Live Trading with Grammatical Evolution

Ian Dempsey[1], Michael O'Neill[1], and Anthony Brabazon[2]

[1] Biocomputing and Developmental Systems Group
University of Limerick, Ireland.
Ian.Dempsey@fmr.com, Michael.ONeill@ul.ie
[2] Department of Accountancy
University College Dublin, Ireland.
Anthony.Brabazon@ucd.ie

Abstract. This study reports work in progress on the development of an on-line evolutionary automatic programming methodology for uncovering technical trading rules for the S&P 500 index. The system adopts a variable sized investment strategy based on the strength of the signals produced by the trading rules. Rogue rules, which generate excessive signals, led to poor market activity. Here we examine the viability of a signal decay constant to reduce the effect of rogue rules. The results show that an aggressive decay rate yielded more profitable results for the trading period January 1st 1991 to December 1st 1997.

1 Introduction

Following on from previous studies, which use Grammatical Evolution (GE) to evolve financial trading systems [1–6] the objective of this study is to perform an initial examination of live evolution and trading using GE. Previous work in this area approached the problem by first evolving trading rules over a training set and then applying the best rules to the out-of-sample or 'test' data. While these studies demonstrated the utility of GE for financial prediction, their results deteriorated over time as the trading rules remained static over the out-of-sample period. In this study the evolved trading system is adaptive, in that it continues to retrain as new data becomes available using a variant on the *moving window* approach, while still maintaining a *memory* of past good rules. This permits the system to adapt to dynamic market conditions, while not completely losing the memory of good solutions that worked well in past market environments. It also has the benefit of being able to take variable sized positions in the market depending on the strength of the signal produced by the technical rules. The population of trading rules is initially created using a set of training data in the usual manner, and after a certain number of generations the system goes 'live' and begins actively trading. The trading rules undergo more generations of evolution as new, additional, data becomes available during the out-of-sample period.

An examination of the problem domain and experimental setup, including the trading methodology used are presented in Section 2, and results is presented in Section 3. Finally the paper concludes in Section 4.

2 Problem Domain & Experimental Approach

The aim of these experiments was to examine the affect on the returns generated by the trading system caused by varying sizes of a decay constant, the significance of which will be described in the next section. The system was applied to the period January 1st 1991 to December 1st 1997 of the S&P 500 with the first 440 days comprising of the initial training period including a 75 day buffer at the beginning to allow for technical indicator calculations, and 364 days being the size of the training window for subsequent increments through the time series. The number of days the training window was incremented upon each trading period was 5 days. The population size used was 500 with 100 generations of training for the initial period and 10 for each increment in the training window. A generational rank replacement strategy was used with 25% replacement on each generation with 30 runs conducted for each setting as in [6].

