Applying Genetic Regulatory Networks to Index Trading

Miguel Nicolau, Michael O'Neill, and Anthony Brabazon

Natural Computing Research & Applications Group
University College Dublin
Dublin, Ireland
Miguel.Nicolau@ucd.ie, M.ONeill@ucd.ie, Anthony.Brabazon@ucd.ie

Abstract. This paper explores the computational power of genetic regulatory network models, and the practicalities of applying these to real-world problems. The specific domain of financial trading is tackled; this is a problem where time-dependent decisions are critical, and as such benefits from the differential gene expression that these networks provide. The results obtained are on par with the best found in the literature, and highlight the applicability of these models to this type of problem.

1 Introduction

The Evolutionary Computation (EC) literature tends to adapt mostly evolutionary models in a Darwinian sense: a population of individuals is created, executed, and assigned fitness scores. The most fit individuals are then more likely to survive, through a stochastic process, and the evolutionary cycle continues in this fashion, until a stopping condition is met. This model has proven to be successful throughout the years, but the knowledge of biological systems is ever increasing, and there is a growing trend in exploring more complex and realistic models [2].

One of the key aspects of genetics that is seeing increasing attention is the developmental processes that occur throughout the life of organisms. Rather than adopting a fixed, direct mapping from genotype to phenotype, developmental systems explore the lifelong, conditional expression of genes.

Genetic Regulatory Networks (GRNs) are a key element of gene expression regulation in biological organisms, and one that has seen recent attention in the EC field [1, 10, 13, 11, 5]. GRN-based algorithms explore the idea of differental gene expression through regulatory processes, and as such are potentially useful for dynamic and noisy environments.

This paper further explores the potential of GRNs for Evolutionary Computation, and exemplifies how to apply a recently introduced model [1] to a financial prediction benchmark. GRN models seem well suited to this kind of problem, where at different times of its life, an individual needs to adapt to a constantly changing environment. The results obtained further highlight the potential of GRNs as a computational device, and hopefully help to pave the future for their adoption within the EC community.

The next section introduces the biological principles behind GRNs, and details the implementation of the model used. Section 3 then introduces the problem domain, and Section 4 details the experimental setup and results achieved. Section 5 concludes this study, and highlights future work directions.

2 Gene Regulatory Networks

2.1 Background

In the cell environment, DNA segments containing genes are transcribed (i.e. expressed) into mRNA (messenger RNA) strands, which, through a translation process, are used to combine amino-acids, thus forming proteins. Some of these proteins are known as *Transcription Factors*: their role is to help regulate the expression of other genes, by binding at specific regulation sites. This process results in complex networks, with genes producing proteins regulating the expression of other genes; these are known as *Gene Regulatory Networks* (GRNs).

In the work presented here, the model originally introduced by Banzhaf [1] is used. This model consists of a binary linear genome, which is scanned for promoter regions, identifying the location of genes. It assumes that each gene is always composed of two regulatory sites (inhibiting and enhancing), and that all proteins produced are transcription factors.

This model has been used frequently in the literature. It was shown to exhibit similar dynamics to its natural counterparts, such as the appearance of specific regulatory network motifs [3] and the resulting network topologies [8], and has been evolved to optimise those topologies [12]; the resulting networks have also been extracted and used as a computational device, for a subset of Genetic Programming benchmark problems [11]. The resulting complex regulatory dynamics have also been studied, from the evolution of oscillatory dynamics [10] to actual control problems such as the pole balancing benchmark [13], and also the flag-colouring developmental problem [5].

2.2 The Model

The model used represents the genome as a binary string. This string is scanned for 32 bit long binary sequences, representing *promoter* regions; if found, these identify the location of a gene. The following 32×5 bits then represent the gene contents, and the previous 32×2 bits represent enhancing and inhibitory regions, respectively. Fig. 1 illustrates this.

