Computational Intelligence Algorithms For Risk-Adjusted Trading Strategies

N.G. Pavlidis, E.G. Pavlidis, M.G. Epitropakis, V.P. Plagianakos, and M.N. Vrahatis

***Abstract*— This paper investigates the performance of trading strategies identified through Computational Intelligence techniques. We focus on trading rules derived by Genetic Programming, as well as, Generalized Moving Average rules optimized through Differential Evolution. The performance of these rules is investigated using recently proposed risk–adjusted evaluation measures and statistical testing is carried out through simulation. Overall, the moving average rules proved to be more robust, but Genetic Programming seems more promising in terms of generating higher profits and detecting novel patterns in the data.**

I. 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 (see [1] for a thorough description of a number of simple trading rules). Recent surveys [2] suggest that TA has been a major constituent of financial practice in foreign exchange markets. Moreover, a number of empirical as well as theoretical studies [3], [4] during the past three decades suggest that the application of TA in the foreign exchange market can yield substantial excess returns. These findings raise doubts on the validity of the efficient market hypothesis. Olson [5], however, argues that abnormal profit opportunities arise due to temporary inefficiencies which are in accordance with an evolving market. He further argues that the returns of simple trading rules over recent periods have declined, if not completely disappeared.

In this work, we employ Genetic Programming (GP) to identify novel trading strategies based only on the information contained in the history of past price movements. GP can be considered as a Computational Intelligence algorithm that mimics the behavior of an optimizing agent in the foreign exchange market. In this process it is critical to select a performance measure that accounts not only for the return obtained from a rule, but also penalizes rules for the risk they undertake. To this end, a recently proposed risk sensitive measure, X_{eff}, is used [6]. The performance of the GP identified rules is compared to that of Generalized Moving Average (GMA) rules [7]. The parameters of the GMA rules are optimized using the Differential Evolution (DE) algorithm [8] and the same objective function as that used in GP.

N.G. Pavlidis, M.G. Epitropakis, V.P. Plagianakos, and M.N. Vrahatis are with the Computational Intelligence Laboratory, Department of Mathematics, University of Patras, GR–26110 Patras, Greece (email: {npav,mikeagn,vpp,vrahatis}@math.upatras.gr)

E.G. Pavlidis is with the Department of Economics, Lancaster University Management School, Lancaster LA1 4YX, UK (email: e.pavlidis@lancaster.ac.uk).

Finally, a simulation methodology is implemented to test the statistical significance of the best performing strategies identified through each approach. Our findings suggest that the widely–used moving average rules exhibit a more robust behavior than that of the more complicated GP generated strategies. However, the hypothesis that the performance of these rules can be attributed to well–known statistical properties of the data cannot be rejected. On the other hand, GP is capable of identifying patterns that cannot be explained by traditional stochastic processes, so as to yield excess returns.

The remaining of this paper is organized as follows. Sections II and III briefly describe the Genetic Programming and the Differential Evolution algorithms. Section IV is devoted to the presentation of the generalized moving average rules, while Section V presents the risk sensitive performance measures. The methodology of the simulations and the experimental results (and their statistical analysis) are exhibited in Sections VI and VII. Finally, the paper ends with a discussion and concluding remarks.

II. Genetic Programming

In this Section we briefly outline the Genetic Programming (GP) algorithm which was applied to identify new trading rules. Conceptually, GP constitutes an extension of Genetic Algorithms (GAs) in which individuals are no longer fixed-length strings but rather computer programs expressed as *syntax trees* [9]. GP individuals consist of function and terminal nodes. Terminal nodes return as output the value of either a constant, or an input variable, or a zero-argument function. Thus, the arity of terminal nodes is zero. The set of possible terminal nodes is called the *terminal set*, $\mathcal{T}$. Function nodes on the other hand, process their inputs to compute an output. The *function set*, $\mathcal{F}$, is composed of the statements and functions available to GP.

The primary GP search operators are *crossover* and *mutation*. In crossover, one subtree from each of the two selected parents is exchanged between them to form two new individuals (offsprings). The motivation is that useful building blocks for the solution of a problem are accumulated in the population and crossover permits the aggregation of good building blocks into even better solutions of the problem [9]. Crossover is the predominant GP search operator [9], [10]. On the other hand, mutation operates on a single individual by altering a random subtree. Next, we briefly describe the GP initialization and the GP operators (selection, crossover, and mutation) used in this paper.

A. The GP Initialization

The individuals in the GP population are initialized by recursively generating syntax trees composed of randomly chosen function and terminal nodes. The two established GP initialization methods are the *grow* and the *full* method. Both methods require from the user to specify the maximum initial tree depth. According to the grow method, nodes are selected randomly from the function and the terminal sets. The grow method, therefore, produces trees of irregular shape, since once a terminal node is inserted the path ending with this node cannot be extended, even if the maximum initial depth has not been reached. On the other hand, in the full initialization method only function nodes are selected until the maximum initial depth is reached. Beyond that depth only terminal nodes are chosen to end the branches. This method results in a balanced tree, every branch of which reaches the maximum initial depth.

