# Effective Searching for Profitable Forex Trading Rules via Genetic Programming: A New Two-Phase Searching Methodology *

Zelin Chen †

*Department of Economics*

*University of Melbourne*

October 16, 2018

*Submitted in part fulfilment of the requirements for the degree of Master of Applied Econometrics of the University of Melbourne, Oct 2018. I appreciate Tomasz Woźniak, Yong Song and Joe Hirschbergfor for helpful comments.

†Contact information: zelinc@outlook.com or https://github.com/ZelinC

Declarations

1. This Project Report is the sole work of the author whose name appears on the title page and it contains no material which the author has previously submitted for assessment at the University of Melbourne or elsewhere. To the best of my knowledge and belief, the report contains no material previously published or written by another person except where due reference is made in the Project Report. I declare that I have read, and in undertaking this research I have complied with, the University's Code of Conduct for Research. I also declare that I understand what is meant by plagiarism and that this is unacceptable. Except where I have expressly indicated otherwise, this Project Report is my own work and does not contain any plagiarised material in the form of unacknowledged quotations or mathematical workings or in any other form.

2. Any data sources used to derive the results presented in this Project Report have been acknowledged. Sufficient information has been provided on data sources, data periods, and data manipulation and transformation to allow for the full replication of the results.

3. This Project Report is 3956 words in length for the main body from section 1 to 5, and the estimation may be biased. I understand that I will be penalised if the Project Report is longer than 4,000 words.

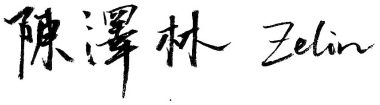
4. I would like to express my appreciation to:

My supervisor Dr.Tomasz Woźniak

My Master course coordinator Associate Professor Jenny Lye

My parents

My girlfriend Ruiyu Su



15/10/2018

**Abstract**

For this report, we replicate a genetic programming based trading rule searching system proposed by Neely et al. (1997) and Allen & Karjalainen (1999) with modifications and extensions using R. We searched 10,000 candidate trading rules using their methods based on combinations of functions that we pre-defined as benchmark results. We also searched another 10,000 rules by the newly proposed second-phase searching methodology. After evaluating trading rules' performance in the out-of-sample period, we find that, first, the proposed second-phase searching methodology can discover better match-ups of long and short indicators using training period data. We also find second-phase searched rules do perform better on average than those rules from the benchmark search in the test period. Second, GP, at least for the second-phase searched top rules, can discover profitable trading rules. However, by investigating the decision of top rules in the test period, we find that the profitability may not all due to active predictability. Instead, it is due to the imitation of the buy-and-hold strategy.

# Contents

# 1 Introduction

Biologically inspired genetic algorithms have broad applications in economics and finance. They have been proved useful in optimisation and model induction in economics. In financial econometrics, the underlying data generating process of a time series is usually unknown and is difficult to reveal, since both model structure and parameters need to be explored. Therefore, it has given genetic algorithms a promising role in these fields. In particular, these methods are helpful in financial forecasting, credit risk assessment, portfolio optimisation, pricing complex financial instruments (e.g. options) and algorithmic trading (Brabazon et al. 2008).

The genetic algorithm (GA) was initially developed by Holland (1962, 1975), who is inspired by the principles of natural evolution. It is ideal for solving optimisation of a problem when its solution form is unknown. Given a group of randomly generated candidate solutions, GA evolves them under "natural selection" rules into a group of better solutions. The representation of a candidate solution is a chromosome which includes a fixed length binary encoding. The binary encoding contains the information about the model structures or optimisation settings. However, GA is sometimes restricted in solution searching as its solution representation in chromosome structure has fixed length. Koza (1992) developed genetic programming (GP), as an extension of GA, that partly loosens the restriction of the fixed length of solution structures. GP presents solutions in a flexible length tree structure.

In the mid-1990s, GP was a popular vehicle for trading rule exploration. The recent research interest has moved to derivative pricing and volatility prediction (Brabazon et al. 2008). The application of genetic programming in financial market research started with pioneers Allen & Karjalainen (1999) and Neely et al. (1997). Neely et al. (1997) use the genetic programming method to search for an optimised combination of mathematical operators based on trading rules. From the exchange rate of the US dollar against four currencies, they find strong evidence of economically significant out-of-sample excess returns. They also prove that the trading rules designed by GP can detect patterns in the data that are not captured by conventional econometric models. Dempster & Jones (2001) also investigate GP's reliability in foreign exchange trading rule finding but focused on high-frequency data. However, their rules are not profit-making as they over-react to short-term market behaviour.

