Soft Memory for Stock Market Analysis using Linear and Developmental Genetic Programming

Garnett Wilson and Wolfgang Banzhaf
Department of Computer Science
Memorial University of Newfoundland, St. John's
NL, Canada A1B 3X5
gwilson@cs.mun.ca, banzhaf@cs.mun.ca

ABSTRACT
Recently, a form of memory usage was introduced for genetic programming (GP) called "soft memory." Rather than have a new value completely overwrite the old value in a register, soft memory combines the new and old register values. This work examines the performance of a soft memory linear GP and developmental GP implementation for stock trading. Soft memory is known to more slowly adapt solutions compared to traditional GP. Thus, it was expected to perform well on stock data which typically exhibit local turbulence in combination with an overall longer term trend. While soft memory and standard memory were both found to provide similar impressive accuracy in buys that produced profit and sells that prevented losses, the softer memory settings traded more actively. The trading of the softer memory systems produced less substantial cumulative gains than traditional memory settings for the stocks tested with climbing share price trends. However, the trading activity of the softer memory settings had moderate benefits in terms of cumulative profit compared to buy-and-hold strategy for share price trends involving a drop in prices followed later by gains.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *Heuristic methods.*

General Terms
Algorithms, Performance, Experimentation.

Keywords
Financial analysis, linear genetic programming, developmental genetic programming.

1. INTRODUCTION
Evolutionary approaches, and genetic programming in general, have become a widely adopted means of analyzing stock data. In these systems, past market performance in terms of price and trading volume is analyzed using standard mathematical operators and established financial technical indicators to create rules or strategies to optimize profit. Linear genetic programming and coevolution-based developmental genetic approaches have been shown to perform well on individual stock analysis [5]. A recent study has adapted standard linear genetic programming so that when an assignment is made to a register of a LGP program, instead of completely overwriting the old value in the register, the soft assignment combines the new and old values [4]. This paper examines the affect of adding soft memory assignment to LGP when applied to the problem of stock analysis, and examines the use of soft assignment in a coevolutionary developmental GP system (PAM DGP). This work includes the first examination of soft memory in a stock analysis implementation, or soft memory usage in a coevolutionary or developmental GP system.

The following section examines related approaches to stock analysis and the concept of soft memory in the literature. Section 3 describes the developmental GP algorithm used in this work, PAM DGP, and the application of soft memory in LGP and PAM DGP. Section 4 discusses GP function set and application to stock analysis. Results are examined in Section 5, with Conclusions and Future Work following in Section 6.

GECCO'09, July 8–12, 2009, Montreal, Quebec, Canada.

2. RELATED WORK

2.1 Associated Genetic Programming Approaches to Stock Analysis
In general, genetic programming approaches have proven successful in a number of financial analysis applications, including analysis of individual stocks and larger portfolios. For instance, Yan et al. demonstrated that standard GP can outperform other machine learning methodologies such as support vector machines for application to portfolio optimization in very volatile markets, due to evolution of solutions that optimize profits instead of just predicting returns [7]. In terms of particular type of genetic programming implementation used, this work examines both a LGP and a coevoluationary developmental GP approach. Both linear GP and coevolutionary approaches have been applied to stock analysis. LGP was used by Grosnan *et al.* [3] previously in a study of Nasdaq and Nifty indices. In that study, multi-expression programming (MEP), LGP, and MEP/LGP ensembles outperformed neural networks and neuro-fuzzy implementations for prediction of interday stock prices. A co-evolutionary approach was applied to the creation of trading rules by Dreżewski and Sepielak [2], where two species represented entry strategies and exit strategies. In addition, a multi-agent version of the co-evolutionary algorithm and an evolutionary algorithm were examined. For the particular data set used by the authors, the multi-agent co-evolutionary approach generated the most profit. This paper represents the first analysis of the soft memory technique in a stock analysis problem domain and the first integration of soft memory in a developmental system.

This paper uses a technique reminiscent of the grammatical evolution (GE) approach of Brabazon and O'Neill [1]. In their work, following a period of initial training, the best evolved rules in the population were used to trade live for a window of *n* days. The window then shifted ahead and the current population was retrained on the data that was in the last window location, so the system could trade live on the following *n* days, and so on. The authors compared two versions of the GE system: one maintained its population across training periods and one re-initialized the population with each training period (window shift). The authors found that the former technique provided better trading rules that yielded greater profits. As detailed in the following section, our technique uses a shifting window of stock prices. Based on the results of [1], populations are not re-evolved with the shifting of each window in this work.

2.2 Soft Memory Allocation

Implementations examined in this work feature the ability to alter traditional GP memory behavior using "soft memory." Traditional GP, as is the case with actual computer hardware, replaces the contents of a particular address in memory with new contents, but never combines the two values. Indeed, to do so would render an actual computer virtually useless. From a biological point of view, however, systems build up in an incremental nature. To model this nature-based component of problem solving, McPhee and Poli employ what they call "soft memory," or "memory with memory" [4]. In this technique, instead of overwriting the old value in a register completely, the old and new value for the register are combined using weighted averaging of the old and new value for the register:

$$v_{combined} = \gamma v_{new} + (1-\gamma) v_{old} \quad (1)$$

where $v_{combined}$ is the actual, final value placed in the register, v_{new} is the new value being assigned to the register, and v_{old} is the original value in the register. γ is a constant providing "hardness" that determines how much the assignment operator affects the previous value in the register. When $\gamma = 1$, the assignment is completely "hard," and corresponds to the register assignment of traditional GP. For $\gamma = 0.5$, the value in the register would be the mean of the new and old values. For $\gamma = 0.5$, all registers are write-protected and all instructions behave like no-ops.

