Outperforming Buy-and-Hold with Evolved Technical Trading Rules: Daily, Weekly and Monthly Trading

Dome Lohpetch and David Corne
School of MACS, Heriot-Watt University
Edinburgh, UK
dl73@hw.ac.uk, dwcorne@macs.hw.ac.uk

Abstract. Genetic programming (GP) is increasingly popular as a research tool for applications in finance and economics. One thread in this area is the use of GP to discover effective technical trading rules. In a seminal article, Allen & Karjalainen (1999) used GP to find rules that were profitable, but were nevertheless outperformed by the simple "buy and hold" trading strategy. Many succeeding attempts have reported similar findings. There are a small handful of cases in which such work has managed to find rules that outperform buy-and-hold, but these have tended to be difficult to replicate. Recently, however, Lohpetch & Corne (2009) investigated work by Becker & Seshadri (2003), which showed outperformance of buy-and-hold. In turn, Becker & Seshadri's work had made several modifications to Allen & Karjalainen's work, including the adoption of monthly rather than daily trading. Lohpetch et al (2009) provided a replicable account of this, and also showed how further modifications enabled fairly reliable outperformance of buy-and-hold. It remained unclear, however, whether adoption of monthly trading is necessary to achieve robust outperformance of buy-and-hold. Here we investigate and compare each of daily, weekly and monthly trading; we find that outperformance of buy-and-hold can be achieved even for daily trading, but as we move from monthly to daily trading the performance of evolved rules becomes increasingly dependent on prevailing market conditions.

Keywords: genetic programming, technical trading rules, data mining

1 Introduction

There are several opportunities in the area of financial markets for advanced machine learning and optimization methods, and applications of evolutionary computation are now common in this area [1]. Genetic Programming (GP) [2,3,4] is a relatively popular technique in this field, with many studies reporting GP applications in finance (e.g. [5—12]). The focus in this paper is the area known as *technical analysis* [13—16]. Technical analysis is the name given to the general enterprise of

forecasting the future direction of equity prices via the study of historical market price data. Technical analysis relies on the principle that patterns and trends exist in markets, and that these can be identified (for example by discovering rules) and exploited to predict price movements in the near future.

Successful technical analysis uses tools such as moving averages (the mean price for a given stock or index over a given recent period), relative strength indicators (a function of the ratio of recent upward movements to recent downward movements), and others. A typical GP approach in this area is for rules to combine technical indicator 'primitives' with other mathematical operations. Such a rule constitutes a 'signal', which may be interpreted, for example, as a recommendation to buy if the signal is above a threshold. The first attempts to use GP in this way were by Chen and Yeh [5] and Allen and Karjalainen [7], and these and succeeding works regularly report that GP is able to find rules that are profitable on unseen future data. However, the 'elephant in the room' in such work has been a common and persistent failure for such rules to show greater returns than a standard "buy-and-hold" trading approach. The 'buy-and-hold' strategy is, for a given trading period, to buy the stock at the beginning of the period, and sell at the end – hence, always a good strategy in an upwardly moving market, and far simpler than using technical indicators.

Nevertheless, a small amount of research in this area seems able to find rules that outperform buy-and-hold [8,17,18]. In particular, GP-evolved technical trading rules with such success have been reported in Becker and Seshadri [19—21], who adopted the overall approach of Allen and Karjalainen [7] (who did not outperform buy-and-hold), and made several alterations. One of Becker & Seshadri's alterations was to adopt monthly trading rather than (as in Allen & Karjalainen) daily trading. That is, in [19], rules assume that trades will only be made (if at all) on the first day of the month, and hence deal with a less volatile view of the market. It is intuitively reasonable to suggest that this was an important feature of Becker & Seshadri's work, in the sense that outperformance of buy-and-hold may not have been achieved without this modification, however that hypothesis has not yet been tested. In this paper we test a modified version of Becker & Seshadri's approach and explore each of monthly, weekly and daily trading. We build on [24], which provided full details to enable replication of [19] as well as showing that a modified experimental setup led to more robust outcomes. In [24], outperformance of buy-and-hold was found to be robustly delivered by using Becker & Seshadri's approach [19] in the context of a monthly trading strategy, as long as the rules chosen for evaluation were selected according to performance over a validation period, and with the additional proviso that there was certainly some dependence on the specific data splits (training/validation/test) employed ([7] and [19] explored only one such data split). In the current work, we continue to evaluate this approach for several data splits, but in the context of weekly and daily trading too.