In this model, a promoter site is the sequence `XYZ01010101`, where `X`, `Y` and `Z` are any 8 bit sequences. The protein produced by the gene is a 32 bit binary sequence, extracted by a majority rule between all 5 sequences of 32 bits that compose it (that is, if 3 or more equally located bits are set to 1, then the corresponding bit in the protein is set to 1).

Regulation works by matching the binary signature of transcription factors and regulating sites with the XOR operation: the result is the regulating

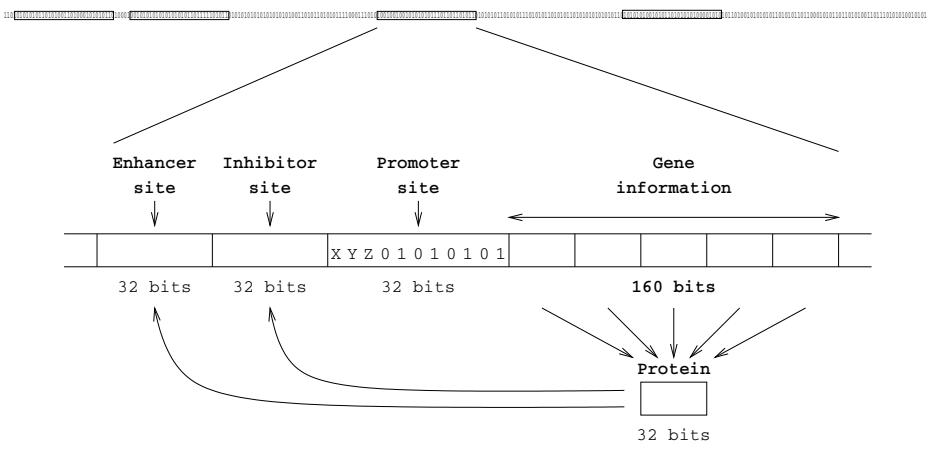

Fig. 1. Bit string encoding of a gene. If a promoter site is found, the gene information is used to create a protein, whose quantity is regulated by the attachment of proteins to the enhancer and inhibitor sites.

strength. The enhancing and inhibiting signals regulating the production of protein p_i are then calculated as:

$$e_i, h_i = \frac{1}{N} \sum_{j=1}^{N} c_j \exp(\beta(u_j - u_{max})) \quad , \tag{1}$$

where N is the total number of proteins, c_j is the concentration of protein j, u_j is the number of complementary bits between the (enhancing or inhibitory) regulating site and protein j, u_{max} is the maximum match observed in the current genome, and β is a positive scaling factor.

The production of p_i is calculated via the following differential equation:

$$\frac{dc_i}{dt} = \delta(e_i - h_i)c_i \quad , \tag{2}$$

where δ is a positive scaling factor (representing a time unit). All the concentrations are normalised at each time step, ensuring that $\sum_i c_i = 1.0$ at all times; this results in competition for resources within the cell environment.

Input and output The original model is a closed world, in that there is no direct interaction with the environment. However, in most problem domains (particularly in reinforcement learning), a training set of input values are associated with a set of responses (or outputs), and the fitness of a solution is typically the difference between the responses obtained and a set of known correct outputs. To this end, a set of I/O extensions were introduced to the original model [13], which are also used in the current work.

To introduce the notion of an input signal, extra regulatory proteins (EPs) are injected into the system. These are not produced by any gene, but also contribute to the regulation of all gene expressions. They represent all the variables required to describe the state of the environment, and their concentrations reflect the (normalised) value of those inputs (and as such are unscaled).

To extract output signals from the model, genes are divided into two classes: TF-genes (i.e. genes encoding transcription factors), and P-genes (genes encoding *product* proteins). These classes of genes are established by scanning the genome for two different promoter sites: in this work, `XYZ00000000` represent TF-genes, and `XYZ11111111` represent P-genes. While the expression levels of TF-genes contribute to the regulatory process as before, the output of P-genes does not; the concentration of the proteins they produce is used as an output signal.