McPhee and Poli applied the linear GP with soft memory to several classes of polynomial regression problems of degree 4 to 15 in [4]. They found that soft memory almost always performed as well as traditional (linear) GP, and significantly outperformed it in several cases. Soft memory also exhibited less variation in best fitness achieved in each run compared to traditional GP. The authors found that soft memory made GP slowly refine approximate solutions over time, and it was less likely to find exact solutions compared to traditional GP. In particular, the authors examined γ values of 1.0, 0.7, 0.5, and 0.3, and found that in terms of both proportion of successful runs and distribution of best fitness values across all runs, $\gamma = 0.7$ outperformed all other thresholds, followed by 0.5, and lastly 1.0 and 0.3.

3. PAM DGP AND LGP

In addition to the more standard LGP algorithm, this work examines soft memory usage in a developmental GP system called PAM DGP. In PAM DGP [6], there are two populations, one of genotypes and one of mappings, that coevolve cooperatively. The algorithm is guided by a probability table with entries corresponding to each pair of individual genotype and mapping from both populations. The table entries correspond to frequencies that determine the probability that roulette selection in a steady state tournament will choose the genotype-mapping pairing corresponding to the indices of the table. The genotype and mapping individual in the current best genotype-mapping pairing are immune to mutation and crossover to avoid destroying the best solution yet discovered. A selection of four unique genotype-mapping pairings is made for each tournament round. Following fitness evaluation and ranking, the probability table columns corresponding to the winning genotype and mapping in the winning pair are updated using Equation 2 and the remaining combinations in that column are updated using Equation 3

$$P(g,m)_{new} = P(g,m)_{old} + \alpha(1 - P(g,m)_{old}) \quad (2)$$

$$P(g,m)_{new} = P(g,m)_{old} - \alpha(P(g,m)_{old}) \quad (3)$$

where *g* is the index corresponding to the genotype individual, *m* is index corresponding to the mapping individual, α is the learning rate (emphasis on current values versus previous search), and *P(g,m)* is the probability given in table element [*g*, *m*]. To prevent premature convergence, a noise threshold is implemented: If a table element exceeds the noise threshold following a tournament round, a standard Gaussian probability in the interval [0, 1] is used on that element and all values in its column are re-normalized so all values in the column sum to unity. The PAM DGP algorithm and selection mechanism are given in Figure 1.

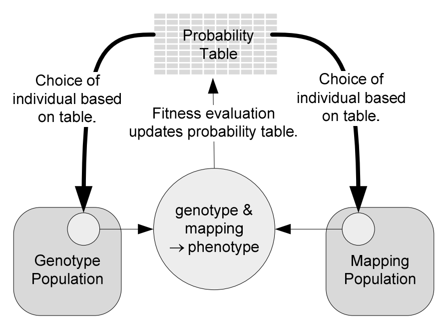

Figure 1. LGP and PAM DGP mapping techniques.

Genotypes in PAM DGP are in the form of binary strings, with interpretation of the binary sequences within the string being instruction-dependent (see Section 4). Mappings are redundant such that individuals are composed of $b \geq s$ 10-bit binary strings, where *b* is the minimum number of binary sequences required to represent a function set of *s* symbols. Each of the *b* 10-bit binary strings are interpreted as a decimal equivalent and normalized to the range [0…1] and mapped to an indexed member of an ordered function set by multiplying by *s* and truncating to an integer value. (This process results in a redundant encoding of symbols). Using this mapping mechanism with co-evolutionary selection, PAM DGP emphasizes the most useful members of the function set, ignores members of the function set which are not pertinent, and simultaneously evolves a genotype solution. In this work, PAM DGP is compared to a standard LGP implementation where both use soft memory. LGP individuals are also bit strings, and there is naturally only a genotype population. LGP can be seen as a special case of PAM DGP that uses a static mapping and constant function set. Effectively, PAM DGP extends LGP such that members of the function set are emphasized using an adaptive mechansim. The interpretation of instructions is the same for LGP as PAM DGP. Additional details of PAM DGP are presented in [6]. The PAM DGP mapping approach is detailed in Figure 2.

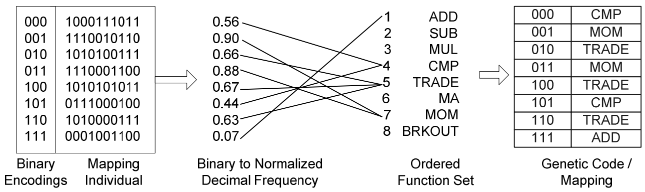

Figure 2. PAM DGP mapping technique.

Each steady state tournament consists of 1000 rounds, with 4 individuals chosen to compete per round. PAM DGP uses genotype and mapping populations each of size 10, with LGP using a genotype population of size 10. Each genotype consists of 320 bits with 4 registers, and each mapping consists of 160 bits (10 bits for each of the 16 encodings required to represent a function set of size 12). XOR mutation on a randomly chosen instruction (with uniform distribution) was the mutation mechanism applied to genotypes. Point mutation with a low threshold was used on mappings to provide a more stable context against which the genotype could evolve. Genotypes had a mutation rate of 0.5 and a crossover rate of 0.9. Mappings used a lower crossover and mutation rate, with both set to 0.1. PAM DGP was set to a conservative learning rate of 0.1 and noise threshold of 0.95 to prevent premature convergence. As in [4], we examine soft memory LGP and PAM DGP stock market analysis using four values of γ: 1.0, 0.7, 0.5, and 0.3.