We end this section with a brief account of other related work which has attempted to outperform buy-and-hold. Potvin et al [12], for example, showed that GP trading rules can be generally beneficial in falling or stable markets – this is not particularly impressive, since buy-and-hold is naturally poor in such markets. In another line of work, risk metrics such as the Sharpe ratio [22] have been included in rules (or in their evaluation). Such metrics typically reduce the fitness of rules that promote trading in volatile conditions, and therefore lead to rules more likely to be

applied by investors. For instance, building on Fyfe et al. [6] (not superior to buy-and-hold), Marney et al. [8,17] included risk metrics, while Cheng and Khai [10] using a modified Sterling return measure, but none of these attempts produced usable rules that compared well in comparison to buy-and-hold.

The incorporation of risk measures nevertheless seems a promising thread of work, but in this paper we concentrate on further exploring the performance of GP for evolving robust technical trading rules. In the remainder, we detail the overall approach (section 2), and summarize the findings of several experiments in section 3; we then have a concluding discussion in section 4, and point to where the reader may obtain our code for further experimentation.

2 Evolving Robust Trading Rules: the Modified AK/BS Approach

2.1 Overview

The approach we use is based on Becker & Seshadri's work [19,21] (BS) which in turn was a modification of Allen & Karjalainen's work [7] (AK). This approach uses standard genetic programming (GP), with a function set comprising arithmetic, Boolean and relational operators, while the terminal set comprises basic technical indicators, along with real and Boolean constants, and real-valued variables (such as *stock price*). An example of a specific rule found by [19] is in Fig. 1.

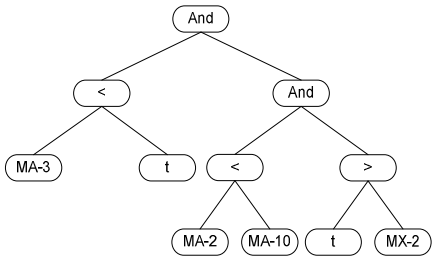

Figure 1. Example of a trading rule.

The rule in Figure 1 is to be interpreted as follows. **"**the 3-month moving average (MA-3) is less than the lower trend line (*t*) and the 2-month moving average (MA-2) is less than the 10-month moving average (MA-10) and the lower trend line (*t*) is greater than the second previous 3-month moving average maxima (MX-2)**"**. The rule therefore evaluates to either true or false. This translates into trading behaviour as follows: "If currently out of the market and the rule yields true, then *buy*; if currently in the market and the rule becomes false, then *sell*.". This procedure assumes a fixed amount to invest (e.g. $1,000) whenever there is a buy signal.

The remaining subsections explain the approach in further detail. Essentially we are explaining the approach in [19], making notes now and then to indicate where this departed from [7]. The data we use (as in [7,19—21] is the Standard and Poors 500 (S & P 500) index – a fixed set of 500 stocks which aggregate to daily price indicators (opening, closing, high, low). When considering weekly and monthly trading, the

opening price (for example) for a week or a month is the opening price on the first day of that week or month.

2.2 Function and Terminal Sets

The function set is: `and`, `or` and `not`, together with the relational operators > and <. We use strongly typed GP, automatically ensuring, for example, that relational operators receive Boolean inputs. The terminal set is as follows, where 'unit' is either day, week or month, depending on whether we are evolving rules for daily, weekly or monthly trading:

- opening, closing, high and low prices for the current unit;
- 2,3,5 and 10-unit moving averages;
- Rate of change indicator: 3-unit and 12-unit;
- Price Resistance indicators: the two previous 3-unit moving average minima, and the two previous 3-unit moving average maxima;
- Trend Line Indicators: a lower resistance line based on the slope of the two previous minima; an upper resistance line based on the slope of the two previous maxima.