The regulation of TF-genes remains as previously stated, using Eq. 1, but they are normalised taking into account the concentration of EPs as well.

The regulation of P-genes is also determined by Eq. 1, but their expression is calculated with the following equation:

$$\frac{dc_i}{dt} = \delta(e_i - h_i) \quad . \tag{3}$$

Like TF-genes, all concentrations are normalised at each time step, ensuring that $\sum_i c_i = 1.0$ at all times; however, the concentration of TF-proteins and P-proteins are normalised independently.

3 Index Trading

In the financial domain, a market index is a weighted average measure of the price of individual shares that compose that market. Rather than trading single shares (or a portfolio of shares), a popular alternative is to trade on the share market index via an exchange-traded fund, which mirrors as close as possible the collective behaviour of the shares comprising the market. This type of trading has the advantage of not being tied to fluctuations of single shares, but rather to a broader market move. This also means that specific and unexpected share fluctuations are slowly absorbed by the market index [6], allowing for some degree of predictability [9].

Evolutionary algorithms have been successfully applied to financial modelling; the reasons for their applicability include their ability to efficiently explore the search space, and uncover dependencies between input variables, leading to their proper inclusion in the final models [7]. Brabazon and O'Neill [4] provide an overview of the application of evolutionary computation to financial modelling.

The work presented here follows closely the methodology of previous applications of Grammatical Evolution [15] to index trading [14, 4], and uses three datasets, from the UK FTSE 100 index, the Japan Nikkei index, and the German Dax index. All data is drawn from the period between 1/1/1991 and 3/12/1997.

3.1 Technical Indicators

Rather than just observing the raw and historical market price data, it is useful to pre-process this information into technical indicators. These potentially uncover possible useful trends and other information from the raw data series, while simultaneously reducing the inherent noise in the series. Although a potentially infinite number of such indicators may exist, certain classes of indicators are regularly used by investors [9, 16]. The following indicators are used in this study:

- **Moving Average**. This indicator returns the average price of the last n days; as it smooths out daily price fluctuations, it can unveil the underlying trend of the market. The parameter n controls the degree of smoothness.
- **Stochastic Oscillator**. This indicator returns the relative location of the current price in relation to its full price range over a period of of n days; it is useful in trying to predict price turning points.
- **Momentum Change**. This indicator compares the closing price with that of n days ago, and returns the rate of change. It indicates trend by remaining positive while an uptrend is sustained, or negative in the opposite case.

3.2 Datasets

Fig. 2 plots each dataset. These were divided into one training and three testing periods, of 365 days each, for the purpose of model validation. In accordance with previous studies [4], the data was pre-processed prior to evolution. Initially the raw prices were transformed into a moving average with a 75 day gap; these values were then normalised into the range of 0 to 1. This means that data from the first 75 days was not used for the purposes of trading simulation, neither was the data remaining after the four training and testing datasets.

3.3 Methodology

An evolved trader produces one of three signals for each day of the training or test periods: *buy*, *sell*, or *do nothing*. Starting with an initial capital of \$10000, the following trading methodology is used [9, 14]. If a buy signal is issued, a fixed \$1000 investment is made in the market index; this position is automatically closed at the end of a ten day period. If a sell signal is issued, an investment of \$1000 is sold short, and it is also closed after ten days. This means that a maximum of \$10000 are invested at any given point in time. The profit or loss at the end of each trading period takes into account a one-way trading cost of 0.2%, and a further 0.3% to account for slippage. Uncommitted funds take into account a risk-free rate of return, which is approximated using the average interest rate over the entire dataset.