4. FUNCTION SET AND APPLICATION TO STOCK ANALYSIS

The soft memory implementations of PAM DGP and LGP are applied to four stocks, two in the technology sector (Google Inc., ticker symbol "NASDAQ:GOOG", and Apple Inc., ticker symbol "NASDAQ:AAPL"), one in the energy sector (Chevron Co., ticker symbol "NYSE:CVX"), and one in the consumer/non-cyclical sector (PepsiCo Inc., ticker symbol "NYSE:PEP"). High, low, open, and close data was provided as input for 200 day periods between 2007 and 2008 chosen to highlight trends in stock prices. The first 16 days of the 200 day period were reserved to produce initial technical indicators. Following the initial 16 reserved days, GP fitness was evaluated on data corresponding to a moving window of 5 days. Individuals represent trading rules, based on the members of the function set (to be described). For each window of 5 days corresponding to trading days *m* to *n*, days *m* to *n-1* were used for evolution of a trading solution, with *m + 1* to *n* being used to evaluate the solution based on the immediately preceding day. Daily values used for the evolution of a trading solution were normalized using two-phase preprocessing as in [1]: All values were transformed by division by a lagged moving average, and then normalized using linear scaling into the range [0, 1] using

$$v_{scaled} = \frac{v_t - l_n}{h_n - l_n} \tag{4}$$

where v_{scaled} is normalized daily trading value, v_t is transformed daily trading value at time step t, h_n is highest transformed value in the last n time steps, l_n is the lowest transformed value in the last n time steps, and n is length of the time lag chosen for the initial transformation.

An individual consists of an instruction set, set of four registers, a flag for storing the current value of logical operations, and an output register for storing a result value corresponding to a trade recommendation. Following the execution of the instruction set (trading rules) of a GP individual, if the value of the trade register is 0, no action is recommended. Otherwise, the final value in the trade register corresponds to a value in the range [0, 1]. This value is multiplied by a maximum dollar amount to be bought or sold per trade. (In these experiments, $10 000 was used based on an initial account balance of $100 000 to give some portion of $10 000 to be traded.) For each trade conducted, there is a $10 commission penalty. The trading system is allowed to have a small deficit >= $10 to handle a sell recommendation when maximally invested (where the deficit would be immediately recouped) or to allow a buy that results in state of maximal investment. Fitness of an individual is the total value of the cash and shares currently held.

The best individual, having the best trading rules, is used by a "live" trading algorithm. There is a live trading system that provides known information to the GP for days *m* to *n*. The GP algorithm returns a trading recommendation that the live trading system then follows on the next day, *n + 1*. In particular, the net number of shares bought and sold by the best evolved individual (trading solution) given the recommendations over all the evaluation cases (4 cases given a 5 day window) is the buy or sell recommendation to the "live" trading system. The best GP individual can thus recommend up to $40,000 worth of shares be bought or sold on an actual trading day by the live system. When the sliding window shifts, the current cash and shares held by the "live" trading system are provided to the GP as starting conditions for the next tournament where trading solutions are based on the daily values in the new window. The trading actions of the live trading system are used to take action on unknown share values for the day that has not occurred yet, and determines the success of the algorithm. The trading system is summarized in Figure 3.

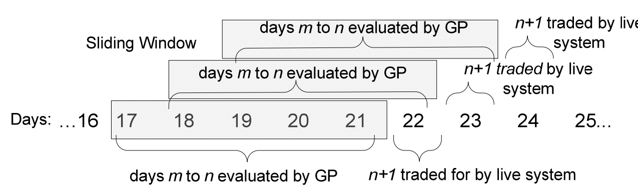

Figure 3. "Live" trading system and GP evolution of trades.

The function set includes basic mathematical operators (+, -, *) and logical operators (<, >, =). In addition, there are established financial analysis metrics such as moving average, momentum, channel breakout (the trading range of a stock using +/- 1 or 2 standard deviations as a metric of volatility), and current day high, low, open, or close price.

5. RESULTS

5.1 Value of Shares and Cash

The value of the assets (shares and cash) held by the live trading system for each of 184 days of trading is examined. (As mentioned in Section 4, 200 fitness cases were used overall, with the first 16 cases reserved for establishment of initial technical indicators.) Fifty trials for each of the four stocks were performed using an Apple iMac Intel Core 2 Duo 2.8 GHz CPU with 4GB RAM running OS X Leopard v10.5.4. Given this hardware configuration, a trial over the entire 180 day time period typically takes under 5 minutes for LGP and under 10 minutes for PAM DGP. Trading commences with $100,000 in cash with which to invest in shares of each stock. The mean worth of the live trading system for LGP and naïve buy-and-hold strategies is given in Figures 4 to 7. Mean worth of PAM DGP and buy-and-hold are shown in Figures 8 to 11. (Buy-and-hold is the strategy where a user simply purchases as much stock as possible with their initial amount and stays fully invested for the entire time period.) Please note different scales are used to make comparisons easier to view.

