CMS 1: 311–327 (2004)

DOI: 10.1007/s10287-004-0018-5

Evolving technical trading rules for spot foreign-exchange markets using grammatical evolution

Anthony Brabazon[1], **Michael O'Neill**[2]

[1] University College Dublin, Ireland (e-mail: anthony.brabazon@ucd.ie)
[2] University of Limerick, Ireland (e-mail: michael.oneill@ul.ie)

Abstract. Grammatical Evolution (GE) is a novel, data-driven, model-induction tool, inspired by the biological gene-to-protein mapping process. This study provides an introduction to GE, and applies the methodology in an attempt to uncover useful technical trading rules which can be used to trade foreign exchange markets. In this study, each of the evolved rules (programs) represents a market trading system. The form of these programs is not specified *ex-ante*, but emerges by means of an evolutionary process. Daily US-DM, US-Stg and US-Yen exchange rates for the period 1992 to 1997 are used to train and test the model. The findings suggest that the developed rules earn positive returns in hold-out sample test periods, after allowing for trading and slippage costs. This suggests potential for future research to determine whether further refinement of the methodology adopted in this study could improve the returns earned by the developed rules. It is also noted that this novel methodology has general utility for rule-induction, and data mining applications.

Keywords: Grammatical evolution, Foreign exchange prediction, Technical trading rules

MSC codes: 91B84, 68W10, 62J02

1 Introduction

The last decade has seen significant advances in the field of computational intelligence, leading to the development of powerful new modelling technologies. Generally, these technologies fall into three categories, those which are inspired by the workings of biological neurons (Neural Networks), those which are inspired by an evolutionary metaphor (Genetic Algorithms, Genetic Programming and Grammatical Evolution), and those which are inspired by studies of animal interactions (Particle Swarm and Ant Colony models). While neural networks, and to a lesser extent genetic algorithms and ant-algorithms, have featured in Management Science applications, other forms of computational intelligence have received relatively less attention.

Grammatical Evolution (GE), represents an evolutionary automatic programming methodology, and can be used to evolve 'rule sets'. These rule sets can be as general as a functional expression which produces a good mapping between a series of known input-output data vectors. A particular strength of the methodology is that the form of the model need not be specified *a priori* by the modeller. This is of particular utility in cases where the modeller has a theoretical or intuitive idea of the nature of the explanatory variables, but a weak understanding of the functional relationship between the explanatory and the dependent variable(s). GE does not require that the model form is linear, nor does the method require that the measure of model error used in model construction is a continuous or differentiable function. A key element of the methodology is the concept of a *Grammar*, which governs the creation of the rule sets. The concept of a Grammar is inspired by the biological process of mapping of genes to proteins. This paper introduces the GE methodology to Management Science, and applies the methodology in an attempt to uncover a series of useful technical trading rules for trading in spot foreign exchange markets (transactions in which two currencies are immediately exchanged).

1.1 Foreign exchange markets

Foreign exchange markets are the most active of all financial markets with average daily trading volumes in traditional (non-electronic broker) foreign exchange markets estimated at $1.2 trillion (Bank of International Settlements, 2001). The average daily trading volume on spot markets exceeds $387 billion. Although the precise scale of speculative trading on spot markets is unknown, the bulk of transactions are not represented by world trade in goods and services (Froot and Thaler, 1990). Only about 15% of the trading is driven by non-dealer/financial institution trading (Bank of International Settlements, 2001). Therefore, it is plausible to assume that the scale of speculative trading is substantial. Speculative foreign currency trading implies the existence of predictive models in the mind of investors. These models could integrate many different forms of information. This study focuses on a subset of this information, that contained in the historical time-series of exchange rates. International foreign-exchange markets are dominated by a small number of heavily-traded currency pairings. Approximately 90% of all foreign currency transactions involve the US Dollar (Bank of International Settlements, 2001). Prior to the recent introduction of the Euro, the most-traded currency pairings were the US Dollar-DM, US Dollar-Yen and US Dollar-Sterling, accounting for 20%, 18%

and 8% of total daily trading volume in 1998 (Bank of International Settlements, 2001). This study develops trading systems for each of these currency pairs.