The *ramped half and half* initialization method [9] employs both grow and full to construct a GP population. Specifically, ramped half and half aims at initializing an equal number of GP individuals with maximum depth starting from the minimum depth of two up to the maximum initialization depth. For each depth level half the individuals are constructed using the grow, while the remaining individuals are constructed using the full initialization method. To obtain promising candidate solutions for the evolutionary process it has been proposed to initialize a much larger population (by a factor of ten) and to select the best performing individuals [9].

B. The GP Selection Algorithm

To derive the individuals that will comprise the population of the next generation, GP initially selects individuals from the current generation. The selection operators that have been proposed for genetic algorithms are also applicable to GP. In this study, we employed the most commonly encountered one, namely *proportionate selection*. We define as E_i the fitness of the i-th individual, where E is the function we wish to maximize. Then the probability of selecting the i-th individual as a parent of an individual of the next generation (offspring) is equal to $p_i = E_i / \sum_{j=1}^{N} E_j$.

C. The GP Crossover Operator

The primary GP search operator is crossover. Crossover operates on two parent individuals and yields two offsprings. Standard crossover randomly selects a node in each parent tree and then swaps the subtrees rooted at these nodes [9]. Koza suggests to use a 90% probability of selecting as crossover point a function node and to select a terminal node with probability 10%. If an offspring exceeds the maximum depth it is discarded and the corresponding parent individual takes its place in the population of the next generation.

Standard crossover, however, tends to produce offsprings that frequently inherit most of their code from one parent, and also favors local adjustments near the leaves of syntax trees [11]. To overcome these limitations, Poli and Langdon [11] proposed the *uniform crossover operator* for GP (GPUX) inspired from the homonymous operator in GAs. GPUX starts by identifying a tree that represents the *common region* between two syntax trees. Each node that lies in the common region is considered for crossover with a constant probability. For nodes that lie in the interior of the common region GPUX swaps the nodes without affecting the subtrees rooted at these nodes. On the contrary, for nodes on the boundary of the common region the subtrees rooted at these nodes are swapped.

Since the diversity of the GP individuals is high during the early stages of the algorithm, the common region between randomly selected individuals tends to be relatively small and hence GPUX favors the global exploration of the search space by swapping large subtrees near the root of the syntax trees. As the population converges the operator becomes more and more local, in the sense that the offsprings it produces are progressively more similar to their parents. An example of GPUX is provided in Fig. 1.

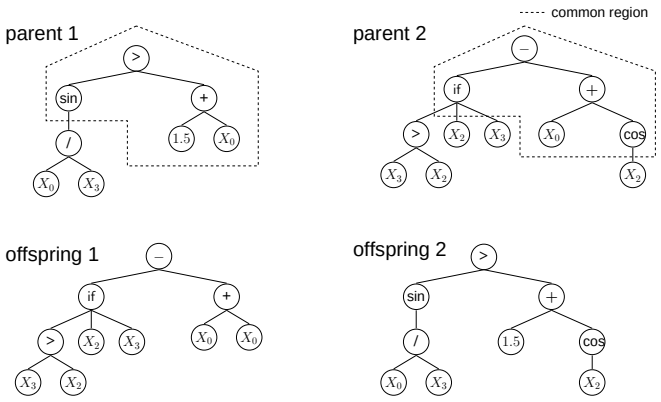

Fig. 1. Uniform crossover operator for GP (GPUX).

D. The GP Mutation Operator

The mutation operator in GP randomly selects a node of the syntax tree and replaces the subtree rooted at the selected node with a newly created tree. This type of mutation is known as *subtree mutation*. For the creation of the trees employed by the mutation operator, the grow initialization method is employed [9]. Fig. 2 illustrates the workings of mutation on a GP individual.

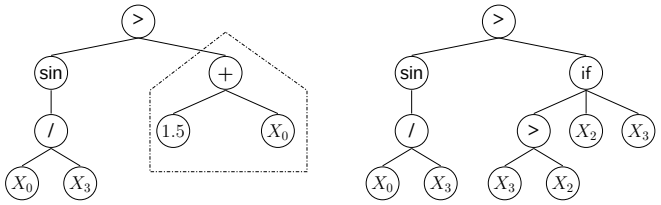

Fig. 2. Subtree mutation.

III. Differential Evolution

Differential Evolution [8] is a stochastic parallel direct search method, capable of handling non-differentiable,

nonlinear and multimodal objective functions. DE is a population–based stochastic algorithm that exploits a population of potential solutions (individuals), to explore the search space. The population of individuals is randomly initialized in the optimization domain with *NP*, n–dimensional, vectors following a uniform probability distribution. The population size, *NP*, is fixed throughout the execution of the algorithm.

At each iteration, called *generation*, new vectors are derived by the combination of randomly chosen vectors from the current population. This operation is referred to as *mutation* and produces the *mutant individuals*. Each mutant individual is then mixed with another, predetermined, vector – the *target* vector – through an operation called *recombination*. This operation yields the *trial* vector. Finally, the trial vector undergoes the *selection* operator, according to which it is accepted as a member of the population of the next generation only if it yields a reduction in the value of the objective function f relative to that of the target vector. Next we briefly review the basic DE variation operators.