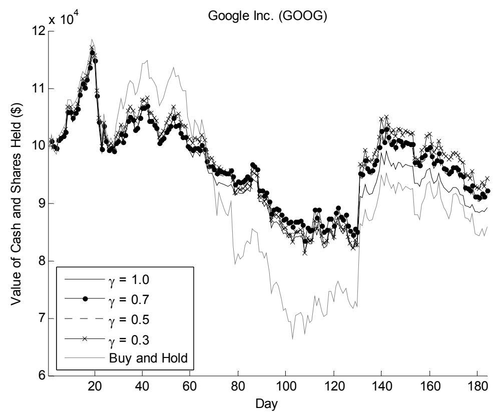

Figure 4. Mean total worth (cash and shares) for buy-and-hold and LGP with γ = 1.0, 0.7, 0.5, and 0.3 for GOOG.

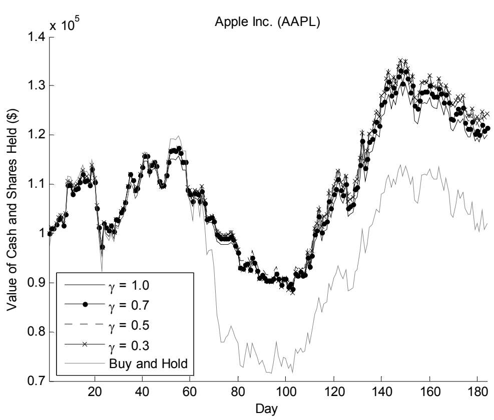

Figure 5. Mean total worth (cash and shares) for buy-and-hold and LGP with γ = 1.0, 0.7, 0.5, and 0.3 for AAPL.

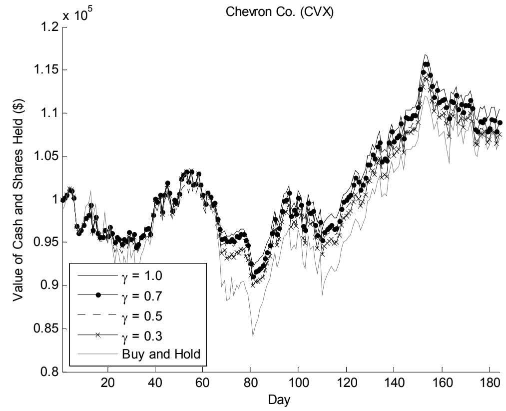

Figure 6. Mean total worth (cash and shares) for buy-and-hold and LGP with γ = 1.0, 0.7, 0.5, and 0.3 for CVX.

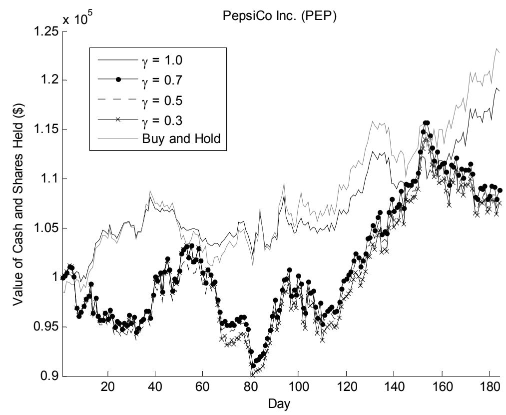

Figure 7. Mean total worth (cash and shares) for buy-and-hold and LGP with γ = 1.0, 0.7, 0.5, and 0.3 for PEP.

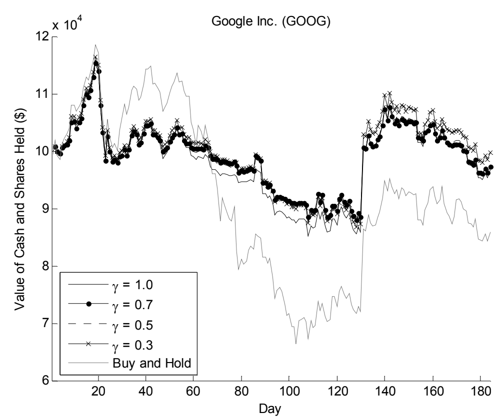

Figure 8. Mean total worth (cash and shares) for buy-and-hold and PAM DGP with γ = 1.0, 0.7, 0.5, and 0.3 for GOOG.

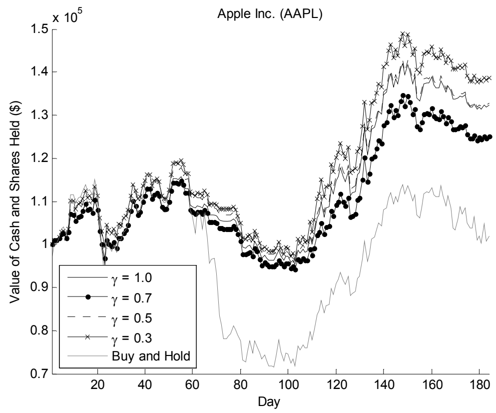

Figure 9. Mean total worth (cash and shares) for buy-and-hold and PAM DGP with γ = 1.0, 0.7, 0.5, and 0.3 for AAPL.

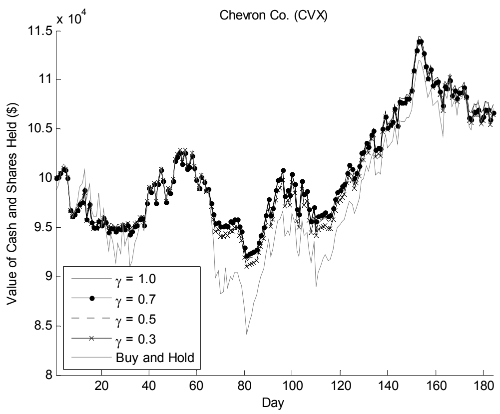

Figure 10. Mean total worth (cash and shares) for buy-and-hold and PAM DGP with γ = 1.0, 0.7, 0.5, and 0.3 for CVX.