1.2 Technical analysis

Technical analysis is defined as the attempt to identify regularities in the time-series of price and volume information from a financial market, by extracting patterns from noisy data (Lo, Mamaysky and Wang, 2000). Some market traders known as technical analysts, believe that prices (exchange rates) move in trends and that price patterns repeat themselves (Murphy, 1999). Although controversy exists amongst financial theorists regarding the veracity of the claims of technical analysts, the methods are widely applied in practice. In a study (Taylor and Allen, 1992) conducted on behalf of the Bank of England, it was found that approximately 90% of financial institutions dealing in foreign exchange in London placed some weight on information obtained from technical analysis. Greatest use was made of this information in forming short-run (a trading horizon of less than three months) exchange rate expectations. In the case of foreign-exchange markets, studies which suggest that technical trading strategies may be profitable include Sweeney (1986) and Levich and Thomas (1993). Further support for a technical analysis argument is found in Osler (2003), where evidence of clustering of currency stop-loss and take-profit orders is noted. The affect of clustering of these orders is that trends in exchange rates will tend to reverse at predictable support and resistance levels, but trends tend to accelerate once these levels are breached. Clusters of take-profit orders were found at exchange rates characterized by ‘round numbers’ (ending in 00) with stop-loss orders clustering just beyond round numbers. If we accept the premise that there are rules, although not necessarily static rules, which partly explain price behavior it follows that opportunities may exist to enhance trading decisions through use of an appropriate rule induction methodology such as GE.

1.3 Technical Indicators

Technical indicators are formed from various combinations of current and historic price (volume) information. Four groupings of indicators are given prominence in the technical analysis literature (Brock, Lakonishok and LeBaron, 1992; Murphy, 1999; Pring, 1991):

i. Moving average indicators
ii. Momentum indicators
iii. Trading range indicators
iv. Oscillators

In a study of the application of technical analysis in the foreign exchange markets, Taylor and Allen (1992) found that the most common indicators utilized were

moving averages, followed by momentum and oscillators. These findings are supported by the recent study of Eun and Sabherwal (2002) which provides empirical evidence, based on currency rate forecasts of major banks, that banks tend to be momentum forecasters. Given the large search space, an evolutionary automatic programming methodology has promise to determine both a good quality combination of, and relevant parameters for, trading rules drawn from individual technical indicators. In this paper, we have limited our attention to the first three groupings of indicators.
The simplest moving average systems compare the current exchange rate with a moving average of the exchange rate over a lagged period, to determine how far the current rate has moved from an underlying trend. As they smooth out daily price fluctuations, moving averages can heighten the visibility of an underlying trend. A trading signal can be generated when an exchange rate moves above or below its moving average. A variation on simple moving average systems is to use a moving average convergence-divergence (MACD) oscillator. This is calculated by taking the difference of a short run and a long run moving average. In a recursive fashion, more complex combinations of moving averages of values calculated from a MACD oscillator can themselves be used to generate trading rules. The momentum of an exchange rate is the ratio of a time-lagged exchange rate to the current rate ($Rate_t / Rate_{t-x}$). The belief underlying this indicator is that a strong tend is likely to persist for a period of time. Trading Range Breakout indicators seek to determine when the price of a currency moves out of a range defined by its maximum or minimum value in a lagged time-period. A simple example of a trading rule derived from a breakout indicator would be to buy a currency when it exceeds its previous high (against another currency) in the last four weeks and conversely to sell if it falls below its previous four week low.

1.4 Motivation for study

Although many studies have examined the potential for technical trading since the move to a floating exchange rate system in the 1970s, typically studies employ a limited, fixed, sub-set of the population of technical indicators. In most studies, the lag periods of the indicators are fixed and the modelling task is the selection of a subset of the pre-specified indicators along with their coefficients in order to minimize model-fit error. As noted by Iba and Nikolaev (2000) there are a number of reasons to suppose that the use of an evolutionary automatic programming (EAP) approach, such as GE, can prove fruitful in the financial prediction domain. EAP can conduct an efficient exploration of a search space and can uncover dependencies between input variables, leading to the selection of a good subset for inclusion in the final model. Additionally, use of EAP facilitates the implementation of complex fitness functions including discontinuous, non-differentiable functions. This is of particular importance in a financial domain as fitness criterion may be complex, usually requiring a balancing of risk and return.

Although quite a number of studies have applied artificial neural networks to the problem of foreign exchange rate prediction (Yao and Tan, 2000; Hu, Zhang, Jiang and Patuwo, 1999; Jamal and Sundar, 1999; Gencay, Dacorogna, Olsen and Pictet, 2003; Franses and Van Homelen, 1998), there have been fewer applications of Genetic Algorithm (GA) (Holland, 1975; Goldberg, 1989) or Genetic Programming (GP) (Koza, 1992; Banzhaf et al., 1998) methodologies to the prediction of financial markets using technical indicators. Notable exceptions include (Allen and Karjalainen, 1999; Colin, 1994; Bauer, 1994; Neely, Weller and Dittmar, 1997; Deboeck, 1994; O'Neill, Brabazon and Ryan, 2002) which concentrated on equity markets. However, little published work other than Bhattacharyya, Pictet, Zumbach (2002) exists concerning the application of evolutionary-inspired algorithms to the technical trading of foreign exchange markets. The need to research the potential of EAP methodologies in the foreign exchange markets has been recognized in the recent past (Jegadeesh, 2000). This study answers this call, and introduces GE to the problem domain.

The rest of this paper is organized as follows. The next section provides an introduction to Grammatical Evolution. Then we outline the data set and methodology utilized. The following sections provide the results of the study followed by a number of conclusions.