A. DE Variation Operators

The DE search operators efficiently shuffle information among the individuals, enabling the search for an optimum to focus on the most promising regions of the solution space. The first operator considered here is mutation. Specifically, for each individual x_g^i, $i = 1, \ldots, NP$, where g denotes the current generation, a new individual v_g^i (mutant individual) is generated according to the following equation:

$$v_g^i = x_g^{\text{best}} + F(x_g^{r1} - x_g^{r2}), \tag{1}$$

where x_g^{best} is the best member of the previous generation; $F > 0$ is a real parameter, called *mutation constant*, which controls the amplification of the difference between two individuals, and is used to prevent the stagnation of the search process; and $r_1, r_2 \in \{1, 2, \ldots, i-1, i+1, \ldots, NP\}$, are random integers mutually different and not equal to the running index i. Although there exist many different mutation operators [12], [13], we chose to use the one described above since it is simple and has the property of fast convergence.

Having performed mutation, the recombination operator is applied to further increase the diversity of the population. The mutant individuals are combined with other predetermined individuals, called the target individuals. Specifically, for each component l ($l = 1, 2, \ldots, n$) of the mutant individual v_g^i, we randomly choose a real number r in the interval $[0, 1]$. Then, we compare this number with the *recombination constant*, CR. If $r \leqslant CR$, then we select, as the l-th component of the trial individual u_g^i, the l-th component of the mutant individual v_g^i. Otherwise, the l-th component of the target vector x_g^i becomes the l-th component of the trial vector. This operation yields the trial individual.

Finally, the trial individual is accepted for the next generation only if it reduces the value of the objective function (selection operator).

IV. Generalized Moving Average Rules

The simplest and most common trading rules employ moving averages (MAs). An MA of length θ is defined as:

$$MA(\theta)_t = \frac{1}{\theta} \sum_{i=0}^{\theta-1} P_{t-i}, \quad t = \theta, \theta+1, \ldots, N.$$

The Generalized MA (GMA) rule can be represented by the following binary indicator function [7]:

$$S(\Theta)_t = MA(\theta_1)_t - (1 + (1 - 2S_{t-1})\theta_3) MA(\theta_2), \tag{2}$$

where $\Theta = [\theta_1, \theta_2, \theta_3]$ are the parameters of the GMA. The GMA rule returns a buy signal (which is encoded as one), when Eq. (2) returns a positive number. A sell signal, on the other hand, corresponds to a non-positive return value and is encoded as zero. Typically, $\theta_1 < \theta_2$ and $MA(\theta_1)_t$ is called the short MA, while $MA(\theta_2)_t$ is the long MA. With this parameter setting the GMA rule identifies an upward trend when the short MA crosses from below the long MA, and vice versa. Finally, θ_3 is a parameter introduced to reduce the number of false buy and sell signals.

Fernández-Rodríguez *et al.* [7] employed Genetic Algorithms to determine the optimal parameter values (Θ) for GMA rules. They applied applied their approach to the Madrid stock market. Their findings indicate that with the exception of zero transactions costs, the best rules are of the form of double MA rules. In other words, $\theta_1 > 0$, $\theta_2 > 0$, and $\theta_3 = 0$. Moreover, the annualized returns, as well as, the Sharpe ratio corresponding to the best GMA rules are higher than those from the corresponding risk–adjusted buy and hold strategy. In this paper, we employ the Differential Evolution algorithm described above to compute and tune the parameters of GMA rules.

V. Risk Sensitive Performance Measures

A critical aspect for the identification of promising trading strategies through Computational Intelligence techniques is the performance evaluation measure. To evaluate the performance of an investment strategy it is necessary to measure not only the increase in capital but also the risk incurred. The first performance measure to incorporate risk is the widely–known *Sharpe ratio* [14]. The Sharpe ratio is defined as:

$$S_I = A_{\Delta t} \frac{\bar{r}}{\sqrt{\sigma_r^2}},$$

where $\bar{r}$ is the average return, σ_r^2 is the variance of the return series, and $A_{\Delta t}$ is an annualization factor that depends on the frequency at which returns are measured. Three drawbacks have been associated with the Sharpe ratio [6]. Firstly, the variance term is placed in the denominator, which makes the ratio numerically unstable when σ_r^2 is close to zero. Secondly, the returns are measured at one frequency, and hence the measure neglects the risk due to unrealized losses at other frequencies. Finally, the Sharpe ratio neglects the clustering of losses and profits.

To this end, Gençay *et al.* [6] have proposed two risk adjusted performance measures, X_{eff} and R_{eff}, that rely

on the expected utility framework. The first measure, X_{eff}, is derived from a constant risk aversion utility function of the form, $u(x) = -\exp(-\gamma x)$, with γ representing the coefficient of risk aversion, and x denoting the wealth reached by unit investment. Thus:

$$x(t) = \tilde{R}(t) - \tilde{R}(t - \Delta t),$$

where $\tilde{R}(t) = R(t) + r_o(t)$, $R(t)$ is the total return of past trades up to time t, and $r_o(t)$ is the unrealized return of the current trading model position. Assuming that x follows the normal distribution with $\mathcal{N}(\bar{x}, \sigma^2)$, the expected utility is:

$$E[u(x)] = -\exp(-\gamma(\bar{x} - \frac{\gamma\sigma^2}{2})).$$

The measure, X_{eff} is obtained by inverting the expected utility:

$$X_{\text{eff}} = \bar{x} - \frac{\gamma\sigma^2}{2}.$$

To permit the comparison between X_{eff} values for different Δt the measure is annualized:

$$X_{\text{eff,ann}} = \frac{1\text{year}}{\Delta t}\left(\bar{x} - \frac{\gamma\sigma^2}{2}\right). \quad (3)$$