The l-unit moving average at time m is the mean of the closing prices of the l units from m back to $m-(l-1)$. The l-unit rate of change indicator measured at time m is: $(p(m)-p(m-(l-1))\times 100)/p(m-(l-1))$, where $p(x)$ indicates the closing price for time x. Previous maxima MX1 and MX2 are obtained by considering the 3-unit moving averages at each point in the previous 12 units. Of the two highest values, the one closest in time to the current is MX1, and the other is MX2. The two previous minima are similarly defined. Finally, to identify trend line indicators, the two previous maxima are used to define a line in the obvious way, and the extrapolated value of that line from the current time becomes the upper trend line indicator; the lower trend line indicator is defined similarly by using the two previous minima.

2.3 The Fitness Function

The fitness function has three aspects. First is 'excess return', which is how much would have been earned by using the trading rule, in excess of the return from a buy-and-hold strategy. The excess return is $E = r - r_{bh}$, where r is the return on an investment of \$1,000, and r_{bh} is the corresponding return that would have been achieved using a buy and hold strategy. To calculate r we use [7,19,21]:

$$r = \sum_{t=1}^{T} r_t I_b(t) + \sum_{t=1}^{T} r_f(t) I_s(t) + n \ln\left(\frac{1-c}{1+c}\right)$$

where: $r_t = \log P_t - \log P_{t-1}$ – indicating the continuously compounded return, where P_t is the price at time t. Meanwhile, $I_b(t)$ indicates the buy signal, and is 1 if the rule indicates *buy* at time t, 0 otherwise. Similarly defined is the sell signal, $I_s(t)$. The first component of r therefore calculates the return on investment over the times when the investor is (as guided by the rule) in the market. In the second component, $r_f(t)$ indicates the risk-free return, which is taken for any particular day t from US Treasury bill data (these data are available from http://research.stlouisfed.org/fred/data/irates/tb3ms). Hence, the second component represents time out of the

market, assuming that the investor's funds are earning a standard risk-free interest. Finally, the third component corrects for transaction costs. The cost of a single buy or sell transaction is assumed to be 0.05% (i.e. 0.005) – e.g. $5 for a transaction of volume $1,000. The number of transactions actioned during the period by the rule is *n*. This component estimates the compounded loss from the expenditure on transactions.

The other two aspects of the fitness function, introduced in [19], are a modification that penalizes rule complexity, and a further modification that considered 'performance consistency' (PC). The second main part of the fitness function, r_{bh}, is calculated as:

$$r_{bh} = \sum r_t + \ln(\frac{1-c}{1+c})$$

where r_t is as indicated above. Hence it calculates the return of buying at the first day and selling at the last day of the period, involving a single buy and a single sell transaction.

The excess return *E*, calculated as described, was originally the objective function in [7], but improvements in [19,21] arose from two adjustments. One of these is an adjustment to fitness according to the size of the tree. Given a fitness value *f*, the adjusted fitness becomes $5f/\max(5,depth)$, where *depth* is the depth of the tree being evaluated, and the constant 5 is a 'desired' depth. Clearly there are many potential alternatives, but we simply adopt the stated method used in [19,21]. The other aspect of the fitness function which led to more consistent results was as follows, which we call *Performance Consistency* (PC). *E* is calculated for each successive period of *K* units covering the entire test period. The value returned is simply the number of these periods for which *E* was greater than both the corresponding buy and hold return (from investing in the index over that period) and the risk-free return during that period.

Finally we can state the objective function *f* used in this work: the fitness of a GP tree was the PC-based fitness (i.e. a number from 0 to *X*, where there were *X* periods covering the test data), adjusted for tree complexity by $5f/\max(5,depth)$.

2.4 Operators and Initialization

We used the four mutation operators described by Angeline [3], as follows:

- Grow: randomly select a leaf and replace with a randomly generated new subtree.
- Shrink: randomly select an internal node and replace the subtree below it with a randomly generated terminal node.
- Switch: randomly select an internal node and reorder its argument subtrees.
- Cycle: select a random node and replace it with a new node of the same type. If a terminal node is selected, it is replaced by a terminal node. If an internal node is selected, it is replaced by a function that takes an equivalent number of arguments.

