensitive to minor changes in the number of generations, population size or the value of CF.

In the experiments, fitness is defined as the number of correct classifications obtained by an evolved discriminant rule. The results for the best individual of each cut of the dataset averaged over all five randomisations of the dataset, are given in Table I. The table also provides the results for the average mean fitness in the final generation in each recut.

To assess the overall hit-ratio of the developed models both in and out of sample, Press's Q statistic [24] was calculated. In all cases the null hypothesis, that the classification accuracies are not significantly better than those that could occur by chance alone, was rejected at the 1% level. A t-test of the hit-ratios also rejected a null hypothesis that the classification accuracies were no better than chance at the 1% level.

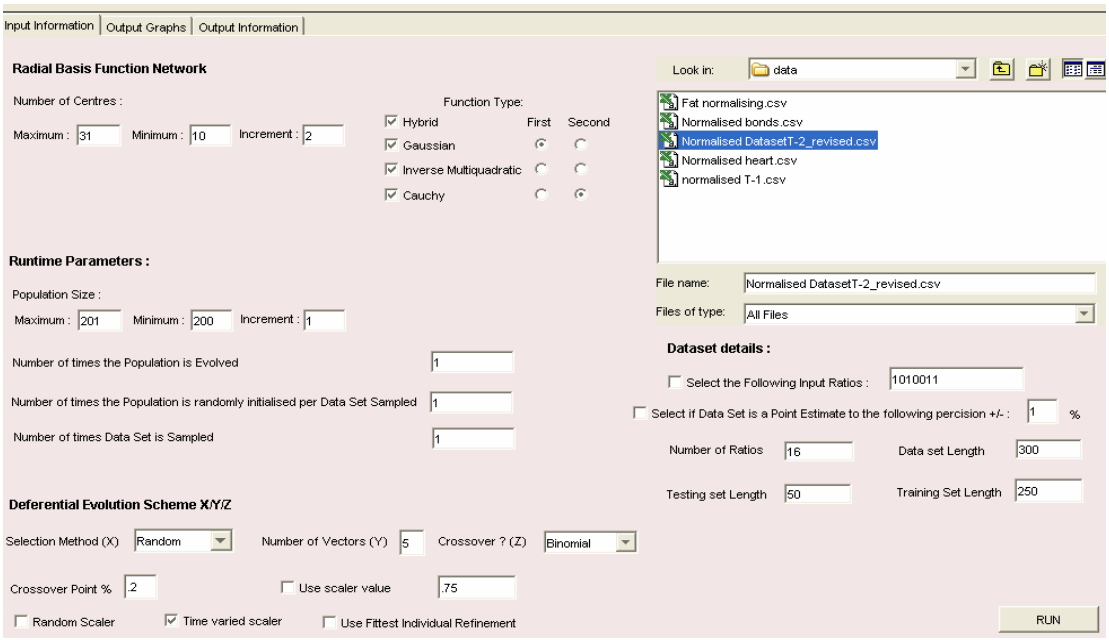

Fig. 3. User interface for simulator.

TABLE I

PERFORMANCE OF THE BEST EVOLVED RULES ON THEIR TRAINING AND OUT-OF-SAMPLE DATASETS, AVERAGED OVER ALL FIVE RANDOMISATIONS.

	In-Sample Average (%)	In-Sample Best (%)	Out-Sample Average (%)	Out-Sample Best (%)
Bonds	82.1	84.4	81.4	83.0
Corporate Failure	77.7	80.7	71.3	86.7
Heart Disease	82.0	85.0	76.4	84.0
Breast Cancer	88.8	91.9	88.7	95.3

In developing the models, the number of centres was varied as noted above. The number of centres which produced the best results on each dataset varied between 26 on the bond data and the Breast cancer data, 22 centres on the corporate failure data, and 20 centres on the heart disease data. Typically, the results produced by the evolved classifiers were not highly sensitive to the number of centres employed, around these optimal values.

A. Comparison of Results

To provide a benchmark for the results obtained by the RBFN/DE hybrid, we compared them with the results obtained on the same data sets by previous authors. In the case of the bond dataset and corporate failure datasets, mean best classification in-sample accuracies of 85% and 85.9% were reported by [20] (out of sample accuracies were 83% and 80% respectively), using a grammar-based evolutionary methodology (grammatical evolution). The classifiers developed by the RBFN-DE hybrid are competitive with these results. In the case of the Cleveland Heart Disease and the Wisconsin Breast Cancer (new) datasets, [25] and [26] show out of sample classification accuracies of around 77% and 96% respectively. Again, the constructed classifiers produce competitive results.

V. CONCLUSIONS & FUTURE WORK

The combination of evolutionary algorithms with various forms of artificial neural network structures is a notable area of current research. It offers the potential to combine the complementary strengths of two distinct methodologies. The objective of this study was to illustrate the potential for automating the construction of a radial basis function network (RBFN) using differential evolution (DE), and to assess the utility of the resulting classification systems on a variety of test datasets. The developed models showed a clear capability to generate effective classification models, and the results from these classifiers proved competitive against prior results from alternative methodologies on the same datasets.