2 Grammatical evolution

A full description of GE can be found in (O'Neill and Ryan, 2003; O'Neill and Ryan, 2001; O'Neill, 2001; Ryan, Collins and O'Neill, 1998). GE is an evolutionary algorithm that can evolve computer programs in any language and as such is an example of an evolutionary automatic programming system. We use the term evolutionary automatic programming to refer to those systems that adopt evolutionary computation to automatically generate computer programs and as such it encompasses Genetic Programming (GP) (Koza, 1992; Koza, 1994; Banzhaf et al., 1998; Koza et al., 1999; Koza et al., 2003) and all its variants. For a full overview of the background on GE and its context in evolutionary automatic programming, the reader is referred to (O'Neill and Ryan, 2003). Rather than representing the programs as syntax trees, as in GP, a linear genome representation is used. Each individual, a variable length binary string, contains in its codons (groups of 8 bits) the information to select production rules from a Backus Naur Form (BNF) grammar. In other words, an individuals binary string contains the instructions that direct a developmental process resulting in the creation of a program or expression. As such, GE adopts a biologically-inspired, genotype-phenotype mapping process. Future advances in the field of evolutionary algorithms can be easily incorporated into this system due to the separation of phenotype from genotype, as the mapping process at the heart of GE can be plugged onto any search algorithm that can operate over binary or integer vectors.

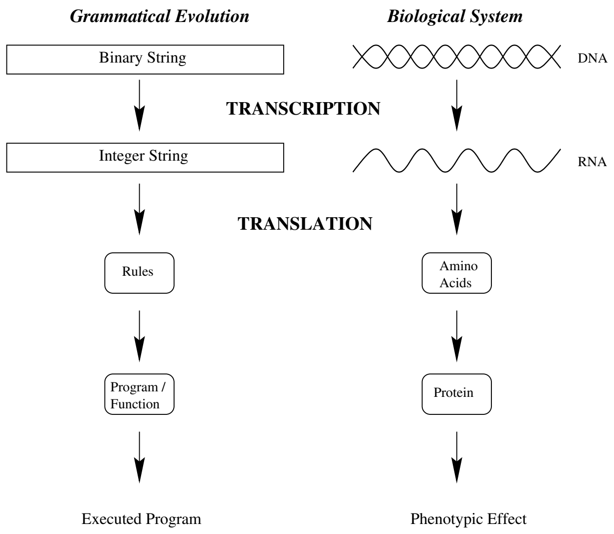

Fig. 1. A comparison between the grammatical evolution system and a biological genetic system. The binary string of GE is analogous to the double helix of DNA, each guiding the formation of the phenotype. In the case of GE, this occurs via the application of production rules to generate the terminals of the compilable program. In the biological case by directing the formation of the phenotypic protein by determining th e order and type of protein subcomponents (amino acids) that are joined together

2.1 The biological approach

The GE system is inspired by the biological process of generating a protein from the genetic material of an organism. Proteins are fundamental in the proper development and operation of living organisms and are responsible for traits such as eye colour and height (Lewin, 2000).

The genetic material (usually DNA) contains the information required to produce specific proteins at different points along the molecule. For simplicity, consider DNA to be a string of building blocks called nucleotides, of which there are four, named A, T, G, and C, for adenine, tyrosine, guanine, and cytosine respectively. Groups of three nucleotides, called codons, are used to specify the building blocks of proteins. These protein building blocks are known as amino acids, and the sequence of these amino acids in a protein is determined by the sequence of codons on the DNA strand. The sequence of amino acids is very important as it plays a large part in determining the final three-dimensional structure of the protein, which in turn has a role to play in determining its functional properties.

In order to generate a protein from the sequence of nucleotides in the DNA, the nucleotide sequence is first transcribed into a slightly different format, that being a sequence of elements on a molecule known as mRNA. Codons within the mRNA molecule are then translated to determine the sequence of amino acids that are contained within the protein molecule. The application of production rules to the non-terminals of the incomplete code being mapped in GE is analogous to the role amino acids play when being combined together to transform the growing protein molecule into its final functional three-dimensional form.

The result of the expression of the genetic material as proteins in conjunction with environmental factors is the phenotype. In GE, the phenotype is a computer program that is generated from the genetic material (the genotype) by a process termed a genotype-phenotype mapping. This is unlike the standard method of generating a solution (a program in the case of GE) directly from an individual in an evolutionary algorithm by explicitly encoding the solution within the genetic material. Instead, a many-to-one mapping process is employed within which the robustness of the GE system lies. Figure 1 compares the mapping process employed in both GE and biological organisms.

2.2 The mapping process

When tackling a problem with GE, a suitable BNF (Backus Naur Form) grammar definition must first be decided upon. The BNF can be either the specification of an entire language or, perhaps more usefully, a subset of a language geared towards the problem at hand.