We used standard subtree-swap crossover [2]. The population was initialized by growing trees to a maximum depth of 5, but no further constraint was placed on tree size during evolution, other than the pressure offered by the objective function.

3 Experiments

3.1 GP Parameters, Data Periods and Consistency-of Performance Periods

In all experiments, the GP approach was as described in section 2, with population size 500. In each generation, the current best was copied into the next generation, and the rest were the produced by crossover of two parents (probability 0.7) or mutation of a single parent. Parents were always selected via rank-based selection. Each run continued for 100 generations.

In common with [7] and [19], the period 1960—1991 was generally used for training in the monthly-trading case. In common with [24] we continued to explore two different regimes for choosing and evaluating a rule from the training run. In regime 1, the fittest rule found during training (as measured on the training set) was applied to test data in an immediately succeeding period of N years. In regime two, each rule found during training was validated against the ensuing N year period, and the rule that was best during this validation period was chosen, and tested over a further K years period beyond. These two regimes were each explored for four data period splits:

Table 1. Monthly splits

Name	Training Length	Validation Length (N)	Test Length (K)
MonthlySplit1	31 years	12 years	5 years
MonthlySplit2	31 years	8 years	8 years
MonthlySplit3	31 years	9 years	9 years
MonthlySplit4	25 years	12 years	12 years

For example, when regime 1 was used for data MonthlySplit1, the rule chosen is the best on the 31-year training period, and this rule is evaluated on the subsequent 12 year period. When regime 2 is used for this split, the rule found, while training on the 31-year period, which happened to be best on the subsequent 12 year period, was then evaluated on the further subsequent 5 year period. The results for Monthly splits 1, 2 and 3 were reported in [24], but are also summarized here to aid contrast and comparison with the daily and weekly trading results. Data periods for weekly and for daily trading were chosen to be reasonably consistent with the monthly splits, so that the numbers of days (weeks) involved corresponded with the number of months involved in the monthly splits. The details are as follows:

Table 2. Weekly splits

Name	Training Length	Validation Length (N)	Test Length (K)
WeeklySplit1	366 weeks from 1st Jan 1960	next 158 weeks	next 157 weeks
WeeklySplit2	366 weeks from 1st Jan 1972	next 158 weeks	next 158 weeks
WeeklySplit3	367 weeks from 1st Jan 1984	next 157 weeks	next 158 weeks
WeeklySplit4	366 weeks from 1st Jan 1996	next 157 weeks	next 158 weeks

Table 3. Daily splits

Name	Training Length	Validation Length (*N*)	Test Length (*K*)
DailySplit1	378 days from 1st Jan 1960	next 126 days	next 127 days
DailySplit2	380 days from 1st Jan 1975	next 127 days	next 127 days
DailySplit3	379 days from 1st Jan 1990	next 128 days	next 127 days
DailySplit4	376 days from 1st Jan 2006	next 128 days	next 126 days

The work reported in [19,21], building on Allen and Karjalainen's seminal approach [7], used what we refer to as regime 1 only, and MonthlySplit1. In [24] we confirmed that regime 2 led to more robust performance, however we continue to experiment with both regimes 1 and 2 here, to examine if that conclusion extends to the weekly and daily trading case. Figure 2 shows the four Monthly data splits aligned against the S&P 500 index for the period 1960—2008. We also show, in Figure 3, a representation of the returns from buy-and-hold for each data split. The market movements were net positive in each part of each split, indicating that outperforming buy-and-hold was in all cases a challenge.

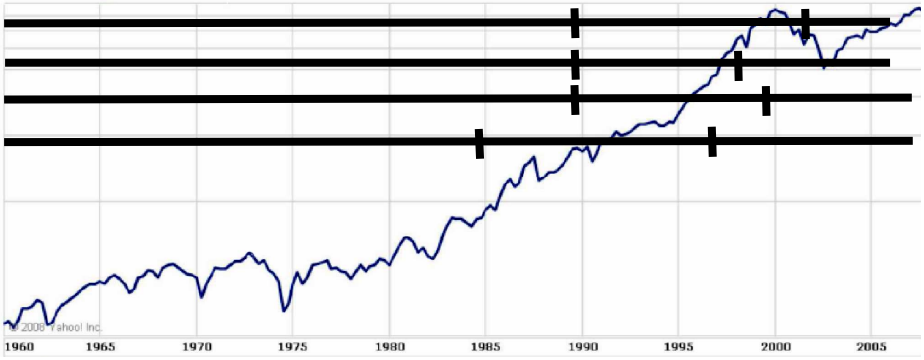

Figure 2. The S&P500 index over the period 1960—2008, illustrating the four data splits for the case of monthly trading.