2.1 The Trading Methodology

```
<expressions> ::= <expressions> <op> <exp> | <exp>
<exp> ::= MA( <numExp> ) | <numExp>
<op> ::= - | + | * | /
<numExp> ::= <numbers> <op> <numbers> | <numbers>
<numbers> ::= <numbers><number> | <number>
<number> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

In the trading model developed, the rules evolved by GE using the above grammar [6], generate a signal that can result in one of three actions to be taken by the model: Buy, Sell or Do Nothing. The `<expressions>` non-terminal produces the signal through expressions using the `MA` terminal which is a moving average function.

2.2 Variable Position Trading

The *live GE* methodology adopted firstly behaves as in earlier studies. An initial training period is set aside on which the population of proto-trading rules is trained, with the aim that a competent population is evolved after a certain number of generations, G. The system then goes 'live', and a number of actions are taken by the system. The trading system takes the best performing rule from the initial training period, and uses this rule to trade for each of the following x days. After x days have elapsed, the *training window* moves forward in the time series by x days, and the population is retrained over the new data window for a number of generations g, where $g < G$. This embeds both a *memory* and an *adaptive potential* in the trading system, as knowledge of good past trading rules is not completely lost, rather it serves as a starting point for their subsequent adaptation.

A small value of g means that memory is emphasised over adaptation, as the new data has relatively less chance to influence the trading rules. This could be

considered a tuning parameter that could be used to alter the adaptive characteristics of the system, and could itself be open to adaptation. For example, in periods of rapid market change a trading system with a 'long memory' could be disadvantageous, whereas in stable periods, a longer memory could well be advantageous. Similar comments can be made for the trading/re-training window parameter x. If its value is large, the trading rules are altered less frequently, but each adaptive 'step' will tend to be larger.

In prior studies which applied GE to evolve financial trading systems, the *entry strategy* for each trade was to invest a constant $ amount on the production of a Buy or Sell signal. The relative strength of the buy or sell signal was not considered. In contrast in this study, the trading system adopts a more complex entry strategy, and a *variable size* investment is made, depending on the strength of the trading signal. The stronger the signal the greater the amount invested, subject to a maximum investment amount of $1,000 (arbitrary). Therefore the amount invested for each signal is:

$$Amount\ invested\ = \frac{Size\ of\ trading\ signal}{Maximum\ trading\ signal} \tag{1}$$

Signals received from individuals will oscillate around a pivot point of zero. Signals greater than zero constitute a buy signal. To allow the system to decide how much to invest on a given trade, the maximum size of a trading signal must be determined. Initially we set the size of the maximum signal as being the size of the first buy signal generated by the system. If a signal is subsequently generated that is stronger than this, the maximum trading signal is reset to the new amount, weaker signals invest a smaller amount in the same ratio as their signal to the maximum. However, this approach can suffer when a *rogue* trading rule emerges, and needs further refinement. A rogue rule is one which returns a large number, eg a buy signal as in Eq. 2:

$$S = 82645 * 1204 * MA\ (2) \tag{2}$$

If a rule such as this is evolved at the right time (when there is an upward trend in the market) it could make it to the top rank of the population, and result in the production of a high maximum signal. Following from this, after a number of increments in the trading window more complex rules like Eq. 3, could emerge, which produce signals that more accurately reflect the index trends

$$S = MA\ (10) - MA\ (50) \tag{3}$$

Unfortunately, because the signal the new rule produces is so weak in relation to the one produced in Eq. 2, the amount invested would be a relatively insignificant

monetary amount. To cater for the production of rogue rules and the affect they can have on results of the trading system, we create two types of trading simulators, a *training simulator*, which is created each time an individual is trained, and a *persistant simulator*, which is created upon start-up and used during live trading to run the best performing rule at each increment and keep track of returns and positions taken as the time series progresses.