4 Experiments

4.1 Encoding Input and Output variables

Four technical indicators were used in this study: a moving average of 10 days ($mAvg(10)$), momentum change of 5 ($mChange(5)$) and 10 days ($mChange(10)$),

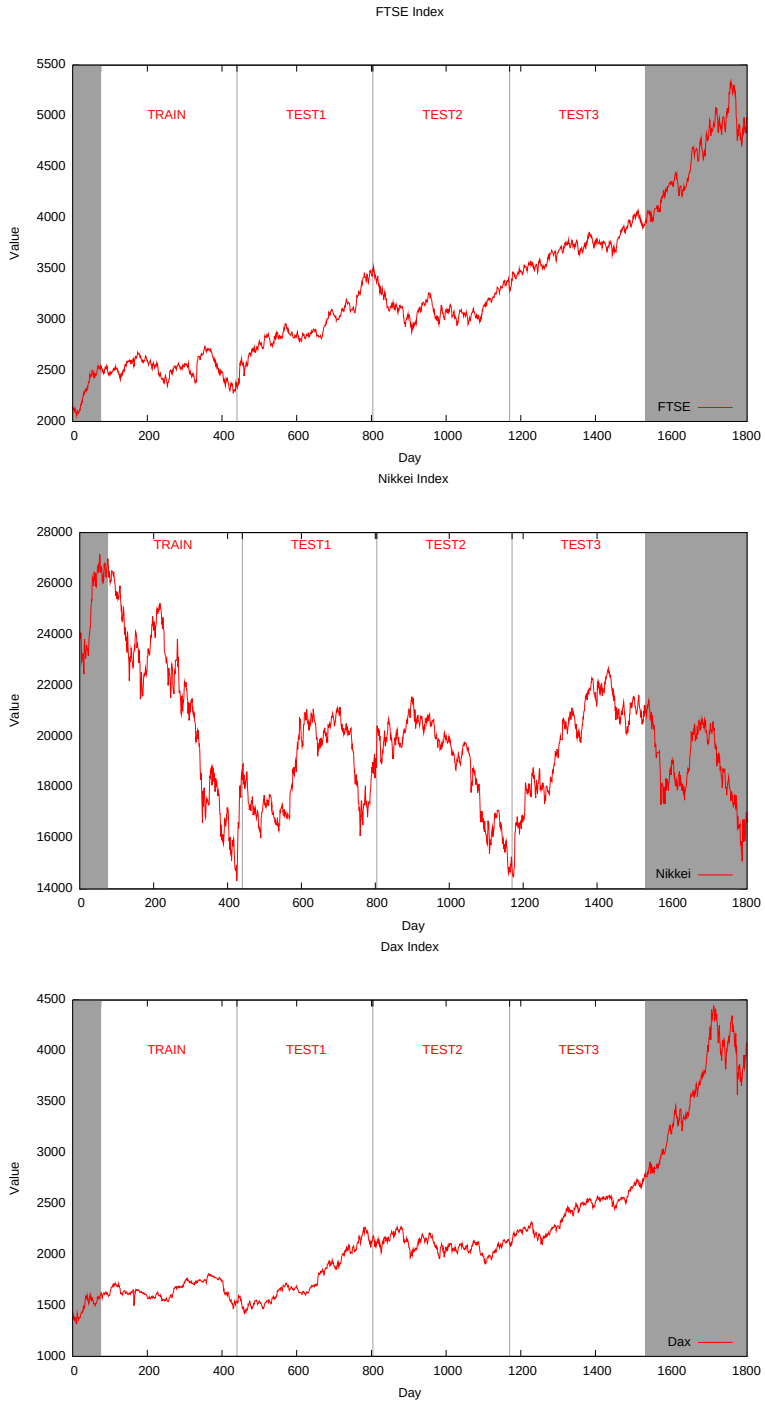

Fig. 2. Plots of the three markets analysed, and the train and test periods used. Gray shaded areas show the initial 75 day moving average gap, and the remaining data after the four year-long training and testing sets, and were not used for the simulations.

and a stochastic oscillator of 10 days ($sOsc(10)$). These were encoded using EPs, as explained in Section 2.2; the signatures for the EPs were chosen to be as different as possible, and were encoded as follows:

$mAvg(10)$: `00000000000000000000000000000000`
$sOsc(10)$: `00000000000000001111111111111111`
$mChange(5)$: `11111111111111110000000000000000`
$mChange(10)$: `11111111111111111111111111111111`

The GRN was allowed to first run for a maximum of 100000 iterations, or until all protein concentrations were stabilised; after this period, the trading session begins. To synchronise the GRN with the trading simulator, a trading signal was extracted every 2000 iterations.