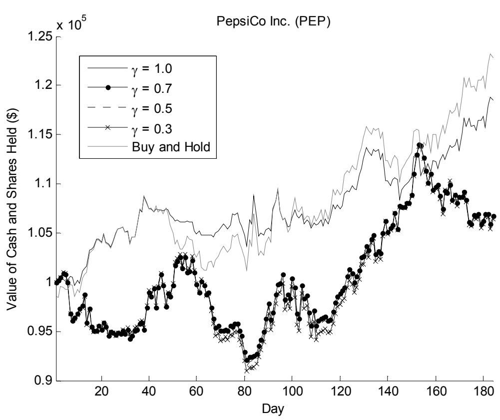

Figure 11. Mean total worth (cash and shares) for buy-and-hold and PAM DGP with γ = 1.0, 0.7, 0.5, and 0.3 for PEP.

Although it is difficult to discern exact differences in performance for levels of soft memory in these graphs, it is evident that for GOOG, AAPL, and CVX all settings of soft memory perform relatively closely, with γ = 1.0 being slightly outperformed by the softer settings for GOOG and APPL. However, for CVX, γ = 1.0 outperforms the softer settings. PEP is unusual in that an upward climbing trend makes the naïve buy-and-hold strategy best. In this scenario, γ = 1.0 significantly outperforms the other soft memory thresholds.

The overall trend of similar behavior among memory settings for PAM DGP in Figures 8 to 11 is evident. There is a greater divergence in profitability of the soft memory settings in AAPL near the end of the time period than for LGP. The extent to which the softer settings outperformed γ = 1.0 for the other stocks, if at all, is unclear at this level of plot refinement. Accordingly, a more detailed comparison of the γ settings of both LGP and PAM DGP is now provided in Section 5.

5.2 Comparative Value Performance

A more exact comparison of the performances in the Section 5.1 can be achieved by examining the ratio of performance of the soft memory settings to the hardest setting. In particular, the softer settings of γ are compared directly as ratios to the hardest setting (γ = 1.0) corresponding to traditional LGP (Figures 12 to 15) and traditional PAM DGP (Figures 16 to 19) over 50 tirals. In all cases, values greater than 1 indicate greater γ worth than standard LGP (γ = 1.0), values less than 1 *vice versa*.

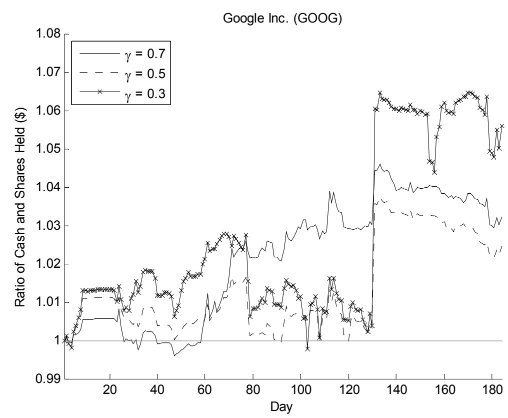

Figure 12. Mean ratios of total worth (cash and shares) for γ = 0.7, 0.5 and 0.3 to γ = 1.0 for GOOG using LGP.

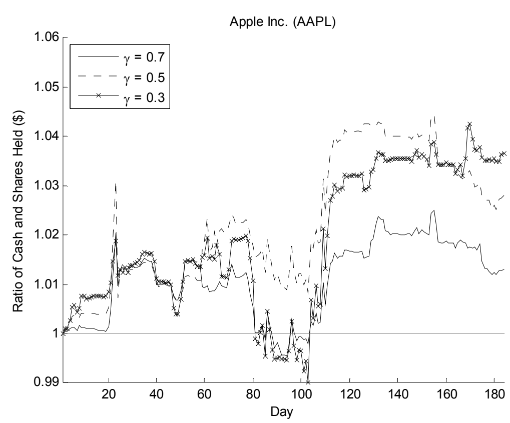

Figure 13. Mean ratios of total worth (cash and shares) for γ = 0.7, 0.5 and 0.3 to γ = 1.0 for AAPL using LGP.

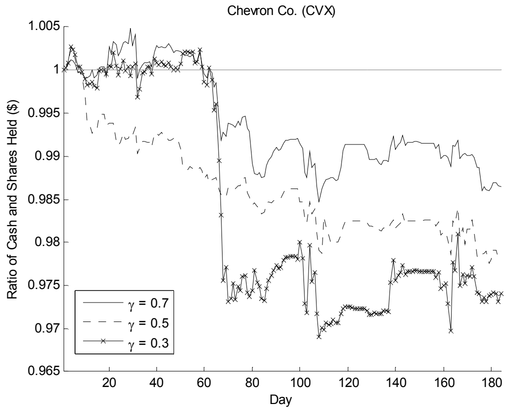

Figure 14. Mean ratios of total worth (cash and shares) for γ = 0.7, 0.5 and 0.3 to γ = 1.0 for CVX using LGP.