The performance of GP in searching profitable rules in the stock market, however, is disappointing. Allen & Karjalainen (1999) apply GP to the S&P 500 index. Their rules do not present consistent excess returns compared to a simple buy-and-hold strategy after subtracting transaction cost. However, GP inspired rule can stay in the market

given positive daily return and low volatility, and out when the market is in the opposite condition. Kaboudan (1999) provide rigorous experiments on the predictability of GP by letting GP predict time series constructed from the data generating process. They found linear and non-linear processes free from noise are both predictable. Predictability is weakened when a linear or non-linear process includes stochastic components, and ultimately losing power for random series. When applying the finding in eight DJ stocks intraday data, they find that no evidence of predictability. Wang (2000) introduce a more flexible structure in GP to allow decision-making in both trading and hedging. They find that the S&P 500 spot market is efficient as most generated rules followed the buy-and-hold strategy and the out-of-sample performance is disappointing as they discover no out-performing rules.

GP and GA are also good at collaborating with statistical models for input variable searching. Thinyane & Millin (2011) propose a hybrid system with a combination of GA and the Artificial Neural Network (ANN) to generate FX trading rules. The first phase of the system uses GA to find trading signals from various technical trading rules for profit maximisation. In the second phase, these signals are fed into ANN and output a single signal of whether to buy, hold or sell. They use daily USD exchange rate against other ten currencies and find 5% return on average when looking per currency under the test period.

The existing literature about the application of GP and GA is rich. However, the best candidate solution form representation under GP remains inconclusive. Therefore, this paper joins the discussion and presents a two-tree-structured rule representation and a two-stage crossover method for evolving this type of rule representation. Section 2 provides a detailed explanation of the genetic programming methodology, section 3 introduces the data and training method, and section 4 presents the results.

## 2 Genetic Programming

This report uses genetic programming (GP) to search for profitable trading rules in the currency market. Figure 1 summaries GP working procedure in finding a trading rule. The procedure starts with a group of randomly generated trading rules. This starting group is called the initial population in GP terminology. Each rule within the initial population has a fitness which represents each rule's performance during the training period. Next, rules from the initial population participate in the construction of the new population through three evolution functions: elitism, crossover and mutation. The participation probability is biased towards rules with higher fitness, therefore

rules performance better be more likely to participate in new population construction. Then, the fitness of rules in the new population is computed again. In an iterative setting of GP, the next population soon becomes the old population and starts to generate the new population again using the same evolution functions. GP process takes multiple iterations and then stop at the latest population. Although no guarantee for global optima, the rule with the highest fitness in the latest population is usually the local optima, we can find in the training period using only the parts of rules we randomly generated at the beginning.

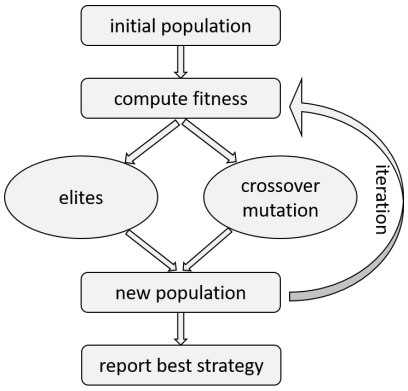



Figure 1: Genetic Programming
source: Koza (1992)

This section will provide a detailed explanation of the above GP overview. 2.1 explains trading rule structure and its random construction. 2.2 explains the evaluation of a rule regarding its fitness. 2.3 explains the main algorithm, three evolution functions and hyper-parameter settings. 2.1-2.3 describe the current method in the literature, as an addition, 2.4 proposes a two-phase searching methodology.

## 2.1 Rule representation and construction

Neely et al. (1997) and Allen & Karjalainen (1999) use one single rule to switch between the long and short position in the market. In contrast, this paper borrows the idea from Matsui & Sato (2009). They separate chromosome representation for a trading rule into buy and short indicators, which intended to capture the best occasion for buy and sell. Similarly, using GP, one trading rule can consist of one long indicator and one short indicator, and both of them follow a tree representation. A tree-structured indicator is constructed from functions randomly chosen from a large candidate function set and each function's parameter set if required.