The measure in Eq. (3) still depends on the choice of Δt and does not reflect changes occurring at shorter and longer horizons. The final form of X_{eff}, therefore, constitutes a weighted average of $X_{\text{eff,ann}}$ for n different time horizons:

$$X_{\text{eff}} = \frac{\sum_{i=1}^{n} w_i X_{\text{eff,ann}}(\Delta t_i)}{\sum_{i=1}^{n} w_i}. \quad (4)$$

The weights, w_i, in Eq. (4) are obtained through the following formula:

$$w(\Delta t) = \frac{1}{2 + \left(\log\left(\frac{\Delta t}{90\text{days}}\right)\right)^2}. \quad (5)$$

The X_{eff} measure of Eq. (4) is obtained assuming a constant risk aversion. A more realistic assumption is that investors are more risk averse to the clustering of losses than they are to the clustering of profits. The R_{eff} algorithm introduces two levels of risk aversion:

$$\rho = \begin{cases} \rho_+, & \text{if } \Delta\tilde{\text{R}} \geqslant 0 \\ \rho_-, & \text{if } \Delta\tilde{\text{R}} < 0 \end{cases},$$

where $\rho_- > \rho_+$. The corresponding utility function is:

$$u(\Delta\tilde{R}) = \begin{cases} -\frac{\exp\left(-\rho_+ \Delta\tilde{R}\right)}{\rho_+}, & \text{for } \Delta\tilde{\text{R}} \geqslant 0 \\ \frac{1}{\rho_-} - \frac{1}{\rho_+} - \frac{\exp\left(-\rho_- \Delta\tilde{R}\right)}{\rho_-}, & \text{for } \Delta\tilde{\text{R}} < 0 \end{cases}.$$

The return can be obtained by inverting the utility function:

$$\Delta\tilde{R} = \begin{cases} -\frac{\log(-\rho_+ u)}{\rho_+}, & \text{for u} \geqslant -\frac{1}{\rho_+} \\ -\frac{\log\left(1 - \frac{\rho_-}{\rho_+} - \rho_- u\right)}{\rho_-}, & \text{for u} < -\frac{1}{\rho_+} \end{cases}. \quad (6)$$

The computation of the utility for one time horizon Δt_j is calculated using returns, $\Delta\tilde{R}_{ji}$, observed at different and potentially overlapping time intervals, Δt_{ji}. The mean utility for Δt_j is:

$$u_j = \frac{\sum_{i=1}^{N_j} \Delta t_{ji} u\left(\Delta\tilde{R}_{ji}\right)}{\sum_{i=1}^{N_j} \Delta t_{ji}} = \frac{\sum_{i=1}^{N_j} \Delta t_{ji} u_{ji}}{\sum_{i=1}^{N_j} \Delta t_{ji}}. \quad (7)$$

The mean utility of Eq. (7) can be transformed back to an effective return for the horizon Δt_j, $\Delta\tilde{R}_{\text{eff,j}}$, using Eq. (6). Thus, the annualized $\Delta\tilde{R}_{\text{eff,j}}$ is defined as:

$$R_{\text{eff,j}} = \frac{1\text{year}}{\Delta t_{\text{eff,j}}} \Delta\tilde{R}_{\text{eff,j}},$$

where $\Delta t_{\text{eff,j}} = \sum_{i=1}^{N_j} (\Delta t_{ji})^2 / \sum_{i=1}^{N_j} \Delta t_{ji}$. The multihorizon version of R_{eff} is as before obtained by taking a weighted average of $R_{\text{eff,j}}$ for different time horizons:

$$R_{\text{eff}} = \frac{\sum_{j=1}^{n} w_j R_{\text{eff,j}}}{\sum_{j=1}^{n} w_j},$$

with the weights, w_j being determined through Eq. (5).

VI. Simulation Methodology

Simulating exchange rate return series from a Data Generating Process (DGP), transforming them into prices and feeding them into a trading model permits us to calculate the probability distributions of different performance measures [6], [15]. In turn, we can test the null hypothesis that the performance of a trading rule can be attributed to standard statistical properties of exchange rate series against the alternative that the observed performance is due to the capability of the rule in detecting patterns that are not in accordance with traditional DGPs. Further, comparisons can be made across rules and DGPs.

We employ three null processes. For the first process we assume that prices follow a Random Walk (RW) with a drift. Consequently, $\log$ returns are generated according to the following equation

$$r_t = \phi_0 + e_t, \quad (8)$$

where ϕ_0 is the sample mean, and $e_t \sim \mathcal{N}(0, \sigma^2)$. The simulated series for the RW model are obtained by adding normally distributed random numbers (with mean zero and standard deviation equal to the sample standard deviation of the residuals) to the returns mean. The artificial returns are independent and identically distributed by construction, while the simulated price series follow a random walk with the same drift and standard deviation as the original series. The artificial return series are transformed into price series by using the first price of the sample. However, according to the RW model the volatility of returns is constant which contrasts with the stylized fact that the foreign exchange market is characterized by time varying volatility [16]. The presence of serial correlation in the second moment of the distribution of the exchange rate series motivates the use of GARCH models. GARCH models are nonlinear conditionally Gaussian models where the conditional variance depends

on its lagged values, as well as, on past error terms. The GARCH(1,1) may be written as:

$$\begin{aligned} r_t &= \phi_0 + u_t, \\ u_t &= e_t\sqrt{h_t}, \quad \text{where } \mathrm{e_t} \sim \mathcal{N}(0,1), \\ h_t &= \omega + \alpha u_{t-1}^2 + \beta h_{t-1}. \end{aligned} \tag{9}$$