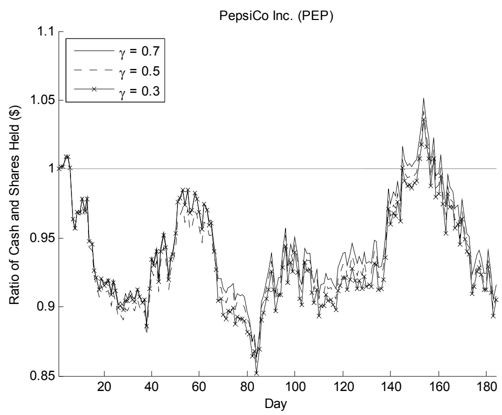

Figure 15. Mean ratios of total worth (cash and shares) for γ = 0.7, 0.5 and 0.3 to γ = 1.0 for PEP using LGP.

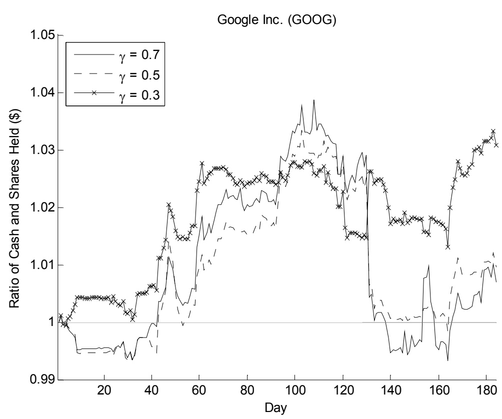

Figure 16. Mean ratios of total worth (cash and shares) for γ = 0.7, 0.5 and 0.3 to γ = 1.0 for GOOG using PAM DGP.

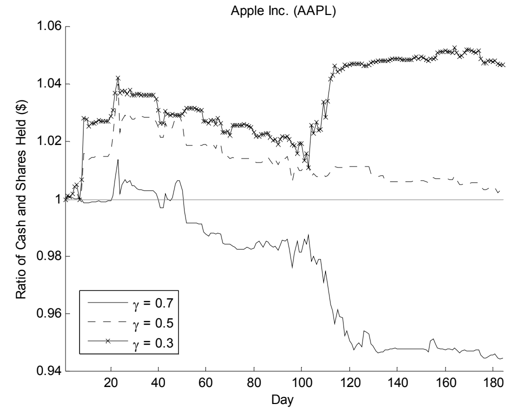

Figure 17. Mean ratios of total worth (cash and shares) for γ = 0.7, 0.5 and 0.3 to γ = 1.0 for AAPL using PAM DGP.

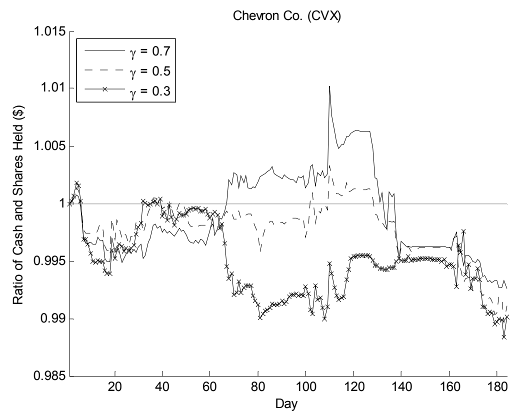

Figure 18. Mean ratios of total worth (cash and shares) for γ = 0.7, 0.5 and 0.3 to γ = 1.0 for CVX using PAM DGP.

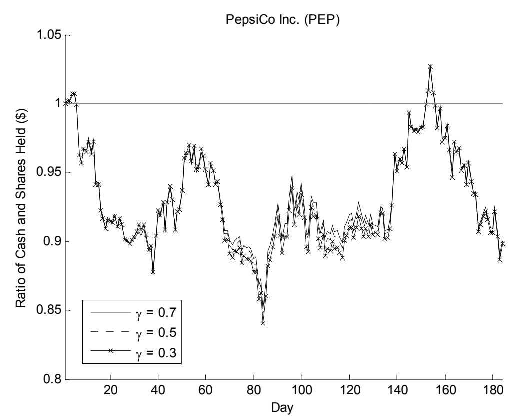

Figure 19. Mean ratios of total worth (cash and shares) for γ = 0.7, 0.5 and 0.3 to γ = 1.0 for PEP using PAM DGP.

Based on analysis of LGP, Figures 12-15, the softest memory settings outperform standard LGP (γ = 1.0) for GOOG and AAPL (Figures 12-13) by a moderate amount (up to 7% and 4%, respectively). Both of these price trends (using buy-and-hold to indicate change in stock price) feature substantial losses after initial investment followed by moderate gain in share price. In the instances of CVX and PEP (Figures 14-15), where a general upward trend in stock price is prevalent, the softer memory settings did not outperform standard LGP. In fact, the softest settings performed up to 3.5% worse for CVX and up to 15% worse for AAPL. The PAM DGP comparative performance (Figures 16-19) show the same trends: the softest settings showed moderate improvement for GOOG and AAPL (up to 4% and 6%, respectively), and performed worse than standard memory PAM DGP for CVX and PEP (by margins of up to approximately 1.5% and 15%, respectively).

5.3 Profit Analysis

Spread of data in terms of final profit, and cumulative profit better than buy-and-sell over time, is presented in the boxplots of Figures 20 and 21, respectively. A value of 0 indicates the breakeven point, and final percentage can be multiplied by starting amount to determine dollar value of profit.