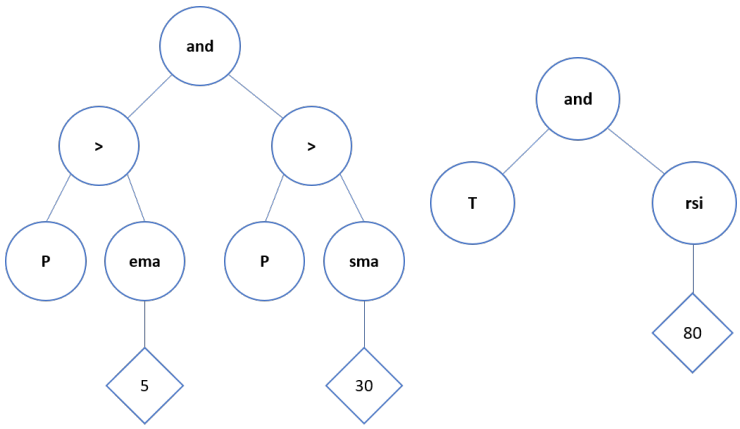



Figure 2: A sample trading rule: long indicator (left) and short indicator (right)

Figure 2 provides an example of a randomly generated trading rule. The left is the long indicator, and the right is the short indicator. The long indicator is read like that if the price is larger than its 5-day exponential moving average, and also larger than its 30-day simple moving average, it indicates a buy signal. The short indicator says that if True and if RSI is larger than 80, it indicates a sell signal. Within the rule, values 5, 30 and 80 are the parameters associated with functions 'EMA', 'SMA' and 'RSI'. Figure 7 provides an example of more complex tree structure.

For the construction of a single tree-structured indicator, the construction starts from the root node which is the top level node (level 1) of the tree by randomly selecting a function from the function set $\{\text{AND}, \text{OR}, \text{XOR}\}$. The level 1 node takes two Boolean inputs and outputs the final long or short decision which is also a Boolean. The level 2 nodes are the nodes that supply inputs for node 1. Therefore, we randomly select level 2 nodes from the functions that output a Boolean. Next, each level 2 node will be assigned its connecting nodes from the next level. It is a recursive process of going deep in the structure of the tree, and the nodes end if the selected function comes from a function that requires no inputs.[1] Table 4 in the Appendix summarises the functions that contribute in indicator construction. The functions that build a rule some come from scratch and some are from Technical Trading Rules (TTR) implemented in existing R packages.

[1]Recursive programming is used for the tree structure construction to avoid redundant code. The construction of the code is guided by Poli et al. (2008) on page 14.

## 2.2 Fitness

A metric is needed for performance comparison among rules, and the metric is called the fitness. The fitness is represented by the trading rule's close balance, as we are interested in how effectively a trading rule can generate profits in the end. Fitness serves for the similar purpose as cost or loss functions in econometrics, but it does not penalise wrong decision making at each trading day, instead only look at the final results of a rule. Fitness is the necessity for evolution functions since the construction of new populations rely on rules that are randomly selected based on their fitness.

Evaluation of a rule's fitness is a process of implementing its daily decisions and aggregating realised profits throughout the time horizon. At each period $t$, two signals are obtained from long and short indicators, determined as $l_t$ and $s_t$ correspondingly. $l_t$ and $s_t$ both have two possible outcomes 1 and 0, which are the numeric counterpart for Boolean True and False. When $l_t$ reports 1, it indicates a long signal, and when $s_t$ reports 1, it indicates a short signal. On the other hand, we obtain no indications when both output 0.

$$l_t \in \{0, 1\}$$

$$s_t \in \{0, 1\}$$

$d_t$ represents the current market position or decision at time $t$, and it is determined by long and short signals based on previously periods' market information. At the first-period market entry, starting position $d_1$ is 0, indicating not in the market position. For each following period $t > 1$, the market position switches between long and short, and decision $d_{t+1}$ are determined by the following conditions.

