Brill Academic Publishers
P.O. Box 9000, 2300 PA Leiden,
The Netherlands

Lecture Series on Computer and Computational Sciences
Volume 1, 2006, pp. 1-3

Genetically Programmed Trading Rules for the Foreign Exchange Market

N.G. Pavlidis†1, E.G. Pavlidis‡ and M.N. Vrahatis†

†Computational Intelligence Laboratory, Department of Mathematics,
University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR-26110 Patras, Greece.

‡Department of Economics,
Lancaster University Management School, Lancaster LA1 4YX, UK.

Abstract: The identification of price patterns and trends, and the formation of rules to generate market signals have a long history in foreign exchange rate markets. Recent studies, however, question the profitability of the simple rules that have been shown to yield abnormal profits in previous decades. Rather than assuming a fixed set of rules, in this paper we employ genetic programming to identify rules in the Euro US Dollar daily exchange rate series over the period 1/1/1999 to 30/12/2005. Preliminary experimental results suggest that genetic programming is capable of generating profitable rules but for a limited time into the future.

Keywords: Genetic Programming, Daily Exchange Rate Time Series, Trading Rules

Mathematics Subject Classification: 68T05, 91B28, 91B84

1 Introduction

Technical Analysis (TA) focuses on the identification of price patterns and trends, as well as, the use of mechanical rules to generate valuable economic signals [8]. Recent surveys (see [1] and the references therein) indicate that since the inception of floating exchange rates TA has been a major constituent of financial practice in foreign exchange markets. Moreover, a number of studies during the 1980s and the early 1990s suggest that the application of simple technical trading strategies (mainly moving average and filter rules) can yield returns net of transaction costs above 5% per annum. Olson [6] argues that these abnormal profit opportunities are due to temporary inefficiencies which are in accordance with an evolving market and that simple trading rule returns have declined, if not disappeared, since then. Olson's view, however, does not rule out the existence of more sophisticated profitable strategies.

Genetic programming (GP) is a domain-independent evolutionary search method that explores a space of computer programs [3]. GP evolves a population of computer programs using genetic operations inspired from Darwinian evolution to identify programs that solve a given task. The advantages of GP for the identification of trading strategies are twofold. First, it allows the construction of rules of arbitrary complexity. Second, by avoiding the ex post specification of the trading rules that are examined, it circumvents a basic, but rarely addressed issue, namely *data-snooping* [8]. Neely *et al.* [5] employ GP to identify profitable trading rules in several daily foreign

[1]Corresponding author. e-mail: npav@math.upatras.gr

exchange rate series. Their findings suggest that trading rules obtained through GP substantially outperform simple trading rules for daily exchange rates. In this study, we employ GP to identify trading rules in the daily exchange rate of the Euro against the US Dollar. Our preliminary experimental results suggest that GP is capable of identifying profitable rules in the training set, but the noisy nature of the time series places limits on the profitability of these rules as we move further into the future. GP, however, was capable of identifying rules whose performance for a period in the future equal in length to the training period, was as good as their performance on the training dataset.

2 Genetic Programming

GP is an extension of Genetic Algorithms (GAs) in which individuals are computer programs expressed as *syntax trees*, rather than fixed length strings. The internal nodes of syntax trees are function nodes. A function node applies one of the user–defined functions to the outputs of its children. The leaves of a syntax tree are terminal nodes. Terminal nodes return as output the value of a constant, an input variable, or a zero–argument function [3].

A critical component of GP is the random-tree creation algorithm(s) it employs. The standard GP initialization technique, *ramped half and half*, relies on two tree-generation algorithms, *GROW* and its full-tree variant, *FULL* [3]. GROW is by far the most commonly employed random-tree generation mechanism, not only for the initialization of individuals, but also for *subtree mutation* [4]. To avoid the shortcomings of GROW a recently proposed algorithm, PTC2 [4], was used instead to construct random trees. The GP operators employed in this study were *proportionate selection*, *reproduction*, *subtree mutation* and uniform crossover. A flowchart of the operation of GP is provided in Fig. 1. According to proportionate selection, the probability of selecting individual i is equal to $E_i/\sum_{j=1}^{N} E_j$, where E is the function that is to be maximized, and E_i is the function value that corresponds to the ith individual. Reproduction inserts a copy of the selected individual to the population of the next generation. Subtree mutation, on the other hand, substitutes a randomly selected subtree with a new subtree. Crossover is the primary GP search operator. We employed the *uniform crossover* (GPUX) operator [7]. At early stages of the algorithm GPUX favors global search by swapping large subtrees near the root, while as the population converges it becomes more and more local in the sense that, the offsprings it produces are progressively more similar to their parents. GPUX starts by identifying the common region between the two syntax trees. Each node that lies in the common region undergoes crossover with a fixed probability. For nodes that lie in the interior of the common region uniform crossover swaps the nodes without affecting the subtree below them. On the contrary, for nodes on the boundary of the common region the subtrees rooted at these nodes are swapped.