The parameter values of the GARCH(1,1) model are estimated by using a Quasi Maximum Likelihood procedure and the simulated series are generated by drawing values from the standard normal distribution and calculating recursively Eq. (9). Finally, we employ an ARMA(2,1)–GARCH(1,1) model so as to take into account possible serial correlation in both the mean and the variance:

$$\begin{aligned} r_t &= \phi_0 + \phi_1 r_{t-1} + \phi_2 r_{t-2} + \theta_1 u_{t-1} + u_t, \\ u_t &= e_t\sqrt{h_t}, \quad \text{where } \mathrm{e_t} \sim \mathcal{N}(0,1), \\ h_t &= \omega + \alpha u_{t-1}^2 + \beta h_{t-1}. \end{aligned}$$

The same procedure as for the GARCH model is applied in order to simulate data.

Table I presents descriptive statistics for the real return series, as well as, the GARCH(1,1) and ARMA(2,1)–GARCH(1,1) models. The p-values are reported in parentheses next to the estimate of each coefficient. For GARCH(1,1) and ARMA(2,1)–GARCH(1,1) models the p-values were calculated using the procedure of Bolleslev and Wooldridge [17]. A clear implication of the results is that both volatility clustering and the ARMA structure are present in the data set. The kurtosis reported in the first column of Table I shows that the return series is highly leptokurtic. This phenomenon was first documented by Mandelbrot [18] in commodity markets and it implies that the normality assumption is violated[1]. Further, the standardized residuals from the GARCH and ARMA–GARCH models also appear to exhibit more density around the mean and fatter tails than normal. It follows that a better approximation of the true DGP can be established by relaxing the normality assumption.

TABLE I
DESCRIPTIVE STATISTICS

Model	RW	GARCH	ARMA–GARCH
ϕ_0	7.71e-05 (0.5859)	6.25e-05(0.6595)	8.79e-05 (0.5149)
ϕ_1	–	–	-0.96561 (0.0000)
ϕ_2	–	–	-0.04164 (0.0820)
θ_1	–	–	0.93178 (0.0000)
ω	–	5.89e-07 (0.0295)	1.50e-06 (0.0054)
α	–	0.00996 (0.1020)	0.01584 (0.0765)
β	–	0.97263 (0.0000)	0.94184 (0.0000)
K_e	-0.3107	-0.3303	-0.3270
S_e	4.5849	4.5561	4.7164
JB_e	212.1939	209.2158	246.7241
p_{JB}	0	0	0

Bootstrapping is a widely used methodology for testing the statistical significance of the performance of trading models. The non-parametric procedure adopted is called resampling [19]. Resampling is in effect drawing with replacement from the sample under examination. The suitability of this approach in the present context is due to the fact that it utilizes the empirical distribution function of the data and therefore, it addresses issues such as leptokurtosis and skewness. Artificial price series are created by bootstrapping from the residuals of the RW and the standardized residuals of the GARCH and ARMA–GARCH in order to calculate the probability distributions of the performance measures under examination.

[1]The p-values corresponding to the Jarque Bera test statistic are virtually zero in all cases, rejecting the null hypothesis of normality

VII. PRESENTATION OF EXPERIMENTAL RESULTS

The dataset presently employed is the daily noon New York buying rates for the US Dollar against the Japanese Yen exchange rate from the H10 Federal Statistical Release. The 5292 observations cover the period from 3/1/1985 to 2/1/2007. In addition to the price series, a normalized series is also provided as input to the algorithm. The normalized series is constructed by dividing each observation with the 250-day moving average [15]. Each input pattern contains the current price and the normalized price, while the algorithm can access past prices using the non-terminal node lag. The maximum lag that the algorithm is allowed to consider is 250. The first 3014 patterns were assigned to the training set, the next 502 patterns are assigned to the validation set, while the last 1508 patterns comprised the test set. The inclusion of a validation set was used to alleviate the problem of overfitting. The fitness of an individual on the validation set was only used during the assignment of the best individual identified during the execution. For both GMA and GP, a rule was assigned as the best identified so far if it was at least as good as the current best on both the training and the validation set, and it improved on the performance on at least one dataset (Pareto domination).

For GP, the terminal set, $\mathcal{T}$, consisted of:

$$\mathcal{T} = \{X_t^n, X_t, \mathrm{rand}\},$$

where X_t^n stands for the normalized exchange rate at date t, X_t is the non-normalized rate, and `rand` denotes a random real constant in the interval [-1,1]. The function set, $\mathcal{F}$, contained the following functions:

- Ternary functions: if then else
- Binary functions: $+, -, *, /, >, <, \mathrm{and}, \mathrm{or}, \min, \max, \mathrm{ma}, \mathrm{lag},$
- Unary functions: $\log, \exp,$

where ma and lag denote the moving average and the lag of the values of the time series.