To extract a trading signal from the network, the rate of change of a given P-gene is analysed: if its concentration has increased by more than 0.1%, then a *buy* signal is issued; if it has decreased by more than 0.1%, a *sell* signal is issued; otherwise, a *do nothing* signal is issued[1]. All P-genes present in the genome are tested, and the most successful one is used.

This methodology thus encodes technical indicators as regulatory proteins, which influence the internal regulatory process of the genome, and therefore influence the resulting concentration of P-genes, which can then be interpreted as a trading signal. It is a very similar process as seen in previous applications of GRNs to time-series datasets [13].

4.2 Evolutionary Setup

A $(250+250)-ES$ evolutionary strategy was used to evolve the binary genomes: a population of 250 individuals is used to create 250 offspring, and the best 250 of all parents and offspring are used as the new parent population (a maximum of 100 iterations were allowed). The variation operator used was a bit-flip mutation, set to 1% and adapted by the 1/5 rule of Evolution Strategies [17].

4.3 Measuring Performance

A two-set methodology was used, with the system being trained in an initial training set of one year. Once the training period was over, the system went "live", and was ran on the three test (out of sample) periods, for the purpose of a live trading simulation.

A common passive trading strategy is *Buy and Hold*, where an investor buys stocks and holds them for a long time. It is based on the idea that financial markets give a good return for investment in the long run, regardless of fluctuations and periods of volatility. In order to evaluate the performance of the evolved

[1] this is an entirely experimental value, and has not been optimised; it was left to the structure of the GRN to adapt to it.

traders, their performance was compared to a buy and hold strategy for each of the training and test datasets.

When evolving an index trader, certain aspects required special attention. It would be inadequate to simply calculate fitness as the profit return, as this fails to consider the risk of deploying an evolved trader [14]. A measurement of this risk is provided by the maximum drawdown, that is, the maximum cumulative loss of the system during each of the datasets. As seen in previous studies [14, 4], this can be incorporated into the fitness calculation by subtracting the maximum cumulative loss from the profit of each period. This is in addition to trading costs and slippage penalties, as detailed in Section 3.3.

4.4 Results and Analysis

Table 1 presents the results obtained with the best evolved controller for the FTSE, Nikkei and Dax markets, for the train and validation periods. The best evolved trader for the FTSE and Nikkei markets outperforms the benchmark buy and hold strategy, whereas on the Dax market it slightly underperforms; this is likely linked to the fact that the Dax market is very well behaved, across the period analysed, with very rare fluctuations. These results are on par with similar EC approaches found in the literature [14, 4].

Another interesting aspect of the evolved traders is their low investment risk. The buy and hold strategy keeps the full available capital of $10000 invested at all times, whereas the evolved traders kept average capital investments of only $3582.19, $2806.85 and $3225.34, for the FTSE, Nikkei and Dax markets, respectively. This is a combination of the inclusion of risk penalties in the fitness function, and the fact that the system can only trade $1000 daily.

Fig. 3 plots the best evolved trader for the FTSE market. It exhibits a very cautious approach to trading, with large periods of inactivity, resulting from a slow reaction to the regulatory proteins representing technical indicators, within the GRN. This is mostly induced by the inclusion of the risk-free rate of return and the maximum drawdown penalty into the fitness function. This is seen in Fig. 3, in the third testing period, where the trader exhibits a very cautious approach, even though the market is generally trending upwards. Further optimisation of structure and parameters of the GRN should improve this issue.