In GE, a BNF definition is used to describe the output language to be produced by the system. BNF is a notation for expressing the grammar of a language in the form of production rules. BNF grammars consist of `terminals`, which are items that can appear in the language, e.g. binary operators **+, -**, unary operators **Sin**, constants **1.0** etc. and `non-terminals`, which can be expanded into one or more terminals and non-terminals. For example from the grammar detailed below, `<expr>` can be transformed into one of four rules, i.e it becomes `<expr><op><expr>`, `(<expr><op><expr>)` (which is the same as the first, but surrounded by brackets), `<pre-op>(<expr>)`, or `<var>`. A grammar can be represented by the tuple $\{N, T, P, S\}$, where N is the set of non-terminals, T the set of terminals, P a set of production rules that maps the elements of N to T, and S is a start symbol which is a member of N. When there are a number of production rules that can be applied to one element of N the choice is delimited with the '|' symbol. For example,

Table 1. The number of choices available from each production rule

Rule no.	Choices
A	4
B	4
C	1
D	2

```
N = { <expr>, <op>, <pre_op> }
T = {Sin, +, -, /, *, X, 1.0, (, )}
S = <expr>
```

And *P* can be represented as:

```
(A) <expr> ::= <expr> <op> <expr>      (0)
             | ( <expr> <op> <expr> )  (1)
             | <pre-op> ( <expr> )     (2)
             | <var>                   (3)

(B) <op> ::= +        (0)
           | -        (1)
           | /        (2)
           | *        (3)

(C) <pre-op> ::= Sin

(D) <var> ::= X       (0)
            | 1.0     (1)
```

The compilable code produced will consist of elements of the terminal set T. The grammar is used in a developmental approach whereby the evolutionary process evolves the production rules to be applied at each stage of a mapping process, starting from the start symbol, until a complete program is formed. A complete program is one that is comprised solely from elements of T.
As the BNF definition is a plug-in component of the system, it means that GE can produce code in any language thereby giving the system a unique flexibility. For the above BNF, Table 1 summarizes the production rules and the number of choices associated with each.
The genotype is used to map the start symbol onto terminals by reading codons of 8 bits to generate a corresponding integer value, from which an appropriate production rule is selected by using the following mapping function:

$$Rule = Codon\ Value\ \%\ No.\ Rule\ Choices$$

Consider the following rule from the given grammar. Given the non-terminal *op*, there are four production rules which can be selected which will replace *op* with a terminal.

```
(B) <op> :: =    +           (0)
               | -           (1)
               | /           (2)
               | *           (3)
```

If we assume the codon being read produces the integer 6, then

$$6\ \%\ 4\ =\ 2$$

would select rule (2) /. Each time a production rule has to be selected to transform a non-terminal, another codon is read. In this way the system traverses the genome.

During the genotype-to-phenotype mapping process, it is possible for individuals to run out of codons, and in this case we wrap the individual and reuse the codons. This is quite an unusual approach in EAs, as it is entirely possible for certain codons to be used two or more times. This technique of wrapping the individual draws inspiration from the gene-overlapping phenomenon that has been observed in many organisms (Lewin, 2000).

In GE, each time the same codon is expressed it will always generate the same integer value, but, depending on the current non-terminal to which it is being applied, it may result in the selection of a different production rule. We refer to this feature as *intrinsic polymorphism*. What is crucial, however, is that each time a particular individual is mapped from its genotype to its phenotype, the same output is generated. This is the case because the same choices are made each time. However, it is possible that an incomplete mapping could occur, even after several wrapping events, and in this case the individual in question is given the lowest possible fitness value. The selection and replacement mechanisms then operate accordingly to increase the likelihood that this individual is removed from the population.

An incomplete mapping could arise if the integer values expressed by the genotype were applying the same production rules repeatedly. For example, consider an individual with three codons, all of which specify rule 0 from below:

```
(A) <expr> :: =  <expr><op><expr>        (0)
                |(<expr><op><expr>)      (1)
                |<pre-op>(<expr>)        (2)
                |<var>                   (3)
```

even after wrapping the mapping process would be incomplete and would carry on indefinitely unless stopped. This occurs because the nonterminal `<expr>` is being mapped recursively by production rule 0, becoming `<expr><op><expr>`. Therefore, the leftmost `<expr>` after each application of a production would itself be mapped to a `<expr><op><expr>`, resulting in a continually growing expression. Such an individual is dubbed invalid as it will never undergo a complete mapping to a set of terminals. For this reason we impose an upper limit on the number of wrapping events that can occur. It is clearly essential that stop sequences are found during the evolutionary search in order to complete the mapping process to a functional program. The stop sequence being a set of codons that result in the non-terminals being transformed into elements of the grammars terminal set.