Several extensions of the methodology in this study are indicated for future work. The developed RBFN-DE hybrid generator program will be tested on additional datasets to further assess the generality of the promising results in this study. The developed generator also allows the testing of a variety of hybrid variants including FIR and random scaling and investigation of these will also be the subject

of future work. We also note that the construction of high-quality classifiers is of general utility in a wide variety of application domains including business, medical diagnosis and engineering. Therefore, the methodology outlined in this study has wide potential application.

REFERENCES

[1] Yao, X. (1999). Evolving artifical neural networks, *Proceedings of the IEEE*, 87(9):1423-1447.

[2] Abbas, H. (2002). An evolutionary artificial neural networks approach for breast cancer diagnosis, *Artificial Intelligence in Medicine*, 25(3):265-281.

[3] Abbas, H. (2003). Pareto neuro-evolution: constructing ensemble of neural networks using multi-objective optimization, *in Proceedings of CEC 2003*, 3:2074-2080, 8-12 Dec. 2003.

[4] Magoulas, G., Plagianakos, V. and Vrahatis, M. (2004). Neural network-based colonoscopic diagnosis using on-line learning and differential evolution, *Applied Soft Computing*, 4(4): 369-379.

[5] Kasabov, N. (2001). Evolving Fuzzy Neural Network for Supervised/Unsupervised On-line, Knowledge-based Learning, *IEEE Trans. on Man, Machine and Cybernetics*, 31(6):902-918.

[6] Liu, J. and Lampinen, J. (2005). A Differential Evolution Based Incremental Training Method for RBF Networks, *in Proceedings of GECCO 2005*, pp. 881-888, June 25-29 2005, Washington, ACM Press.

[7] Moalla, S., Alimi, A. and Derbel, N. (2002). Design of beta neural systems using differential evolution, *in Procedings of IEEE International Conference on Systems, Man and Cybernetics 2002*, Vol. 3, 6-9 Oct. 2002.

[8] Price, K. (1999). An introduction to differential evolution, in *New Ideas in Optimization*, eds. Corne, D., Dorigo, M. and Glover, F., pp. 79-108, McGraw-Hill, London.

[9] Storn, R. and Price, K. (1995). Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, *Technical Report TR-95-012: International Computer Science Institute, Berkeley*, 1995.

[10] Storn, R. and Price, K. (1997). Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, 11:341-359.

[11] Storn, R. (1999). System design by constraint adaptation and differential evolution, *IEEE Transactions on Evolutionary Computation*, 3:22-34.

[12] Holland, John H. (1975). *Adaptation in Natural and Artificial Systems*, Michigan: University of Michigan Press.

[13] Price, K., Storn, R. and Lampinen, J. (2005). *Differential Evolution A Practical Approach to Global Optimization*, Springer.

[14] Das, S., Konar, A. and Chakraborty, U. (2005). Two Improved Differential Evolution Schemes for Faster Global Search, *in Proceedings of GECCO 2005*, pp. 991-998, June 25-29 2005, Washington, ACM Press.

[15] Noman, N. and Iba, H. (2005). Enhancing Differential Evolution Performance with Local Search for High Dimensional Function Optimization, *in Proceedings of GECCO 2005*, pp. 967-974, June 25-29 2005, Washington, ACM Press.

[16] UCI Machine Learning Repository, http://www.ics.uci.edu/ mlearn/MLRepository.html.

[17] Brabazon, A. and O'Neill. M. (2004). Diagnosing Corporate Stability using Grammatical Evolution, *International Journal of Applied Mathematics and Computer Science*, 14(3):363-374.

[18] Brabazon, A. and Keenan, P. (2004). A hybrid genetic model for the prediction of corporate failure, *Computational Management Science*, 1(3-4):293-310.

[19] Brabazon, A., Meagher, K., Carty, E. , O'Neill, M. and Keenan, P. (2005). Grammar-mediated time-series prediction, *Journal of Intelligent Systems*, 14(2-3):123-143.

[20] Brabazon, A. and O'Neill, M. (2006). *Biologically Inspired Algorithms for Financial Modelling*, Berlin: Springer-Verlag.

[21] Easterbrook, F. (1990). Is corporate bankruptcy efficient?, *Journal of Financial Economics*, 27(2):411-417.

[22] Schumpeter, J. (1934). *The Theory of Economic Development*, Cambridge, MA: Harvard Business Press.

[23] Argenti, J. (1976). *Corporate Collapse: The Causes and Symptoms*, London: McGraw-Hill.

[24] Hair, J., Anderson, R., Tatham, R. and Black, W. (1998). *Multivariate Data Analysis*, Upper Saddle River, New Jersey: Prentice Hall.

[25] Aha, D. and Kibler, D. (1988). Instance based prediction of heart disease presence with the Cleveland database, Technical Report, University of California at Irvine, Department of Information and Computer Science, Number ICS-TR-88-07, March 1988.

[26] Smith, M. and Bull, L. (2005). Genetic Programming with a Genetic Algorithm for Feature Construction and Selection, *Genetic Programming and Evolvable Machines*, 6(3):265-281.