A positive evaluation of an individual over a pattern is assumed to signal that the current holdings should be held in the base currency (in this case US Dollars), and vice versa. In particular, if the system at date t, holds US Dollars and the evaluation of the individual over the corresponding input pattern is positive then all the available funds are converted to Yen. On the contrary, if the system holds Yen and the individual evaluation is non-positive, then the amount is

converted to US Dollars. In all other cases, the holdings do not change currency at date t.

The last observation of the series is always employed to convert the final holdings to the base currency. A one-way transaction cost equal to 0.5% and 0.125% for the training, and test periods, respectively, was used. A larger transactions cost was imposed during training to penalize rules that trade very frequently. The fitness function returns the X_{eff} measure with the parameter γ in Eq. (4) set to $\gamma = 0.11$.

Regarding the parameters employed by the GP algorithm, the maximum tree depth, D, at initialization was set to 5, while in subsequent generations D was equal to 8. Population size was 100 and the maximum number of generations was 200. The reproduction, mutation, and crossover probabilities were 0.05, 0.5, and 0.45, respectively. Finally, the probability of performing uniform crossover at each node of the common region was 0.5. The stopping criterion for the algorithm was to reach the maximum number of generations. To optimize the values of the constant nodes in all the GP individuals the Hooke–Jeeves [20] procedure was applied. The GP output was the best individual encountered during its execution. Regarding the parameters employed by the DE algorithm, the population size was equal to 200 and the maximum number of generation to 2000; the mutation and recombination constants were set to $F = 0.1$ and $CR = 0.3$, respectively.

Tables II and III report the minimum ($\min$), mean, maximum ($\max$), and the standard deviation (std) for the number of trades, the mean annualized return, X_{eff}, and R_{eff}, of the GMA rules identified through DE over 50 experiments. Figs. 3 and 4 illustrate the cumulated return of the best performing (on the test set) GMA rule on the training and test set, respectively. In each figure the evolution of the time series on the corresponding data set is also plotted. In 32 cases out of the fifty experiments the DE optimized GMA rule was unable to identify a trading strategy that improved over the simple strategy of holding the base currency over the entire period. For these rules the variance of the return series is zero and hence the computation of the mean and the standard deviation for the Sharpe ratio is performed without taking into consideration these cases. Excluding these no trade strategies, the remaining MA rules performed substantially more trades than the corresponding GP rules. The best performing GMA rule was of the double MA form (i.e. $\theta_1 > 0$, $\theta_2 > \theta_1$, and $\theta_3 = 0$).

Tables IV–V provide the same information for the GP identified trading rules over 50 experiments, while the cumulated return of the best performing rule on the test set is provided in Figs. 5 and 6.

TABLE II

DE–GMA TRAINING SET PERFORMANCE

	$\min$	mean	$\max$	std
num trades:	0.000000	7.560000	38.000000	10.799017
annualized ret:	0.000000	0.360000	3.284896	1.228251
X_{eff}:	-0.032121	-0.006980	0.000000	0.010022
R_{eff}:	-0.042298	-0.010640	0.000000	0.014626
Sharpe Ratio:	0.071835	0.288762	0.388624	0.080077

TABLE III

DE–GMA TEST SET PERFORMANCE

	$\min$	mean	$\max$	std
num trades:	0.000000	5.120000	24.000000	7.241152
annualized ret:	-0.597215	0.360000	1.812633	0.562487
X_{eff}:	-0.035841	-0.007994	0.000116	0.013372
R_{eff}:	-0.038477	-0.008895	0.000000	0.014521
Sharpe Ratio:	-0.088653	0.076804	0.297369	0.139295

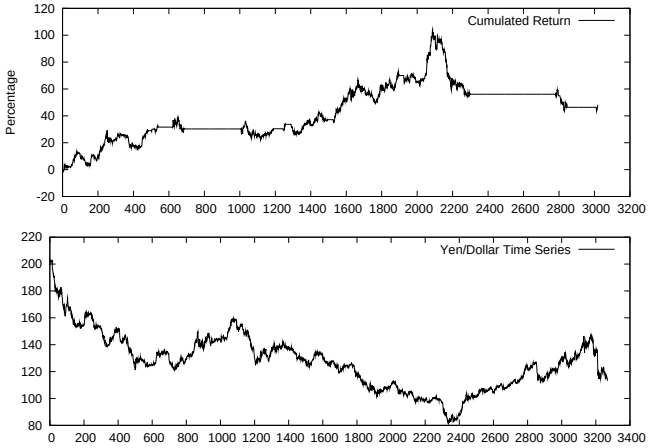

Fig. 3. **Top**: Cumulated return of best performing rule on the training set. **Bottom**: Evolution of time series on the training set.