Beginning from the left hand side of the genome then, codon integer values are generated and used to select rules from the BNF grammar, until one of the following situations arise:

i. A complete program is generated. This occurs when all the non-terminals in the expression being mapped are transformed into elements from the terminal set of the BNF grammar.
ii. The end of the genome is reached, in which case the *wrapping* operator is invoked. This results in the return of the genome reading frame to the left hand side of the genome once again. The reading of codons will then continue, unless an upper threshold representing the maximum number of wrapping events has occurred during this individual's mapping process.

iii. In the event that a threshold on the number of wrapping events has occurred and the individual is still incompletely mapped, the mapping process is halted, and the individual is assigned the lowest possible fitness value.

To reduce the number of invalid individuals being passed from generation to generation, a steady state replacement mechanism is employed. One consequence of the use of a steady state method is its tendency to maintain fit individuals at the expense of less fit, and in particular, invalid individuals.

In this study, the GE algorithm uses a steady state replacement mechanism, such that, two parents produce two children the best of which replaces the worst individual in the current population, if the child has greater fitness. The standard genetic operators of bit mutation (probability of 0.01), and crossover (probability of 0.9) are adopted as is a duplication operator (probability of 0.01), that duplicates a random codon and inserts this into the penultimate codon position on the genome. A series of functions, in this study, technical indicators, are pre-defined as are a series of mathematical operators. A population of initial trading rulesets (programs) are randomly generated, and by means of an evolutionary process, these are improved. No explicit model specification is assumed *ex-ante*, although the choice of mathematical operators defined in the grammar do place implicit limitations on the model specifications amongst which GE can search.

3 Experimental approach

This study uses daily closing exchange rate data drawn from the London market for the period 23/10/92 to 13/10/97. The training data set was comprised of the first 799 trading days of the data set. In the application of any iterative modelling technique it is important to consider the danger of data-snooping. Data-snooping can occur when a dataset is used more than once in a model construction process (Sullivan, Timmermann and White, 1999). To reduce this danger, the developed models are tested using an extensive out-of-sample dataset of 548 trading days. The out-of-sample data is divided into two hold-out samples (274 * 2) to allow comparison of the hold-out results across different market conditions. This is done in order to assess the stability and degradation characteristics of the developed models' predictions. The rules evolved by GE are used to generate one of three signals for each day of the training or test periods. The possible signals are *Buy*, *Sell*, or *Do-Nothing*. Permitting the model to output a Do-Nothing signal reduces the hard threshold problem associated with production of a binary output. Markets do not invariably trend upwards or downwards, but may trade sideways in a relatively narrow band of prices (a trading range). Trend-following trading systems will tend to struggle in sideways-moving markets, and hence should be allowed to 'opt-out' of trading these markets for which they are not well-suited.

3.1 Trading methodology

A variant on the trading methodology developed in Brock, Lakonishok and LeBaron (1992) is then applied. If a buy signal is indicated on a given day, a fixed investment of \$1,000 is made in the foreign currency using borrowed funds. This position is automatically closed at the end of a five day trading period. On the production of a sell signal, an investment of \$1,000 is made on the short side and again this position is closed out after a five day period. This gives rise to a maximum potential investment of \$5,000 at any point in time.[1] The total return (measured as a percentage) generated by the developed trading system is a combination of its trading return net of trading costs and an interest differential.

The technical indicators adopted in this study are the moving average, momentum and trading range breakout where the relevant periods and combination of indicators used are to be determined by evolution. Apart from the raw components of the technical indicators adopted (average, moving average, lag, each of which takes a real-valued argument representing the chosen time-window) the BNF grammar also allows the use of standard arithmetic operators (+, -, * and inequality operators) and the binary operators f_and, f_or, and the unary operator f_not.[2] The current day's price is also provided as a model input. The signals generated for each day are post-processed using the following rule:

$$Sell = Value < .33$$

$$Do - Nothing = .33 >= Value < .66$$

$$Buy = .66 >= Value$$

Consider a daily observed exchange rate series S_t with $t = 1, 2, ...n$. We define the five day (%) trading return (before trading costs) for a long position as $(\frac{S_t}{S_{t+5}} - 1)$ and as $-1[1 - (\frac{S_{t+5}}{S_t})]$ for a short position.

In this trading strategy, the home currency is the US dollar. When a long position is taken, the foreign currency is purchased and the return equals the trading return plus the interest earned in the foreign currency, less the cost of the home funds borrowed to invest in the foreign currency. When a short signal is produced by the trading system, the foreign currency is borrowed, at a cost of the foreign interest rate, and is invested in US dollars for the trading period. The dollar interest rate is earned on these funds. A trading cost c of 0.025% and a slippage allowance of 0.01% in each (single) direction is included (Levich and Thomas, 1993).

To encourage the evolution of a trading system with good risk to return characteristics, we penalize trading systems leading to volatile patterns of returns. The

[1] The potential loss on an unprotected short sale is in theory infinite but in practice is unlikely to exceed \$1,000. On a short sale, an amount of \$1,000 is considered invested for the five day period of the transaction.