$$d_{t+1} = \begin{cases} d_t, & \text{for } l_t = s_t \\ 1, & \text{for } l_t = 1 \text{ and } s_t = 0 \\ -1, & \text{for } l_t = 0 \text{ and } s_t = 1 \end{cases}$$

The above functions indicate that market position only alters when non-conflicting signals are received from $l_t$ and $s_t$. If both long and short signals reveal at the same, no position changes will occur. For the value of market position

$d_t$, 1 means go long in the market, $-1$ means go short in the market. It expects no more out-of-market decision ($d_t = 0$) after the first market entry because we plan to capture the profit from both sides of market movement by switching between long and short positions. For the exchange rate of a given currency pair, we obtain its close price $p_t$ at each period $t$. The return $r_t$ for $p_t$ at period $t$ is computed in Equation 1.

$$r_t = \frac{p_t}{p_{t-1}} - 1 \tag{1}$$

Equation 2 indicates the return $\pi_t$ for trading rule $i$ given market position $d_t^i$ determined at period $t-1$. $c$ is the spread cost for a transaction and $n_t$ determines whether a transaction happened in period $t$.

$$\pi_t^i = r_t d_t^i - c n_t. \tag{2}$$

Overall, the fitness for rule $i$ is its cumulative profit plus its opening balance $b_0$ as shown in Equation 3.

$$f_i = b_0 \prod_{t=1}^{T} (1 + \pi_t^i) \tag{3}$$

### 2.3 Population and Evolution

Genetic programming is an iterative process, as it evolves a group of candidate trading rules repeatedly into a group of better ones. At the beginning of the iteration, it randomly generates a group of trading rules, and this group is called the initial population. Next, GP evolves the initial population into its next population which is a new population. During this process, it implements three evolution functions: crossover, mutation and elitism. Following Neely et al. (1997) and Allen & Karjalainen (1999), the detailed explanation of the algorithm of GP shown in Figure 1 are stated as follows:

---

I. Create an initial population consists of $n$ randomly generated trading rules and compute the fitness for each rule.

II. Evolve the current population into a new population via the following:

a) (Elitism) the top $n * e$ rules are copied into the new population, where $e$ is the elite ratio.

b) The rest $n - n * e$ rules in the new population are constructed from the process of crossover and mutation.

The following process creates one new rule, and it runs for $n - n * e$ iterations:

i) Select two rules from the current population based on the Roulette Wheel selection probability.[2]

ii) For two long indicators taken from two rules, do the following steps (crossover and mutation) to create a new long indicator. Then repeat the following steps for two short indicators.

1) Randomly select one level 1 node for the new indicator from two level 1 nodes in the selected indicators.

2) The following nodes connected to the level 1 node for the new indicator are either each randomly chosen from selected indicators under probability $1 - m$ (Crossover) or randomly generated completely under probability $m$ (Mutation).

3) Assemble randomly chosen nodes into the new indicator. Figure 8 provides a visualisation of crossover and mutation process.

iii) Combine new long indicator and new short indicator into a new trading rule, then put the new trading rule into the new population.

vi) Combine the new long and short rules into a new rule, then put this newly generated rule into the new population.

c) Compute the fitness for each rule in the new population.

III. Repeat II for $\eta$ iterations.

IV. Output the rule that produces the highest fitness in the last population (i.e. after $\eta$ iterations).

---

Neely et al. (1997) and Allen & Karjalainen (1999) called the above process as one trial of genetic programming, as one trial generates one locally optimal trading rules among all available information within the initial population. Table 1 provides a summary of hyper-parameters for one trial. In each trial, we first randomly generate 30 rules to form the initial population. Each indicator has a maximum level of 10 to prevent infinite recursive construction a tree. The initial population evolves for 20 iterations, and the rule with the best fitness in the latest population will be taken out as one candidate trading rule. We conduct the trial for 10,000 times to obtain a group of good trading rules.

[2]The probability distribution used for rule selection is not specified in Neely et al. (1997) and Iskrich & Grigoriev (2017), as they both only mentioned the use of probability distribution that is biased to better rules. We follows the Roulette Wheel selection probability distribution specified in Iskrich & Grigoriev (2017), each rule has probability $\frac{f_i}{\sum_i^N (f_i)}$ of being selected, where $f_i$ is the fitness of rule $i$.