The difference between the two simulators, is that the persistent live simulator includes a *decay constant*, which is applied to the maximum signal at each increment of the training window, in order to reduce the affect of 'rogue' rules like Eq.2, and still maintain an element of continuity from one training increment to the next. The maximum signal is reduced by being multiplied by the decay constant. In this study the decay constants used were: 1.0, 0.75, 0.25 and 0.0, where 1.0 will maintain the last strongest signal through out the life of the time series and 0.0 will cause the maximum amount to be invested after each training window increment, i.e. the maximum signal does not roll over when the trading window is incremented. Aside from this both simulators are the same in all respects.

If the sum to be invested is greater than the cash available, the model will invest the cash available less the costs of the transaction. Upon receipt of a sell signal (one less than zero) all positions are closed.

2.3 Return Calculation

The total return in this model is a combination of its generated return from market activity and the interest gained from the risk free cash position. In this case the interest rates used were the actual US Deposit Interest rates for the same period as the S&P 500 data set, affording an accurate reflection in the cash position. Transaction costs in this model are based on the cost structure used by online trading houses, where flat fees are incurred for the opening and closing of positions, $10 fees were charged upon entry and exit. When the transaction costs represent a certain percentage of the sum to be invested, the investment becomes unfeasible. Therefore the model is made aware of the transaction costs, when the entry and exit costs arise to 20% of the sum to be invested the model will not take the position, effectively ruling out very small investments arising from weak trading signals. This represents the Do Nothing stance, the model will hold all current positions. Thus the pure return over the training period was chosen as the fitness measure.

3 Results

This section is broken up into two parts. The first, 3.1, reports the returns of each setup over its trading range for the final generation of each trading window increment versus the returns made by the index over the same period in training. The second section 3.2 reports the returns made during live trading versus the index.

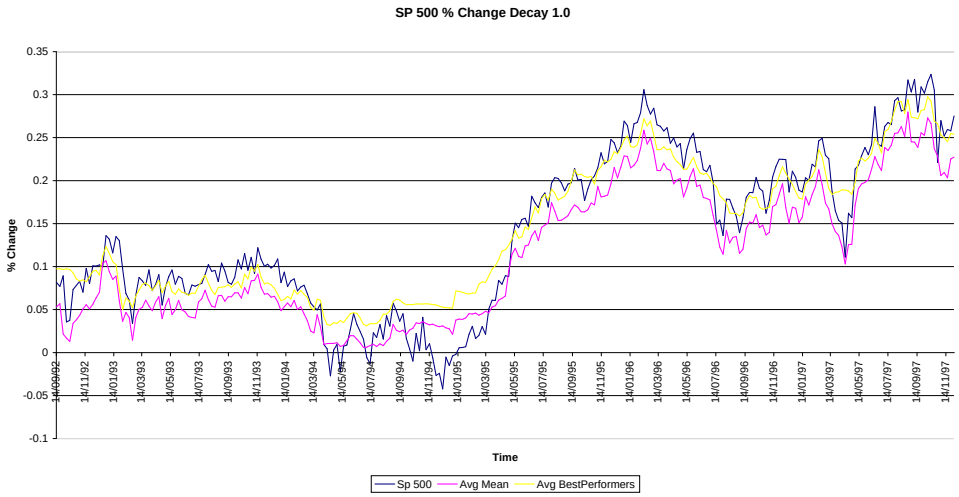

Fig. 1. Performance on the S&P 500 with a decay constant of 1.0.

3.1 Live Training Returns

Upon each increment in the training window, the system's population is retrained to incorporate the new data for 10 generations. On the final generation of each live training period the return of the best performer, i.e. its fitness over the training window, was taken for each increment in the window, this led to the sampling of 273 points. These points represent the average return of the best rule on the final generation for each window increment. The values of the S&P 500 at these points were also taken along with its value at the start of the window range to calculate its return over the same period, proxying the return to a buy-and-hold investment strategy. The result is a series of graphs, which show how closely the system follows the S&P 500 for returns over the same periods (see Figures 1, 2, and 3).

3.2 Live Trading Returns

Six sets of experiments were performed to measure the influence the decay constant had for varying values from 0.0 to 1.0. For each value 30 runs were conducted. Table 1 presents the percentage profits of the best performers from each experiment as well as the average return seen across each run in each setup.
Table 1 identify a decay rate of 0.0 as producing the best performer across all settings with the performance of the best performers improving the stronger the

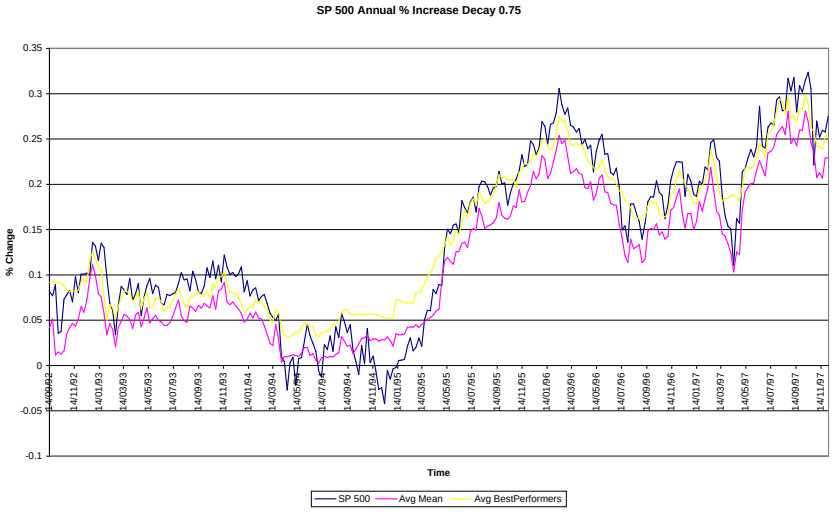

Fig. 2. Performance on the S&P 500 with a decay constant of 0.75.

Table 1. % profit for each decay setup.

	Decay Constant				S&P
	0.0	**0.25**	**0.75**	**1.0**	**500**
Best Individual Return (%)	70	59	40	24	43
Average Individual Return (%)	34	22	29	16	-

decay rate. However this trend is not as strong with the average performers, here a setting of 0.75 proved to produce more profitable individuals than 0.25 though still not as strong as the return of 34% from the 0.0 setup.

4 Conclusions & Future Work

This paper made initial inroads into the area of live evolution for the technical analysis of financial markets focusing in this case on a decay constant, which is used to determine the size of the investment for each trade. The results presented here show the viability of a strong decay constant as a solution. The potential of a Live GE system is also highlighted by the high yields of the experiments,