[2] The operations f_and(argument 1, argument 2), f_or (argument 1, argument 2), and f_not (argument 1) return the minimum, maximum, of the arguments, and 1 - the argument, respectively.

risk of catastrophic loss is reduced by incorporating a measure of the system's drawdown (maximum cumulative loss since commencing trading with the system) into the fitness function:

$$Fitness = Return - x * (Max.drawdown)$$

'x' can be considered as a tuning parameter for the penalty function. The more risk-adverse the investor, in terms of wishing to avoid a catastrophic loss, the higher the value of this parameter. We initially set this parameter to 1.

3.2 Benchmark for trading system

Before the returns generated by the trading system can be assessed, an appropriate benchmark must be defined. A wide variety of benchmarks could be utilized including:

- i A no-change model
- ii A perfect foresight model
- iii Simple buy-and-hold

The no-change model assumes that foreign-exchange rates follow a random-walk, therefore the current (spot) foreign exchange rate is the best predictor of the future rate. Under this perspective, the expected rate of return from speculating in foreign currencies, net of financing costs, is zero. The perfect foresight model compares the generated trading returns with the returns the investor could have generated if he had perfect foresight of all future changes in the exchange rate. This is a very stringent benchmark as we know that financial time-series have a low signal to noise ratio. This benchmark is not used in this study. A simple buy-and-hold strategy results when an investor buys the foreign currency long or short, and closes out the position at the end of the trading period. This provides a basic test of whether the trading system is covering its associated trading costs. Although this is a simple metric, it is not risk-comparable with the trading model, as it maintains a fully-invested position in the currency markets at all times. To the extent that the buy-and-hold strategy maintains a greater investment in currency markets than the trading model, it has more capital at risk.

4 Results

The results from our experiments are now provided. 30 independent trials were conducted with a population size of 500 individuals and an evolutionary period of 50 generations. To assess the quality of the results obtained they are compared against a 'buy-and-hold' benchmark. The 'buy-and-hold' benchmark is determined by comparing the return, measured as the percentage gain on the initial investment, obtained from investing \$5,000 in the foreign currency, net of the interest rate differential, during each training and out-of-sample period.

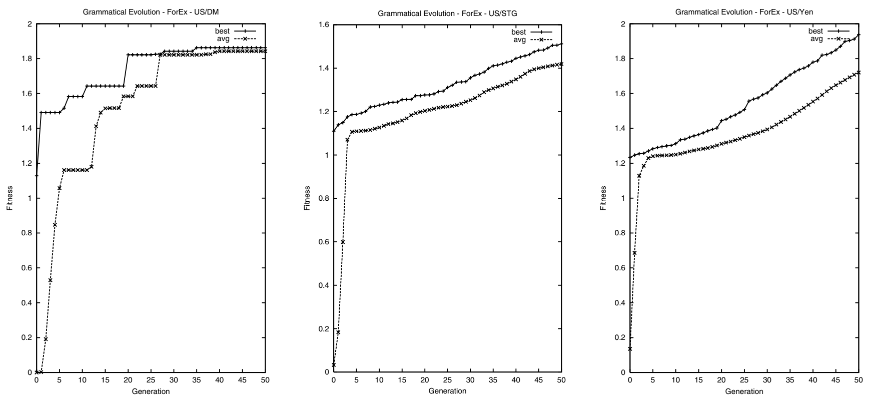

Fig. 2. In-sample fitness of overall best individual and mean of best individuals over the 30 runs, plotted for each currency pair

Table 2. Results for the best (mean) evolved rule set over the 30 runs compared to the benchmark buy-and-hold strategy on the US-DM dataset

Trading period	*Evolved Rule set*	*Buy-and-Hold*
Training	2.02357 (1.865)	-0.112199
Test 1	0.017998 (0.017998)	0.068231
Test 2	0.302671 (0.302671)	0.153231

4.1 US-DM

A plot of the mean best and mean average fitness at each generation over the 30 runs can be seen in Figure 2. Table 2 provides a comparison of the performance (percentage return on the initial capital of $5,000, where 0.01 = 1%) of the best evolved ruleset against that of the benchmark investment strategy.

4.2 US-STG

Table 3 provides a comparison of the performance of the best evolved ruleset against that of the benchmark investment strategy.

4.3 US-Yen

Table 4 provides a comparison of the performance of the best evolved ruleset against that of the benchmark investment strategy.