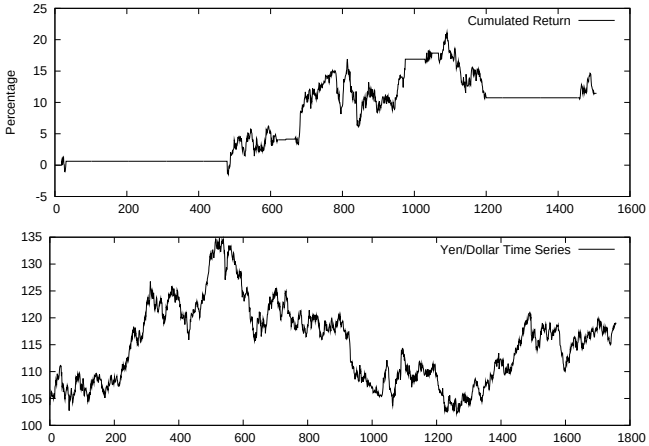

Fig. 4. **Top**: Cumulated return of best performing rule on the test set. **Bottom**: Evolution of time series on the test set.

TABLE IV

GP TRADING RULE PERFORMANCE ON THE TRAINING SET

	min	mean	max	std
num trades:	6.000000	14.100000	36.000000	7.163230
annualized ret:	2.021642	1.000000	5.848466	1.009530
X_{eff}:	0.013441	0.020439	0.030558	0.004680
R_{eff}:	0.015576	0.023328	0.036446	0.005235
Sharpe Ratio:	0.590329	0.785202	0.963684	0.089573

Tables VI and VII present the results for the simulation exercise for the best GP and GMA rules, respectively. For each of the four evaluation measures each table firstly reports the realized values for the corresponding rule (Realized).

TABLE V

GP TRADING RULE PERFORMANCE ON THE TEST SET

Test Set:	min	mean	max	std
trades:	0.00000	8.933333	32.000000	6.144769
annualized ret:	-1.633383	0.950000	2.669512	1.075628
X_{eff}:	-0.037450	-0.018569	0.013903	0.011517
R_{eff}:	-0.048849	-0.021780	0.015108	0.013432
Sharpe Ratio:	-0.451822	-0.118973	0.513813	0.242092

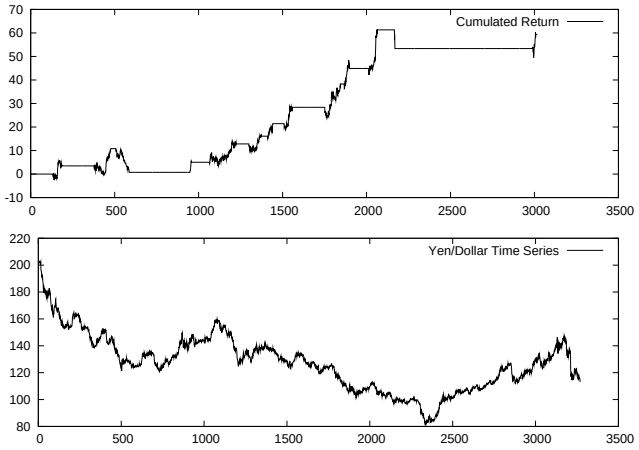

Fig. 5. **Top**: Cumulated return of best GP rule on the training set. **Bottom**: Evolution of time series on the training set.

The various p-values corresponding to the different null models for both the bootstrap and the simulation based on the normality assumption are reported in the subsequent rows of the tables. These values are calculated using 1000 simulations from each DGP and are defined as the percentage of times that a rule yielded a higher value for each performance measure on the artificial series than on the real series. Overall, the performance of the GP rule on the actual test set exceeds that of the optimized GMA rule with respect to all measures. For the GP rule, all measures are positive indicating that annualized returns after transactions costs cover the cost of risk taken by the model. It is also interesting to note that X_{eff} is smaller than R_{eff} showing that penalizing the clustering of profits equally with the clustering of losses has caused an overestimation of risk for this rule. The GMA rule also generates positive annualized returns after transactions costs. However, X_{eff} is very close to zero and R_{eff} is negative, suggesting that there is a significant clustering of losses (see Fig 4).

The results indicate that the performance of the GP rule is also superior in terms of statistical significance. The null hypothesis that the examined DGPs can explain the performance of the GP rule is rejected at the five percent significance level for all measures. On the contrary the same null hypothesis for the GMA rule is not rejected for any measure. The examination of the p-values of the various performance measures reveals that they are higher for the Sharpe Ratio, S_I, and the annualized return for all DGPs for both models. As indicated in [6] an explanation for this finding is that these two measures use limited information

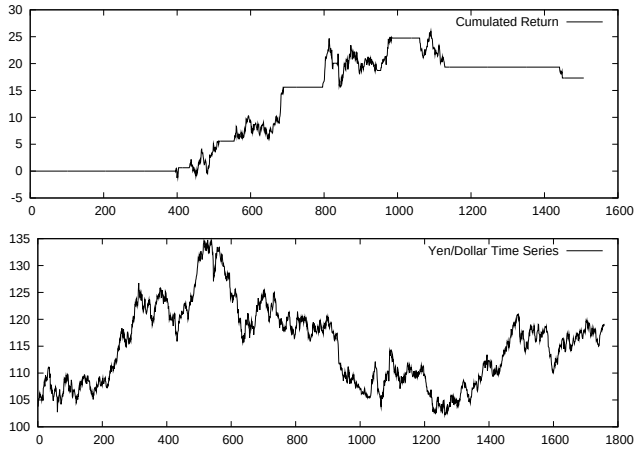

Fig. 6. **Top**: Cumulated return of best GP rule on the test set. **Bottom**: Evolution of time series on the test set.

from the entire return path.