We experimented also with the evaluation periods within the Performance Consistency (PC) term of the fitness function. In Becker and Seshadri's work, the PC term clearly results in improved performance (this is also true in our replication; we omit the comparative results for reasons of space). However they only report on the use of 12-month periods. We experiment with four different lengths for the "PC period" for each trading situation, namely: 6, 12, 18 and 24 months periods for monthly trading; 12 and 24 weeks for weekly trading, and 12 and 24 days for daily trading.

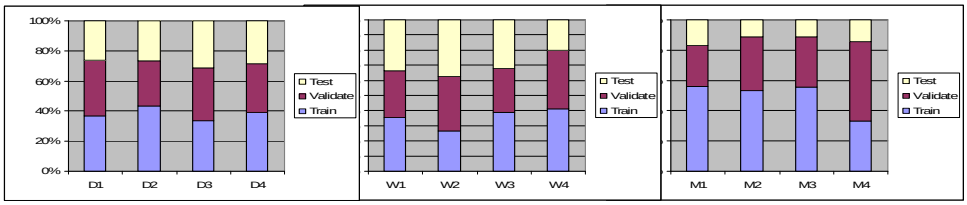

Figure 3. Characterising the buy-and-hold performance for each data split. Daily splits are on the left, weekly splits in the middle, and monthly splits on the right. Each bar shows relative proportions of the buy-and-hold performance in the training (lower), validation (middle) and test (upper) periods of the data split.

3.2 Results

For each trading period (monthly, weekly, daily), we performed 10 runs each for each combination of data split and consistency of performance period, and we report results for each of regime 1 and regime 2. To save space, we summarize each set of 10 runs in terms of the number of times that the result outperformed buy-and-hold. All results are summarized in Tables 4—6.

Table 4. Summary of results for monthly trading

Data split	PC Period	Eval. regime	Trials outperforming buy-and-hold.	PC Period	Eval. regime	Trials outperforming buy-and-hold.
Monthly Split1	6	1	10 out of 10	18	1	10 out of 10
		2	10 out of 10		2	10 out of 10
Monthly Split2	6	1	5 out of 10	18	1	4 out of 10
		2	9 out of 10		2	10 out of 10
Monthly Split3	6	1	9 out of 10	18	1	7 out of 10
		2	10 out of 10		2	9 out of 10
Monthly Split4	6	1	9 out of 10	18	1	6 out of 10
		2	10 out of 10		2	10 out of 10
Monthly Split1	12	1	10 out of 10	24	1	10 out of 10
		2	10 out of 10		2	10 out of 10
Monthly Split2	12	1	4 out of 10	24	1	4 out of 10
		2	8 out of 10		2	10 out of 10
Monthly Split3	12	1	10 out of 10	24	1	5 out of 10
		2	8 out of 10		2	7 out of 10
Monthly Split4	12	1	9 out of 10	24	1	5 out of 10
		2	10 out of 10		2	10 out of 10

As Table 4 shows, Monthly split 1 was clearly well-disposed to good performance. This split, evaluated with regime 1, was that used in [7] and [19], and perhaps gave an over-optimistic view of the general promise of the method. But it is clear from these results (and from [24]) that performance depends on the details of the data split. It is also clear that regime 2 is a better choice, leading to ideal performance also on Monthly split 4. If we now consider Figure 3, in attempt to understand relative performance in terms of the overall market movements in the data splits, we find that this is quite hard to do. Outperforming buy and hold would seem to be more likely when the performance of buy-and-hold in the test period is relatively weak, but this is not the case for Monthly splits 1 and 4. Referring to Figure 2, we see that the market

conditions were fairly similar for the training and validation parts of each monthly split, and were `up and down' for each of the four test periods.