3 Experimental Results and Discussion

The dataset used in this study was the daily closing prices for the Euro US Dollar exchange rate from Barclays Bank International provided by Datastream. The 1825 observations cover the period from January 4th 1999 to December 30th 2005. After normalization and embedding a total of 1525 patterns was available. The first 500 patterns were assigned to the training set, while the remaining 1025 were assigned to the test set. The computational experiments were performed using a C++ implementation based on the interface for the creation of expressions described in [2, Chap.8] and the GNU compiler collection (gcc) version 4.0.3. The terminal set, $\mathcal{T}$, consisted of, $\mathcal{T} = \{x_t^n, x_{t+1}^n, \ldots, x_{t+50}^n, \text{Rand}\}$, where x_t^n stands for the daily exchange rate at date t divided by the 250-day moving average, and Rand denotes a random constant in the interval $[-50, 50]$. The function set used was, $\mathcal{F} = \{+, -, *, /, \texttt{OR}, \texttt{AND}, \leqslant, \geqslant, ==, \sin, \cos, \text{sqrt}, \log, \exp, \max, \min, \text{average}\}$.

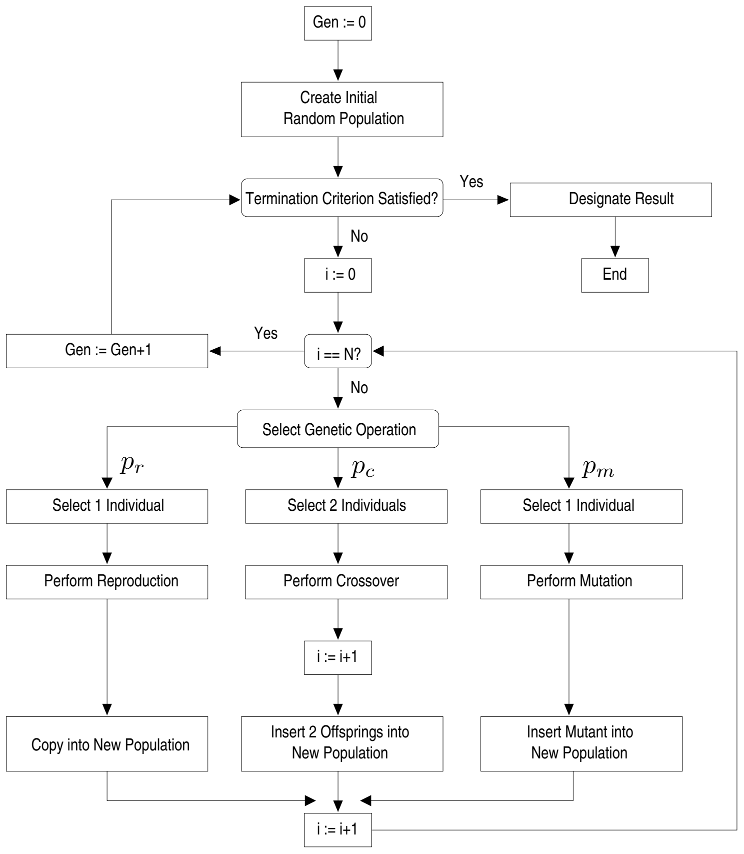
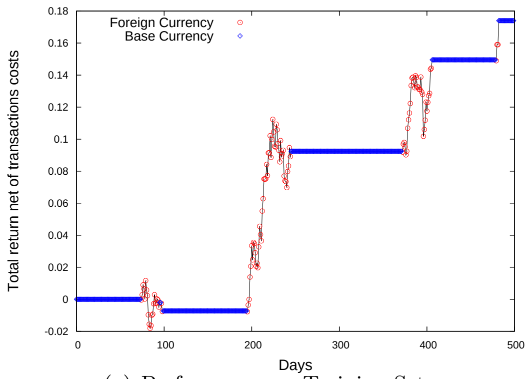