5 Conclusions

This paper explored the computational power of regulatory networks, and how to apply them to a real-world financial trading domain. The methodology required to apply GRNs was explored, and the results obtained show the potential of this approach, with results on par with the literature.

There remains much work to be done. Regarding the current problem domain, the evolved trader seems occasionally unresponsive to market changes; the use of different technical indicators could improve this issue. Also, regarding the applicability of the model, further research is required, in particular with what concerns its parameterisation.

Table 1. Best evolved traders for all datasets compared to Buy & Hold benchmark

FTSE market

Period (days)	Buy & Hold	Best-of-run	Avg. daily inv.
Train (75 to 439)	-1269.28	3275.96	5939.73
Test 1 (440 to 804)	4886.9	1083.58	2191.78
Test 2 (805 to 1169)	-1089.8	541.806	3709.59
Test 3 (1170 to 1534)	1908.53	500.949	2487.67
Total	**4436.35**	**5402.295**	

Nikkei market

Period (days)	Buy & Hold	Best-of-run	Avg. daily inv.
Train (75 to 439)	-6345.5	6163.38	5128.77
Test 1 (440 to 804)	1014.79	1125.6	1457.53
Test 2 (805 to 1169)	-5263.49	2144.71	3679.45
Test 3 (1170 to 1534)	4040.59	1331.56	961.644
Total	**-6553.61**	**8514.05**	

Dax market

Period (days)	Buy & Hold	Best-of-run	Avg. daily inv.
Train (75 to 439)	-882.241	2899.86	4586.3
Test 1 (440 to 804)	4047.63	952.689	3347.95
Test 2 (805 to 1169)	-551.995	608.161	1471.23
Test 3 (1170 to 1534)	2972.24	992.868	3495.89
Total	**5585.634**	**5453.578**	

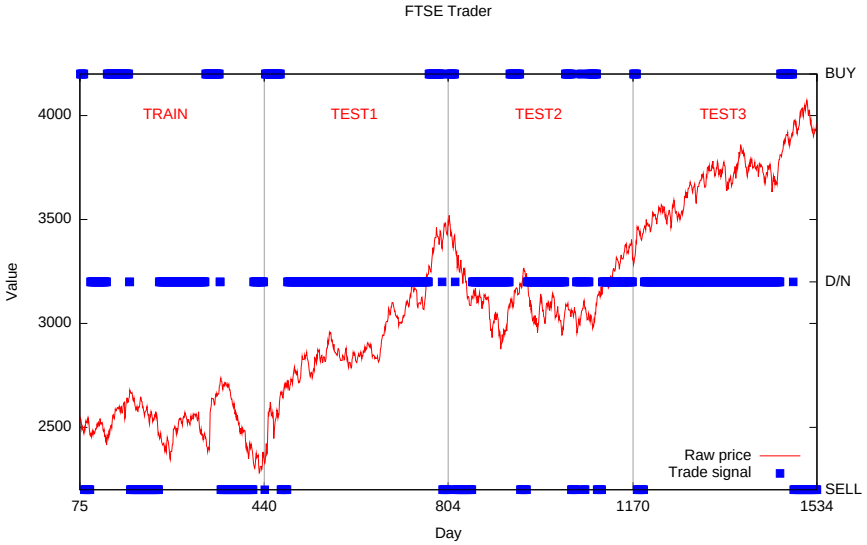

Fig. 3. Best evolved trader for the FTSE index. Blue dots indicating buy, do nothing and sell signals are plotted along with the raw market prices.