Table 5. Summary results for weekly trading

Data split	PC Period	Eval. Regime	Trials outperforming buy-and-hold.	PC Period	Eval. regime	Trials outperforming buy-and-hold.
Weekly Split1	12	1	6 out of 10	24	1	2 out of 10
		2	2 out of 10		2	7 out of 10
Weekly Split2	12	1	10 out of 10	24	1	9 out of 10
		2	10 out of 10		2	5 out of 10
Weekly Split3	12	1	4 out of 10	24	1	3 out of 10
		2	4 out of 10		2	4 out of 10
Weekly Split4	12	1	10 out of 10	24	1	10 out of 10
		2	10 out of 10		2	10 out of 10

Table 5 shows the results, summarized in the same way, for the case of weekly trading, and Table 6 presents the corresponding results for the case of daily trading. These clearly show increasingly less robust results. It certainly seems that the method can find robust rules for Weekly trading that outperform buy-and-hold in some circumstances (splits 2 and 4), with less reliable performance in other cases. However, again, it seems we cannot discern any pattern that underpins this from the basic summary of the data splits' buy-and-hold performance in Figure 3. For daily trading, Table 6 shows that outperforming buy-and-hold is less likely, with strong performance in only one of the four data splits, and very poor performance in two of the data splits.

Table 6. Summary of results for daily trading

Data split	PC Period	Eval. regime	Trials outperforming buy-and-hold.	PC Period	Eval. regime	Trials outperforming buy-and-hold.
Daily Split1	12	1	0 out of 10	24	1	0 out of 10
		2	0 out of 10		2	0 out of 10
Daily Split2	12	1	0 out of 10	24	1	0 out of 10
		2	0 out of 10		2	0 out of 10
Daily Split3	12	1	10 out of 10	24	1	10 out of 10
		2	10 out of 10		2	9 out of 10
Daily Split4	12	1	2 out of 10	24	1	3 out of 10
		2	2 out of 10		2	4 out of 10

4 Concluding Summary and Discussion

The use of genetic Programming (GP) to induce technical trading rules remains an active thread of research. The most common outcome tends to be that, despite finding rules that seem successful on their own terms, they are usually not competitive with straightforward "buy and hold" (in upwardly moving markets) or the exploitation of risk-free investments (in downward markets). Building on Allen & Karjalainen's work in 1999 [7], however, Becker & Seshadri's approach [19] was one of few so far that have shown more promise. This was replicated in [24], with further advice on how reliably to generate effective rules. In particular, in [24] it was shown that the

approaches in [19,21] were somewhat sensitive to the data splits involved, and it is clearly better to use a validation set to choose the rule (regime 2).

However, this was in the context of monthly trading, one of a number of modifications made by Becker & Seshadri to the approach in [7], which used daily trading. It is reasonable to suppose that this might be a salient factor in the ability to beat a buy-and-hold strategy. We examined this by testing the Becker and Seshadri approach in the context of each of monthly, weekly and daily trading. We again found fairly robust generation of rules which outperform buy-and-hold for monthly trading, but with such being relatively rare for daily trading, while the situation for weekly trading is somewhere inbetween. In more detail, it seems that this approach is capable of finding rules that outperform buy-and-hold, even when tested in upwardly-moving markets, but the performance depends on the data split, and as we move from monthly to daily trading, this dependence on the data split seems to increase sharply. What is quite striking to us, however, is that it turns out to be very difficult to pin down the likelihood of success in advance based on simple summary analyses of the data splits. Visual analysis of the charts during the split periods (not shown here for space limitations), and summary measures such as Figure 3, so far fail to yield obvious indicators that might correlate with the possibility of evolving successful rules. This is a topic of continuing research, in which we are currently performing experiments with a large and varied collection of data splits. Finally, we note that interested researchers may pick up our source code at http://www.macs.hw.ac.uk/~dwcorne/gptrcode.

References

[1] Brabazon, A. & O'Neill, M. (2005). Biologically Inspired Algorithms for Financial Modelling (Natural Computing Series), New York: Springer.

[2] Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by means of Natural Selection, Cambridge, MA: MIT Press.