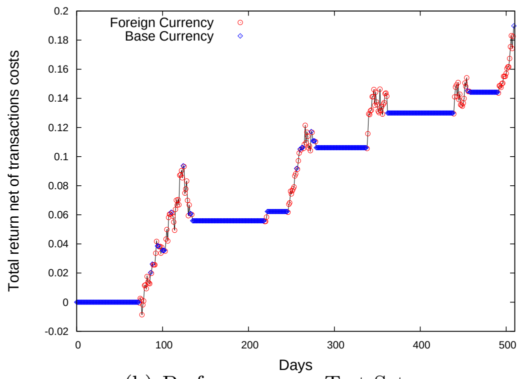
Figure 1: Flowchart of Genetic Programming.

A positive evaluation of an individual over a pattern was assumed to signal that the current holdings should be held in the base currency (in this case dollars), and vice versa. The fitness measure of each individual was the rate of return over the training period. A one-way transaction cost equal to 0.5% and 0.05% for the training, and test period, respectively, was used. The choice of 0.05% is in accordance with the cost faced by large institutional investors. A larger transactions cost was imposed during training to penalize rules that trade very frequently, and hence discourage overfitting [5]. Regarding the parameters of the GP algorithm, the maximum tree depth at initialization was set to 5, while the maximum tree depth in all other generations was set to 8. The maximum number of nodes was set to 50. Population size was 5000 and the maximum number of generations was 500. The reproduction, mutation, and crossover probabilities were $p_r = 0.1$, $p_m = 0.2$, and $p_c = 0.7$, respectively. Finally, the probability of performing uniform crossover at a node was 0.5. The output of the algorithm was the individual with the highest fitness encountered during its execution.

Initially 30 experiments were performed using the aforementioned configuration. GP identified highly profitable rules in the training set, but their performance on the test set was very poor. To investigate the impact of the time horizon on generalization, the test set was segmented into two sets, with 510 and 515 patterns each. Performing 30 experiments, the six out of the 30 best individuals yielded a positive rate of return on the test set that contained the patterns immediately following the training period. The performance of the rule that yielded the best performance on this dataset is visualized in Fig. 2. Taking into account the fact that daily exchange rates are



(a) Performance on Training Set



(b) Performance on Test Set

Figure 2: Trading performance of the best rule identified with transactions cost 0.05%.

highly noisy, and noise causes overfitting, the fact that GP was able to identify a rule yielding a return close to 19% over the test set is promising. At present we are considering the construction of a portfolio of genetically programmed trading rules to mitigate the problem of overfitting [5]. We also intend to verify the statistical significance of our results using the bootstrap method, as well as, to examine if the trading frequency of our approach is in line with real world practice.

References

[1] Y.-W. Cheung and M. D. Chinn, Currency traders and exchange rate dynamics: a survey of the US market, *Journal of International Money and Finance* 20(4) 439–471(2001).

[2] A. Koenig and B. Moo: *Ruminations on C++: A Decade of Programming Insight and Experience*. Addison-Wesley, 1996.

[3] J. R. Koza: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[4] S. Luke, Two fast tree-creation algorithms for genetic programming, *IEEE Transactions on Evolutionary Computation* **4**(3) 274–283(2000).

[5] C. J. Neely, P. A. Weller, and R. Dittmar, Is technical analysis in the foreign exchange market profitable? A genetic programming approach, *Journal of Financial and Quantitative Analysis* **32**(4) 405–426(1997).

[6] D. Olson, Have trading rule profits in the currency market declined over time? *Journal of Banking and Finance* **28**(4) 85–105(2004).

[7] R. Poli and W. B. Langdon, On the search properties of different crossover operators in genetic programming (Editors: J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, R. Riolo), *Genetic Programming 1998: Proceedings of the Third Annual Conference* 293–301(1998).

[8] R. Sullivan, A. Timmermann, and H. White, Data-snooping, technical trading rule performance, and the bootstrap. *Journal of Finance* **54**(5) 1647–1691(1999).