References

1. Banzhaf, W.: Artificial regulatory networks and genetic programming. In: Riolo, R., Worzel, B. (eds.) Genetic Programming Theory and Practice, chap. 4, pp. 43–62. Kluwer Publishers, Boston, MA, USA (November 2003)
2. Banzhaf, W., Beslon, G., Christensen, S., Foster, J.A., cois Képès, F., Lefort, V., Miller, J.F., Radman, M., Ramsden, J.J.: From artificial evolution to computational evolution: a research agenda. Nature Reviews Genetics 7, 729–735 (2006)
3. Banzhaf, W., Kuo, P.D.: Network motifs in natural and artificial transcriptional regulatory networks. Biological Physics and Chemistry 4(2), 85–92 (2004)
4. Brabazon, A., O'Neill, M.: Biologically Inspired Algorithms for Financial Modelling. Springer (2006)
5. Cussat-Blanc, S., Bredeche, N., Luga, H., Duthen1, Y., Schoenauer, M.: Artificial gene regulatory networks and spatial computation: A case study. In: Lenaerts, T., Giacobini, M., Bersini, H., Bourgine, P., Dorigo, M., Doursat, R. (eds.) Advances in Artificial Life, 11th European Conference, ECAL 2011, Paris, August 8-12, 2011, Proceedings. pp. –. MIT Press (2011)
6. Hong, H., Lim, T., Stein, J.C.: Bad news travels slowly: Size, analyst coverage, and the profitability of momentum strategies. Journal of Finance 55(1), 265–295 (2000)
7. Iba, H., Nikolaev, N.: Genetic programming polynomial models of financial data series. In: 2000 Congress on Evolutionary Computation, CEC 2000, San Diego, CA, USA, July 16-19, 2000, Proceedings. vol. 2, pp. 1459–1466 (2000)
8. Kuo, P.D., Banzhaf, W.: Small world and scale-free network topologies in an artificial regulatory network model. In: et al., J.P. (ed.) Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems. pp. 404–409. Bradford Books, USA (2004)
9. LeBaron, B., Lakonishok, J., Brock, W.: Simple technical trading rules and the stochastic properties of stock returns. Journal of Finance 47(5), 1731–1764 (1992)
10. Leier, A., Kuo, P.D., Banzhaf, W., Burrage, K.: Evolving noisy oscillatory dynamics in genetic regulatory networks. In: et al., P.C. (ed.) Proceedings of EuroGP 2006. LNCS, vol. 3905, pp. 290–299. Springer (2006)
11. Lopes, R., Costa, E.: ReNCoDe: A Regulatory Network Computational Device. In: et al., S.S. (ed.) Proceedings of EuroGP 2011. LNCS, vol. 6621, pp. 142–153. Springer (2011)
12. Nicolau, M., Schoenauer, M.: On the evolution of scale-free topologies with a gene regulatory network model. BioSystems 98(3), 137–148 (December 2009)
13. Nicolau, M., Schoenauer, M., Banzhaf, W.: Evolving genes to balance a pole. In: et al., A.I.E.A. (ed.) Proceedings of EuroGP 2010. LNCS, vol. 6021, pp. 196–207. Springer (2010)
14. O'Neill, M., Brabazon, A., Ryan, C., Collins, J.: Evolving market index trading rules using grammatical evolution. In: Boers, E.J.W., Cagnoni, S., Gottlieb, J., Hart, E., Lanzi, P.L., Raidl, G.R., Smith, R.E., Tijink, H. (eds.) EvoWorkshops 2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM, Como, Italy, April 18-20, 2001, Proceedings. LNCS, vol. 2037, pp. 343–352. Springer-Verlag (2001)
15. O'Neill, M., Ryan, C.: Grammatical Evolution - Evolutionary Automatic Programming in an Arbitrary Language, Genetic Programming, vol. 4. Kluwer Academic (2003)
16. Pring, M.J.: Technical Analysis Explained: The Successful Investor's Guide to Spotting Investment Trends and Turning Points. McGraw-Hill (1991)
17. Rechenberg, I.: Evolutionsstrategie '94. Frommann-Holzboog, Stuttgart (1994)