[3] Angeline, P. J. (1996). Genetic Programming's Continued Evolution. *Advances in Genetic Programming*, Vol. 2, editor, P. J. Angeline and K. Kinnear, Cambridge, MA: MIT Press, pp. 89-110.

[4] 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, San Francisco: Morgan Kaufmann

[5] Chen, S. H. & Yeh, C. (1996). Toward a Computable Approach to the Efficient Market Hypothesis: An Application of Genetic Programming, *J. of Econ Dynamics & Control*, 21: 1043-1063.

[6] Fyfe, C., Marney, J. P. & Tarbert, H. (1999). Technical Trading versus Market Efficiency: A Genetic Programming Approach, *Applied Financial Economics*, 9: 183-191.

[7] Allen, F. & Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules, *Journal of Financial Economics*, 51:245-271.

[8] Marney, J. P., Fyfe, C., Tarbert, H. & Miller, D. (2001). Risk Adjusted Returns to Technical Trading Rules: A Genetic Programming Approach, *Computing in Economics and Finance*, Soc. for Computational Economics, Yale University, USA, June 2001.

[9] Chen, S. H. (2002). Genetic Algorithms and Genetic Programming in Computational Finance, Boston, MA: Kluwer.

[10] Cheng, S. L. & Khai, Y. L. (2002). GP-Based Optimisation of Technical Trading Indicators and Profitability in FX Market, *Proc. 9th ICONIP* , Vol. 3, pp. 1159-1163.

[11] Farnsworth, G. V., Kelly, J. A., Othling, A. & Pryor, R. (2004). Successful Technical Trading Agents Using Genetic Programming, SANDIA Report SAND2004-4774, Sandia National Laboratories.

[12] Potvin, J. Y., Soriano, P. & Vallée, M. (2004) Generating Trading Rules on the Stock Markets with Genetic Programming, *Computers and Operations Research*, Vol. 31(7) (June 2004): 1033 - 1047

[13] Pring, M. J. (1980). *Technical Analysis Explained*, New York: McGraw-Hill.

[14] Ruggiero, M. A. (1997). *Cybernetic Trading Strategies*, NY: Wiley.

[15] Murphy, J. J. (1999). *Technical Analysis of the Financial Markets*, NY: New York Inst. of Finance.

[16] Lo, A. W., Mamaysky, H. & Wang, J. (2000). Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation, *J. of Finance*, 55:1705-1770, 2000.

[17] Marney, J. P., Miller, D., Fyfe, C. & Tarbert, H. (2000). Technical Analysis versus Market Efficiency: A Genetic Programming Approach, *Computing in Economics and Finance*, Society for Computational Economics, Barcelona, Spain, July 2000 (paper #169).

[18] Neely, C. (2001). Risk-adjusted, ex ante, optimal technical trading rules in equity markets, Working Papers 99-015D, Revised August 2001, Federal Reserve Bank of St. Louis.

[19] Becker, L.A. & Seshadri, M. (2003). Comprehensibility and Overfitting Avoidance in Genetic Programming for Technical Trading Rules, Worcester Polytechnic Institute, Computer Science Technical Report WPI-CS-TR-03-09.

[20] Becker, L.A. & Seshadri, M. (2003). Cooperative Coevolution of Technical Trading Rules, Worcester Polytechnic Institute, Computer Science Technical Report WPI-CS-TR-03-15.

[21] Becker, L.A. & Seshadri, M. (2003). GP-evolved technical trading rules can outperform buy and hold, In *Proc. 6th Int'l Conf. on Comp. Intelligence and Natural Computing*, North Carolina USA, 2003.

[22] Sharpe, W. F. (1966). "Mutual Fund Performance". *Journal of Business* **39** (S1): 119–138. doi:10.1086/294846.

[23] Marney, J. P., Tarbert, H. & Fyfe, C. (2005). Risk Adjusted Returns from Technical Trading: A Genetic Programming Approach, *Applied Financial Economics*, 15: 1073-1077.

[24] D. Lohpetch, D. Corne (2009) Discovering Effective Technical Trading Rules with Genetic Programming: Towards Robustly Outperforming Buy-and-Hold, in World Congress on Nature and Biologically Inspired Computing (NABIC) 2009, IEEE Press, to appear..