The fact that the p-value of the Bootstrap ARMA–GARCH DGP is the higher for the GP rule suggests that part of the models predictability can be attributed to the ARMA–GARCH structure of the underlying series. Finally, it is noted that the normality assumption results in rejecting the null hypothesis much more frequently than the bootstrap, especially for the GP rule.

TABLE VI

BEST GP RULE PERFORMANCE

	Annual Return	X_{eff}	R_{eff}	S_I
Realized	2.6695	1.3903	1.5108	0.5138
Bootstrap				
p RW	0.04	0.026	0.026	0.046
p GARCH	0.031	0.028	0.022	0.039
p ARMA–GARCH	0.045	0.035	0.036	0.037
Normal				
p RW	0.032	0.025	0.023	0.036
p GARCH	0.026	0.02	0.017	0.03
p ARMA–GARCH	0.035	0.027	0.024	0.034

TABLE VII

BEST GMA RULE PERFORMANCE

	Annual Return	X_{eff}	R_{eff}	S_I
Realized	1.8126	0.0116	-0.1188	0.2974
Bootstrap				
p RW	0.124	0.074	0.07	0.112
p GARCH	0.117	0.079	0.061	0.106
p ARMA–GARCH	0.13	0.075	0.069	0.107
Normal				
p RW	0.099	0.058	0.052	0.087
p GARCH	0.100	0.061	0.056	0.087
p ARMA–GARCH	0.124	0.081	0.067	0.098

VIII. CONCLUSIONS

Technical analysis has a long history in financial markets and its application in the foreign exchange market is gaining ground according to the evidence accumulated over the past years. Along with conventional trading rules there is a growing interest in the development of automated

methods to detect novel patterns in the data. In this paper, we considered Genetic Programming to address this task and compared its performance to traditional moving average rules. The parameters of the latter were optimized through the Differential Evolution algorithm.

Both algorithms were capable of generating highly profitable rules in the portion of the data used for training. On the test set, the moving average rules proved to be more robust compared to the Genetic Programming rules. However, Genetic Programming managed to create the most profitable rule encountered. Moreover, a statistical evaluation of the properties of the best moving average rule showed that the null hypothesis that its performance is attributable to well–known properties of the data–generating process cannot be rejected. The opposite held for the GP identified rule. Another interesting feature of these rules was their ability to take accurate positions long into the future.

REFERENCES

[1] R. Sullivan, A. Timmermann, and H. White, "Data-snooping, technical trading rule performance, and the bootstrap," *Journal of Finance*, vol. 54, no. 5, pp. 1647–1691, 1999.

[2] 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*, vol. 20, no. 4, pp. 439–471, 2001.

[3] B. L. Baron, "Technical trading rule profitability and foreign exchange intervention," *Journal of International Economics*, vol. 49, no. 1, pp. 125–143, 1999.

[4] P. D. Grauwe and M. Grimaldi, "Exchange rate puzzles. a tale of switching attractors," *European Economic Review*, vol. 50, no. 1, pp. 1–33, 2006.

[5] D. Olson, "Have trading rule profits in the currency market declined over time?" *Journal of Banking and Finance*, vol. 28, no. 4, pp. 85–105, 2004.

[6] R. Gençay, M. Dacorogna, U. Müller, and O. Pictet, "Effective retrun, risk aversion and drawdowns," *Physica A*, vol. 289, no. 1–2, pp. 229–248, 2001.

[7] F. Fernández-Rodríguez, C. González-Martel, and S. Sosvilla-Rivero, "Optimization of technical rules by genetic algorithms: evidence from the madrid stock market," *Applied Financial Economics*, vol. 15, pp. 773–775, 2005.

[8] R. Storn and K. Price, "Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

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

[10] B. Wolfgang, P. Nordin, R. Keller, and F. Francone, *Genetic programming: An Introduction: on the automatic evolution of computer programs and its applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

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

[12] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel differential evolution," in *IEEE Congress on Evolutionary Computation (CEC 2004)*, 2004.

[13] D. Tasoulis, V. Plagianakos, and M. Vrahatis, "Clustering in evolutionary algorithms to efficiently compute simultaneously local and global minima," in *IEEE Congress on Evolutionary Computation*, vol. 2, Edinburgh, UK, 2005, pp. 1847–1854.

[14] W. F. Sharpe, "Mutual fund performance," *Journal of Business*, vol. 39, no. 1, pp. 119–138, 1966.

[15] 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*, vol. 32, no. 4, pp. 405–426, 1997.

[16] R. F. Engle, T. Ito, and W.-L. Lin, "Meteor showers or heat waves? heteroskedastic intra-daily volatility in the foreign exchange market," *Econometrica*, vol. 58, pp. 525–542, 1990.

[17] T. Bollerslev and J. M. Wooldridge, "Quasi–maximum likelihood estimation and inference in dynamic models with time varying covariances," *Econometric Reviews*, vol. 11, pp. 143–172, 1992.

[18] B. Mandelbrot, "The variation of certain speculative prices," *Journal of Business*, vol. 36, pp. 394–419, 1963.

[19] B. Efron, *The jackknife, the bootstrap and other resampling*. SIAM, Philadelphia, PA, 1982.

[20] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in Fortran 77*. Cambridge University Press, 1992.