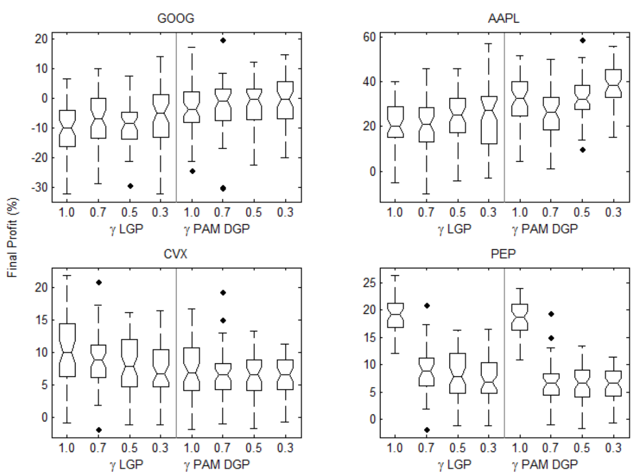

Figure 20. Mean final profit (%) for γ settings of PAM DGP and LGP.

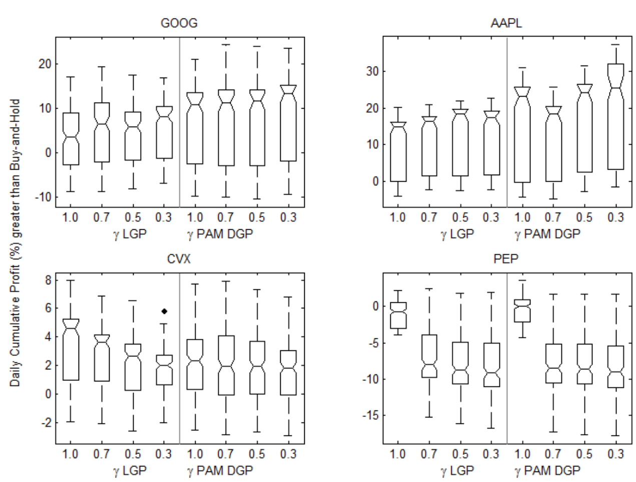

Figure 21. Mean daily cumulative profit (%) greater than buy-and-hold for γ settings of PAM DGP and LGP.

In terms of final profit (Figure 20), we can see that for both LGP and PAM DGP for GOOG, AAPL, and CVX there is no statistically significant difference between standard implementations ($\gamma = 1.0$) and soft memory implementations ($\gamma = 0.7$, 0.5, and 0.3). However, generally all algorithms were profitable for well-chosen stocks, in some instances with profits as high as almost \$60 000 for AAPL, over \$20 000 for CVX, and over \$25 000 for PEP on the initial \$100 000 investment. (Notches around the median in all boxplots indicate confidence intervals of 95%, where non-overlapping notches around the medians when comparing boxplots indicate statistically significant difference at the 95% confidence interval in the remainder of this paper.) For PEP, on the other hand, standard implementations dramatically outperform the softer memory settings. In fact, the softer memory settings cause significant losses at points in time in a scenario that has the potential for dramatic gains during the entire time period (compare to Figures 7 and 11). For cumulative profit outperforming buy-and-hold, Figure 21, we see that for LGP the softest memory settings ($\gamma = 0.5$ and 0.3) can slightly outperform that standard. However, given LGP for CVX and PEP, all softer memory settings ($\gamma = 0.7$, 0.5, and 0.3) do not perform as well as the standard implementation. Given PAM DGP, there is no statistical difference in performance of γ thresholds for GOOG, AAPL, or CVX. Cumulative profits of PEP, as for final profits, indicate dramatic underperformance of soft memory ($\gamma < 1$) compared to the standard ($\gamma = 1$).

5.4 Trading Analysis

Proportion of profitable trades is a common metric for evaluation of trading performance, but it can mask other elements of performance that are prudent to consider. For instance, the metric does not even indicate an algorithm's ability to generate actual profit [1]. Many trades are not profitable, but are highly beneficial in preventing loss when share price drops. Thus, rather than percentage of profitable trades, the percentage of profitable buy trades and percentage of sell trades preventing loss are provided in Figures 22 and 23, respectively. Trading activity, as the percentage of all trading opportunities where a trade was executed, is shown in Figure 24.

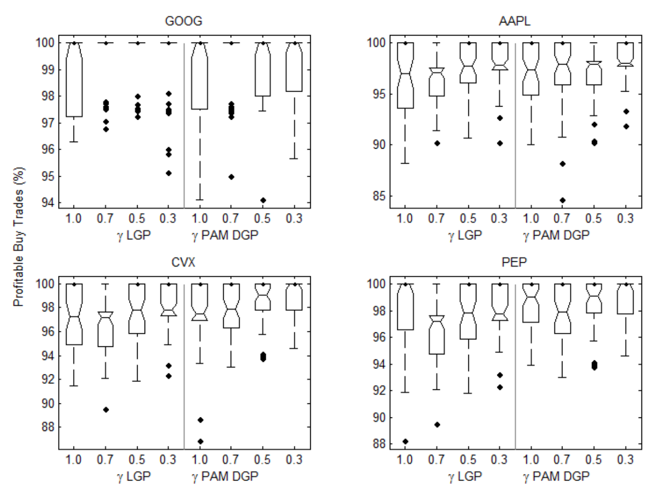

Figure 22. Percentage of profitable buy trades for γ settings of PAM DGP and LGP.