The feature that each trial is independent provides an opportunity for parallel computing to cut down computational time dramatically. We are inspired by Straßburga et al. (2012) who provide evidence about the improvement of the computational efficiency of the genetic algorithm in MATLAB using parallel tools. Given the nature that each R only runs on one core, parallel computing is easily achievable via operation of multiple R sessions simultaneously. For 10,000 trials, we open 25 R sessions, and each operates 400 trials.[3] Then results from 25 R sessions are combined into the final candidate rule set. Overall, we call the process of searching for these 10,000 rules the first-phase searching.

| hyper-parameter | notation | value |
|---|---|---|
| population for each trial | n | 30 |
| iteration for each trial | $\eta$ | 20 |
| maximum level of trees | $\gamma$ | 10 |
| elite ratio | e | 10% |
| elite size | n*e | 3 |
| mutation rate | m | 5% |

Table 1: Summary of GP hyper-parameters in first-phase evolution

## 2.4 Two-phase evolution method

Each rule found in the first-phase comes from 20 iterations on a population of 30 rules, that means each rule's match-up of long and short indicators is only using information within that small population. However, we may be able to find better match-ups of long and short indicators by searching among all 10,000 relatively good rules. We believe that an excellent long indicator in one rule may have a better short indicator match-up sitting in other rules. The second-phase searching is inspired from this rationale, and it tries to explore the existence of better match-ups of long and short indicators within the rules obtained in the first-phase searching.

In the second phase, all 10,000 first-phase searched rules altogether are treated as initial population, similar to the first phase, elitism, crossover and mutation are implemented for new population evolution. This process will run for multiple but relatively shorter iterations, as instead of looking for only the best one in the population, we would like to keep a group of best rules with variations. Shorter iterations help to reduce the situation that a small group of rules or a specific part in a rule become popular and dominant in the population. Because if one rule is

[3]Great appreciation to Dr.Tomasz Woźniak for providing the workstation which has 32 cores and 2.0 GHz processor frequency.

outperforming, it is more likely to be selected for evolution and, via several iterations, it will produce many rules with the same or similar parts.

| hyper-parameter | notation | value |
|---|---|---|
| total population | n | 10,000 |
| iteration | $\eta$ | 5 |
| elite ratio | e | 0.5% |
| elite size | n*e | 20 |
| mutation rate | m | 5% |

Table 2: Summary of GP hyper-parameters in second-phase evolution

Figure 3 summaries the proposed two-phase evolution methodologies and Table 2 lists the second-phase GP configuration. In the first phase, we run 10,000 independent GP trials and obtain 10,000 trading rules. In the second phase, we treat all 10,000 rules as the initial population and evolve it for five iterations, then take out the top rules in the latest population.

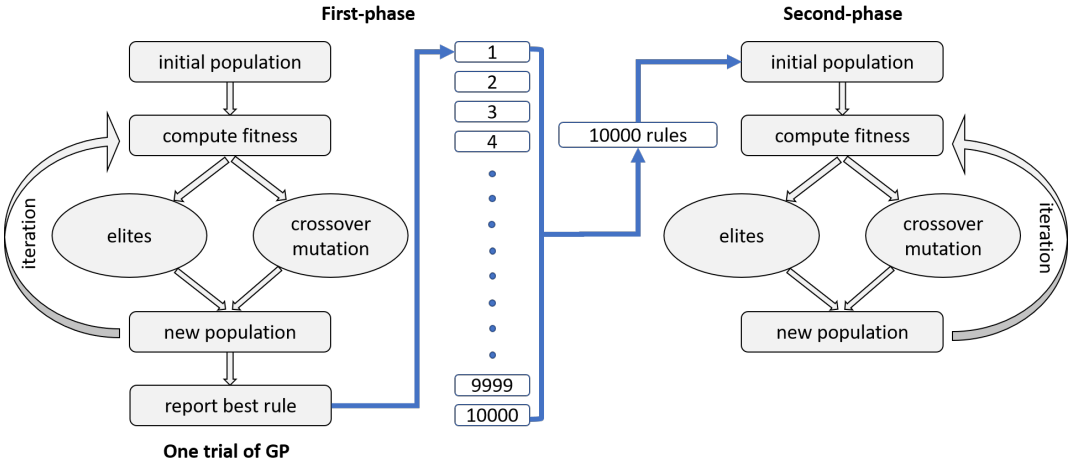



Figure 3: Two-phase Genetic Programming

## 3 Data, training and filtering

This report aims to search for profitable trading rules for the USD/JPY currency pair from daily data. The data contains the daily close, high and low price of USD/JPY for a period starting on 01/01/1994 and finishing on 31/12/2017. Forex market runs overnight and only closes at the weekends, the data each trading day is collected