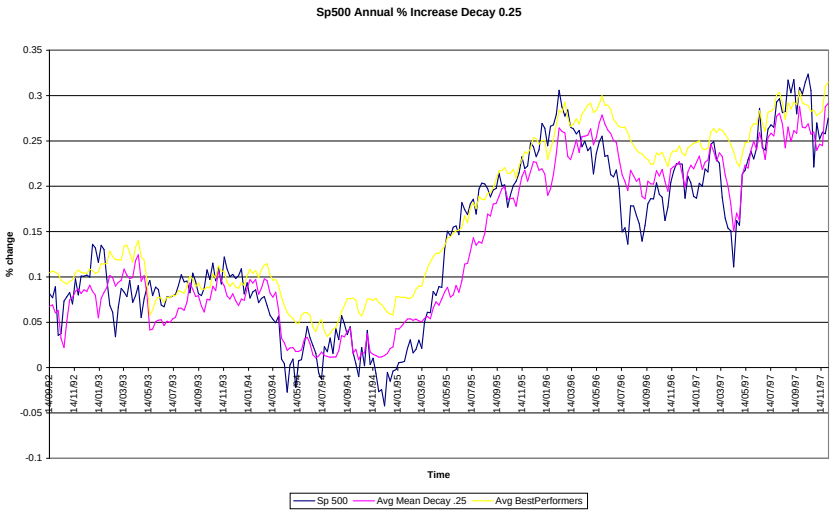

Fig. 3. Performance on the S&P 500 with a decay constant of 0.25.

which incorporated the strong decay constant, with an exceptional performance, yielding returns 27% greater than the buy and hold benchmark strategy on the S&P 500 for the 0.0 setting along with a strong average performance across the runs. It should also be born in mind however that the more aggressive decay settings may have benefited from a strongly upward trending S&P 500. Further experiments on indices with a differing behaviour will shed further light on this issue. Though these experiments have shown that continuity between trading window increments does not hold for the size of the investment made.

Further development, allowing the system to take short positions as well as the long positions will be considered as well as exploration in varying trading window sizes and whether this has a relationship to the number of generations evolved during live training. Also, the use of the N-Version paradigm [7] to incorporate different ranges as well as different grammars using a particular technical indicator each may have a positive influence on the returns made in the system.

Given the dynamic nature of the live trading environment the importance of population diversity can play an important role in the systems performance. Further experiments will be conducted to measure and analyse population diversity and its affect on performance.

References

1. O'Neill, M., Brabazon, A., Ryan, C., and Collins, J. (2001a). Developing a market timing system using grammatical evolution. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1375–1381, San Francisco, California, USA. Morgan Kaufmann.
2. O'Neill, M., Brabazon, A., Ryan, C., and Collins, J. J. (2001b). 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., and Tijink, H., editors, *Applications of Evolutionary Computing*, volume 2037 of *LNCS*, pages 343–352, Lake Como, Italy. Springer-Verlag.
3. O'Neill, M., Brabazon, A., and Ryan, C. (2002). Forecasting market indices using evolutionary automatic programming: A case study. In Chen, S.-H., editor, *Genetic Algorithms and Genetic Programming in Economics and Finance*. Kluwer Academic Publishers.
4. Brabazon, A. and O'Neill, M. (2002). Trading foreign exchange markets using evolutionary automatic programming. In Barry, A. M., editor, *GECCO 2002: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 133–136, New York. AAAI.
5. Brabazon, A. and O'Neill, M. (2004). Evolving Technical Trading Rules for Spot Foreign-Exchange Markets Using Grammatical Evolution. *Computational Management Science*. Springer, 2004.
6. Dempsey, I., O'Neill, M. and Brabazon, T. (2002). Investigations into Market Index Trading Models Using Evolutionary Automatic Programming, In *LNAI 2464, Proceedings of the 13th Irish Conference in Artificial Intelligence and Cognitive Science*, pp. 165-170, edited by M. O'Neill, R. Sutcliffe, C. Ryan, M. Eaton and N. Griffith, Berlin: Springer-Verlag.
7. Imamura, K., Heckendorn, R.B., Soule, T., Foster, J.A. (2002). N-Version Genetic Programming via Fault Masking. In *LNCS 2278, Proceedings of EuroGP 2002*, Kinsale, Ireland, pp. 172-181. Springer 2002.
8. Burke, E., Gustafson, S., Kendall, G. A Survey and Analysis of Diversity Measures in Genetic Programming. In *Proceedings of GECCO 2002, the Genetic and Evolutionary Computation Conference*, New York, USA, pp. 716-723, Morgan Kaufmann.
9. Koza, J.R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press.
10. O'Neill, M., Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers.
11. O'Neill, M. (2001). *Automatic Programming in an Arbitrary Language: Evolving Programs in Grammatical Evolution*. PhD thesis, University of Limerick, 2001.
12. O'Neill, M., Ryan, C. (2001) Grammatical Evolution, *IEEE Trans. Evolutionary Computation*, 5(4):349-358, 2001.
13. Ryan C., Collins J.J., O'Neill M. (1998). Grammatical Evolution: Evolving Programs for an Arbitrary Language. *Lecture Notes in Computer Science 1391, Proceedings of the First European Workshop on Genetic Programming*, 83-95, Springer-Verlag.
14. O'Neill, M., Ryan, C., Keijzer M., Cattolico M. (2003). Crossover in Grammatical Evolution. *Genetic Programming and Evolvable Machines*, Vol. 4 No. 1. Kluwer Academic Publishers, 2003.

15. Koza, J.R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press.
16. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D. (1998). *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann.
17. Koza, J.R., Andre, D., Bennett III, F.H., Keane, M. (1999). *Genetic Programming 3: Darwinian Invention and Problem Solving*. Morgan Kaufmann.
18. Koza, J.R., Keane, M., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers.
19. S-Lang. http://www.slang.org