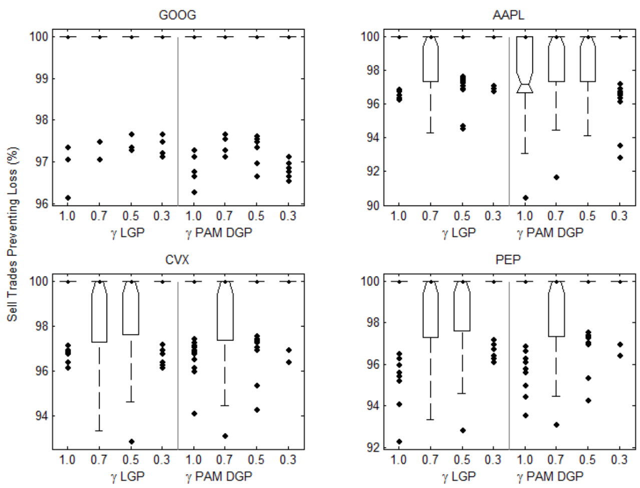

Figure 23. Percentage of sell trades preventing losses for γ settings of PAM DGP and LGP.

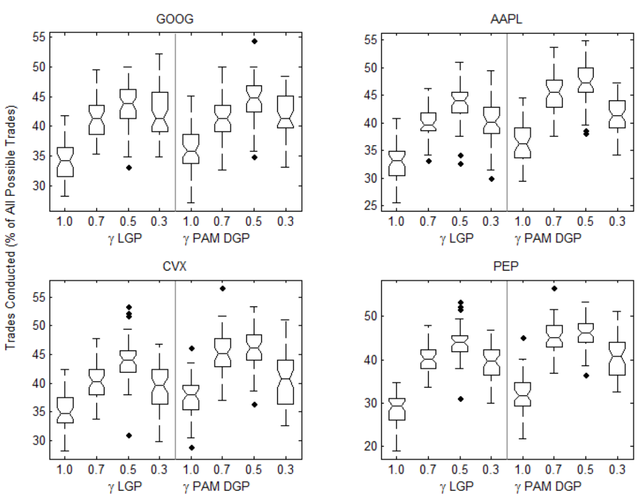

Figure 24. Percentage of trades executed out of all trading opportunities for γ settings of PAM DGP and LGP.

Figure 22 shows that for LGP and PAM DGP, across all soft memory settings, there is no statistical difference in number of profitable trades for GOOG, APPL, and CVX. For LGP with PEP, soft memory did not achieve the same number of successful trades as the standard implementation. PAM DGP with PEP did not show any statistically significant difference. Overall, the percentage of profitable buy trades across all settings was high. Examining trades made to prevent losses (Figure 23), all implementations exhibited medians at or near 100% with no statistically significance difference among them, with the exception of standard PAM DGP for AAPL (but the median was still very high at approximately 97%). The success of the parameterizations in terms of profitable buys and protective sells are high in every memory, algorithm scenario and thus do not seem to relate to general profitability (compare to Figures 20-21). Figure 24 shows that for LGP and PAM DGP, the softer memory settings ($1 > \gamma$) traded more actively than standard memory, with $\gamma = 0.5$ exhibiting the most active trading for all algorithms. This finding was statistically significant except for PAM DGP with CVX and PEP (but $\gamma = 0.5$ median was still highest).

6. CONCLUSIONS

Soft memory ($\gamma < 1$), in both linear GP and developmental GP implementations, resulted in more active trading systems than standard memory ($\gamma = 1$). While softer memory did not result in lower numbers of profitable buys or sells stopping losses compared to standard memory in the vast majority of cases, this did not necessarily translate into greater profitability. Given the active trading exhibited by the softer memory settings combined with profitability analysis, this activity is likely related to greater profitability for GOOG and AAPL in taking advantage of lower share prices to sell higher. However, this active trading resulted in losses when a volatile but gradual (CVX) or steady climb (PEP) in share price occurred (compare soft memory to buy-and-hold trends in Figures 4-11). Based on analysis of final and (especially) cumulative profitability, the moderate gains provided by soft memory for certain stocks/trends do not outweigh the more substantial lack of performance they incur when they fail to perform as well as traditional memory. For diverse function sets such as those used to analyze the stock market, it may not be appropriate to meld new contents for a register with its old contents if that content pertained to a different context. Future work will examine the effect of such context changes, and the trend conditions under which soft memory ought to be used.

7. REFERENCES

[1] Brabazon, A. and O'Neill, M. *Biologically Inspired Algorithms for Financial Modelling*. Springer , Berlin, 2006.

[2] Drezewski, R. and Sepielak, J. Evolutionary System for Generating Investment Strategies. *Applications of Evolutionary Computing (EvoWorkshops 2009)*, Springer (2008), 83-92.

[3] Grosan, Crina and Abraham, Ajith. Stock Market Modeling Using Genetic Programming Ensembles. *Studies in Computational Intelligence 13*, (2006), 131-146.

[4] McPhee, N. and Poli, R. Memory with Memory: Soft Assignment in Genetic Programming. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, ACM Press (2008), 1235-1242.

[5] Wilson, G. and Banzhaf, W. Interday Stock Prices using Developmental and Linear Genetic Programming. *Applications of Evolutionary Computing (EvoWorkshops 2009)*, Springer (2009), upcoming.

[6] Wilson, G. and Heywood, M. Introducing Probabilistic Adaptive Mapping Developmental Genetic Programming with Redundant Mappings. *Genetic Programming and Evolvable Machines 8*, 2 (2007), 187-220.

[7] Yan, W., Sewell, M., and Clack, C. Learning to Optimize Profits Beats Predicting Returns —Comparing Techniques for Financial Portfolio Optimisation. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, ACM Press (2008), 1681-1688.