at its midnight. The data is downloaded from Meta-Trader 4, which is a popular and standard platform among individual traders.[4] Given no commission using this platform, the cost of trading is mainly through the spread, which costs 0.00001 per trade for USD/JPY.


Figure 4: USD/JPY daily excange rate 1994-2017

Figure 4 is the whole sample USD/JPY exchange rate. We see the price is fluctuating around during the period, and observe no clear long-term trend. It indicates that a simple buy-and-hold strategy will not make a significant profit over time. In comparison, given observed the fluctuation we prefer rules that can respond actively to the up-and-down cycle of the price.

We separate the whole sample into three periods: training, validation and test periods. The training period contains 14 years data starting on 01/01/1994 and finishing on 31/12/2007. Our rules are searched using their fitness during this period. The Validation period starts on 01/01/2008 and finishes on 12/31/2013 is implemented as a filter to drop out those over-fitted rules. Rules searched in the training period have not seen the validation period data, therefore using fitness during this period can select out those rules that have good predictability or generalisation in the future market. In the end, we check the performance of those rules survived in the validation period for another four-year horizon starts from 01/01/2014 and finishes on 31/12/2017. The price movement in the test period has an entirely different shape as it is in the validation period. It starts from a quick surge then follows by a huge slump and a quick recovery, and finished with relatively tranquil fluctuations. We are interested in

[4]The data is download from Meta Trader 4 using real trading account opened at Pepperstone.com.

comparing performance between the first-phase and the second-phase searched rules in the test period on average. However, practically it is not sensible for traders to implement all profitable rules found after the validation period. Instead, traders tend to pick and implement those best performing rules during the validation period. We, therefore, select the top 50, top 20, top 10, top 5 rules as well as the best rule based on fitness in the validation period, and track their performance in the test period.

# 4 Results

## 4.1 Fitness boost in second-phase searching

We first present how well the second-phase is snooping better match-ups of long and short indicators in training periods. For each iteration during the second-phase searching, we record the average of all rules, top 50, top 20, top 5, as well as the best rule for the new population it evolved. Figure 5 summarises the path of these average fitness improvement via second-phase GP. The starting point in the graph for each line is the first-stage average fitness. Through second-phase iterations, we observe clear improvement in average fitness for top rules in each population. However, the iteration of GP on the total of 10,000 rules are telling a different story. Given the starting value is the first-stage average fitness, the first iteration of second-phase evolution drop average fitness of 10,000 rules substantially. It is because that all 10,000 considerably good match-ups are messed up and recombined. Later, the average fitness lines grow up slowly, but they still do not exceed their initial fitness level. Overall, although the second phase does not improve all rules on average, it indeed can find better match-ups of buy and sell indicators that can perform better in the training period.

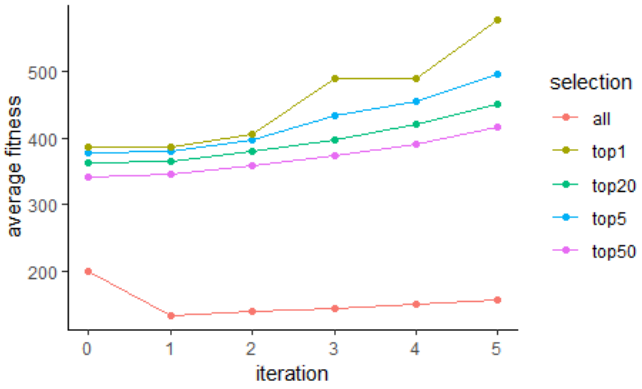



Figure 5: Second phase fitness improvement via GP iteration

## 4.2 Validation and test period performance

In this section, we first present the performance of training period searched rules in the validation period, then track the performance of those passed the filter in the test period. Table 3 presents the average performance (i.e. close balance) of rules found in both phases. For those top ones identified in the training period, we are interested in their generalisation in the validation period. We filter all training period discovered rules in the validation period to drop over-fitting rules. We report the average performance for all rules and those identified as top ones in the validation. We observe that all the first- and second-phase searched rules generate closely $99 close balance. We find that, on average, all top 50, top 20, top 5 rules as well as top 1 rule identified in the second-phase searched rules generate positive returns and out-perform those top ones identified in the first phase.