Table 3. Results for the best (mean) evolved rule set over the 30 runs compared to the benchmark buy-and-hold strategy on the US-STG dataset

Trading period	*Evolved Rule set*	*Buy-and-Hold*
Training	1.75642 (1.5123)	−0.055855
Test 1	0.343709 (0.26103)	−0.094723
Test 2	0.500649 (0.47824)	−0.041456

Table 4. Results for the best (mean) evolved rule set over the 30 runs compared to the benchmark buy-and-hold strategy on the US-Yen dataset

Trading period	*Evolved Rule set*	*Buy-and-Hold*
Training	2.3803 (1.9374)	-0.080146
Test 1	0.342609 (0.066474)	0.133476
Test 2	0.195778 (0.32521)	0.090996

5 Discussion

A total of 799 days of data were used to train the models, which were then tested on a further 548 days of out-of-sample data. The out-of-sample data is split into two sub-samples of 274 days, in order to assess whether the results of the trading system degrade noticeably during the out-of-sample period. Despite the lengthy out-of-sample period, the evolved trading rules generated positive returns on all the hold-out samples, after allowance for trading, slippage and net interest costs. In five of the six hold-out-periods, the best evolved rule notably outperforms the benchmark buy-and-hold strategy. The mean (over all 30 runs) best evolved rule outperforms the benchmark in four of the six hold-out periods. It is also notable that the out-of-sample results appear robust over the two out-of-sample periods ('Test 1' and 'Test 2'), and do not indicate that the performance of the trading system is declining.

In evaluating the performance of any market timing system, a number of caveats must be borne in mind. Any trading model constructed and tested using historic data will tend to perform less well in a live environment than in a back-test period for a number of reasons. Live markets have attendant problems of delay in executing trades, illiquidity, interrupted / corrupted data and interrupted markets. The impact of these issues is to raise trading costs and consequently to reduce trading profitability. An allowance for these costs (slippage) has been included in this study but it is impossible to determine the scale of these costs ex-ante with complete accuracy. In addition, markets are competitive. As new computational technologies spread, opportunities to utilize these technologies to earn excess risk-adjusted profits are eroded (Gencay, Dacorogna, Olsen and Pictet, 2003). Hence, estimates of trading performance based on historical data may not be replicated in live trading as other market participants will apply similar technologies. It must also be remembered that foreign exchange trading takes place 24 hours per day, hence there is no true opening or closing price for a foreign exchange market. In view of the difficulties in constructing and testing models using continuous data over a prolonged period, this study like most others, adopts a sampling approach by employing daily closing exchange rate data, drawn from the London market.

6 Conclusions & Future work

In this paper a novel methodology GE was introduced, and applied for the purposes of foreign-exchange prediction. It is noted that this novel methodology has general utility for rule-induction, data mining applications. GE is an evolutionary algorithm that can evolve 'rulesets'. In this study each of the evolved rulesets represents a market trading system. GE was shown to evolve trading rules that were superior to the benchmark buy-and-hold strategy in five of the six hold-out periods. A number of additional extensions of this work are left for future development. One extension would be to incorporate a more complex model of learning (forgetting). Glassman (1973) suggested that the "fallibility of memory" (p. 88) may represent a useful adaptive device when faced with a dynamic environment. At present in our model, all historic data observations are given equal weighting which implicitly assumes model stationarity. By a suitable modification of the fitness function, whereby more recent data observations are assigned a higher weighting in the model construction process, model development could be biased towards more recent data (Refenes et al., 1997). The weighting parameter could also be evolved as a component of the developed model. Our methodology for this initial study has included a number of simplifications, for example we only considered a small set of technical indicators, one metric for trading risk, and a relatively low frequency sampling of foreign-exchange data. The study could also be extended by constructing a longer horizon predictive model using GE, employing a series of fundamental as distinct from technical indicators. Scope also exists to develop more sophisticated money-management strategies than those employed in this study. For example, rather than inventing for a set period of five trading days, each open position could be re-assessed every day, with stop-losses being used to reduce the risk of major losses. Another interesting strategy is to vary the amount invested on each trade. Rather than investing a fixed amount in the currency market on each trade, the investment amount could be linked to the strength of the trading signal, with strong signals resulting in the assumption of a bigger position than weaker signals.

References

[1] Allen F, Karjalainen R (1999) Using genetic algorithms to find technical trading rules. *Journal of Financial Economics* **51**(2): 245–271

[2] Bank for International Settlements (2001) Central bank survey of foreign exchange and derivatives market activity in April 2001. *Press Release, 31/2001E, Bank of International Settlements*, October 2001