| Close balance | training period | | validation period | | validation period | |
|---|---|---|---|---|---|---|
| (positive ratio) | 1994-2007 | | 2008-2013 | | 2014-2017 | |
| rules: | 1st phase | 2nd phase | 1st phase | 2nd phase | 1st phase | 2nd phase |
| total num | 10,000 | 10,000 | 10,000 | 10,000 | 4610 | 4129 |
| total avg | 199.7 | 157.5 | 99.3 | 99.9 | 98.6 | 101.2 |
| top 50 | 342.3 | 415.4 | 105.0 | 113.7 | 98.0 | 100.7 |
| top 20 | 362.4 | 449.8 | 103.8 | 113.7 | 98.2 | 101.7 |
| top 5 | 379.0 | 495.9 | 99.9 | 101.2 | 104.0 | 106.5 |
| top 1 | 386.1 | 577.5 | 101.2 | 111.87 | 107.5 | 117.4 |

Table 3: Summary of average close balance (opening balance $100)

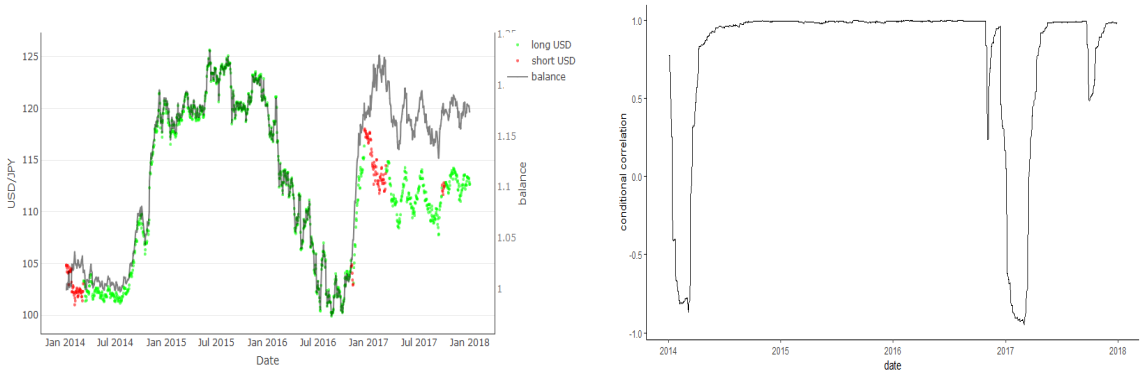
In a practical trading rule searching process, training and validation periods help investors to search and filter for profitable rules. Intuitively, investors tend to select those rules that have good generalisation for their implementation. Next, we present what rules will perform if they are implemented. This process aims to prove that GP searched rules can generate positive out-of-sample returns. We evaluate the performance of all filtered rules and identified top rules based on their validation period fitness in the test period. Rules are tracked for their performance in four years. On average, we see first-phase searched and filtered top 50, and top 20 rules do not generate positive returns on average in the test period, and second-phase ones although can generate positive returns, the magnitude is insignificant in the four-year horizon. For the top 5 rules, both first- and second-phase can generate positive returns in the test period on average. Interestingly the best one from second-phase searched rules continues

to dominate the best one from the first-phase, with close balance $117.4 compares to $107.5.

Therefore, through direct fitness comparison between rules searched from two phases, it shows that second-phase searching can explore rules that perform better on average than the rules searched in the conventional method. The best rule identified in the second phase can also largely out-perform the best rule from the first phase.

### 4.3 Test period performance interpretation

In section 4.2, we find positive returns from GP searched rules, especially for those from the second phase. Next, we investigate whether the positive return is due to a rule's active switching in the long and short position for profit-making or is due to the passive buy-and-hold strategy.