[3] Banzhaf W, Nordin P, Keller RE, Francone FD (1998) *Genetic Programming – An Introduction: On the Automatic Evolution of Computer Programs and its Applications*. San Francisco: Morgan Kaufmann
[4] Bauer R (1994) *Genetic Algorithms and Investment Strategies*. New York: Wiley
[5] Bhattacharyya S, Pictet O, Zumbach G (2002) Knowledge-intensive genetic discovery in foreign exchange markets. *IEEE Transactions on Evolutionary Computation* **6**(2): 169–181
[6] Brown S, Goetzmann W, Kumar A (1998) The Dow Theory: William Peter Hamilton's Track Record Reconsidered. *Journal of Finance* **53**(4): 1311–1333
[7] Colin A (1994) *Genetic Algorithms for Financial Modelling*. In: Deboeck G (ed) Trading on the edge: neural, genetic and fuzzy systems for chaotic and financial markets. New York: Wiley
[8] Deboeck G (1994) *Using GAs to optimise a trading system*. In: Deboeck G (ed) Trading on the edge: neural, genetic and fuzzy systems for chaotic and financial markets. New York: Wiley
[9] Eun C, Sabherwal S (2002) Forecasting exchange rates: Do banks know better? *Global Finance Journal* **13**(2): 195–215
[10] Franses P, Van Homelen P (1998) On forecasting exchange rates using neural networks. *Applied Financial Economics* **8**(6): 589–596
[11] Froot K, Thaler R (1990) Anomalies: Foreign Exchange. *Journal of Economic Perspectives* **4**(3): 179–192
[12] Gencay R, Dacorogna M, Olsen R, Pictet O (2003) Foreign exchange trading models and market behavior. *Journal of Economic Dynamics & Control* (in press)
[13] Glassman R (1973) Persistence and Loose Coupling in Living Systems. *Behavioral Science* **18**: 83–98
[14] Goldberg DE (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston: Addison Wesley
[15] Holland JH (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press
[16] Hu M, Zhang G, Jiang C, Patuwo E (1999) A cross-validation analysis of neural network out-of-sample performance in exchange rate forecasting. *Decision Sciences* **30**(1): 197–216
[17] Iba H, Nikolaev N (2000) Genetic Programming Polynomial Models of Financial Data Series. In: *Proc. of CEC 2000*, pp 1459–1466. New York: IEEE Press
[18] Jamal A, Sundar C (1999) Modelling exchange rates with neural networks. *Journal of Applied Business Research* **14**(1): 1–5
[19] Jegadeesh N (2000) Discussion - Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. *Journal of Finance* **55**(4): 1765–1770
[20] Koza J (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Massachusetts: MIT Press
[21] Koza JR (1994) *Genetic Programming II: Automatic Discovery of Reusable Programs*. Massachusetts: MIT Press
[22] Koza JR, Andre D, Bennett III FH, Keane M (1999) *Genetic Programming 3: Darwinian Invention and Problem Solving*. San Francisco, Morgan Kaufmann
[23] Koza JR, Keane M, Streeter MJ, Mydlowec W, Yu J, Lanza G (2003) *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Boston: Kluwer Academic Publishers
[24] Lewin B (2000) *Genes VII*. Oxford: Oxford University Press
[25] Levich R, Thomas L (1993) The significance of technical trading-rule profits in the foreign exchange market: a bootstrap approach. *Journal of International Money and Finance* **12**(5): 451–474
[26] Lo AW, Mamaysky H, Wang J (2000) Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. *Journal of Finance* **55**(4): 1705–1765
[27] Murphy JJ (1999) *Technical Analysis of the Financial Markets*. New York: New York Institute of Finance
[28] Neely C, Weller P, Dittmar R (1997) Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *Journal of Financial and Quantitative Analysis* **32**(4): 405–428
[29] O'Neill M (2001) Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution. PhD thesis, University of Limerick, Ireland, 2001
[30] O'Neill M, Brabazon A, Ryan C (2002) Forecasting Market Indices Using Evolutionary Automatic Programming. In: Chen S (ed) *Genetic Algorithms and Genetic Programming in Computational Finance*. Boston: Kluwer Academic Publishers, pp 175–195

[31] O'Neill M, Ryan C (2001) Grammatical Evolution. *IEEE Trans. Evolutionary Computation* **5**(4): 349–358

[32] O'Neill M, Ryan C (2003) *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Boston: Kluwer Academic Publishers

[33] Osler C (2003) Currency orders and exchange rate dynamics: An explanation for the predictive success of technical analysis. *Journal of Finance* (forthcoming)

[34] Pring M (1991) *Technical analysis explained: the successful investor's guide to spotting investment trends and turning points*. New York: Mc Graw-Hill Inc

[35] Refenes AN, Bentz Y, Bunn DW, Burgess AN, Zapranis AD (1997) Financial time series modelling with discounted least squares backpropagation. *Neurocomputing* **14**(2): 123–138

[36] Ryan C, Collins JJ, 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*, pp 83–95. Berlin Heidelberg New York: Springer

[37] Sweeney R (1986) Beating the Foreign Exchange Market. *Journal of Finance* **41**(1): 163–182

[38] Sullivan R, Timmermann A, White H (1999) Data-Snooping, Technical Trading Rule Performance, and the Bootstrap. *Journal of Finance* **54**(5): 1647–1691

[39] Taylor M, Allen H (1992) The use of technical analysis in the foreign exchange market. *Journal of International Money and Finance* **11**(3): 304–314

[40] Yao J, Tan C (2000) A case study on using neural networks to perform technical forecasting of forex. *Neurocomputing* **34**: 79–98