(a) Best second-phase searched and filtered rule

(b) Conditional correlation with buy-and-hold strategy

Figure 6

Figure 6a is the plot of test period performance of second-phase top 1 rule identified based on its validation period fitness. The solid line is the fitness of the rule over time. The coloured dot line is the price, where the colour indicates the rule's decision on each trading day as green for long and red for short. Figure 6b shows the rule's conditional correlation with buy-and-hold strategy.[5] Longtime 1-to-1 correlation means that the rule, instead of actively searching for arbitrage opportunities from market fluctuation, it simply mimics the buy-and-hold strategy. We also investigate the rest identified top 5 rules from the second phase. The rules in Figure 9g and 9a also follows the buy-and-hold strategy. Rules in Figure 9c and 9e are active in position switching, the former is profit-making, but the later generates loss. We see that the former one in Figure 9c can predict the market upward and downward trend to some degree, as its conditional correlation with the market return is actively changing during a clear trend

[5]The conditional correlation is obtained by modelling Dynamic Conditional Correlation GARCH(1,1) on return series for the rule and the buy-and-hold strategy.

period. However, although the rule is actively capturing the trend, its wrong decision making in some periods made it only slightly out-stands to the rule that mimics the buy-and-hold strategy.

# 5 Conclusion and discussion

For this report, we replicate a GP system proposed by Neely et al. (1997) and Allen & Karjalainen (1999) with modifications and extensions using R. In the first phase searching, 10,000 independent candidate trading rules are discovered based on combinations of functions that we pre-defined. Next, these 10,000 rules are inserted into second-phase GP for exploring better match-ups of long and short indicators. After rule searching in the training period, we drop out those overfitting rules based their performance in the validation period. Last we evaluate the performance of those remain in the test period.

We find that, first, the proposed second-phase searching methodology can discover better match-ups of long and short indicators using training period data. We also find second-phase searched, and filtered rules do perform better on average than those first-phase rules in the test period. Second, we find that GP, at least for the second-phase searched and filtered top rules, can discover profitable trading rules. However, by investigating the conditional correlation of return of top rules with market returns using DCC-GARCH, we found that a rule's profitability maybe not always because of its predictability. Instead, it is due to a rule's imitation of the buy-and-hold strategy.

This report does not prove that hyper-parameters and train, validation and test period separation implemented for GP are the optimal ones. Therefore, further experiments are required. It is also worthwhile to apply this two-phase searching methodology to another forex market to validate its effectiveness.

# References


Allen, F. & Karjalainen, R. (1999), 'Using genetic algorithms to find technical trading rules', *Journal of Financial Economics* **51**, 245–271.

Brabazon, A., O'neill, M. & Dempsey, I. (2008), 'An introduction to evolutionary computation in finance', *IEEE Computational Intelligence Magazine* **3**(4), 42–55.

Dempster, M. A. H. & Jones, C. M. (2001), 'A real-time adaptive trading system using genetic programming', *Quantitative Finance* **1**, 397–413.

Iskrich, D. & Grigoriev, D. (2017), Generating long-term trading system rules using a genetic algorithm based on analyzing historical data, *in* '2017 20th Conference of Open Innovations Association (FRUCT)', pp. 91–97.

Kaboudan, M. A. (1999), 'A measure of time series' predictability using genetic programming applied to stock returns', *Journal of Forecasting* **18**(5), 345–357.

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

Matsui, K. & Sato, H. (2009), A comparison of genotype representations to acquire stock trading strategy using genetic algorithms, *in* '2009 International Conference on Adaptive and Intelligent Systems', pp. 129–134.

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**, 405–426.

Poli, R., Langdon, W. B. & McPhee, N. F. (2008), *A Field Guide to Genetic Programming*.

Straßburga, J., ChristianGonzàlez-Martelb, C. & Alexandrovc, V. (2012), Parallel genetic algorithms for stock market trading rules, *in* 'International Conference on Computational Science, ICCS 2012', pp. 1306–1313.

Thinyane, H. & Millin, J. (2011), 'An investigation into the use of intelligent systems for currency trading', *Computational Economics* **37**, 363–374.

Wang, J. (2000), 'Trading and hedging in s&p 500 spot and futures markets using genetic programming"', *Journal of Futures Markets* **20**(10), 911–942.

# 6 Appendix

## 6.1 Figures

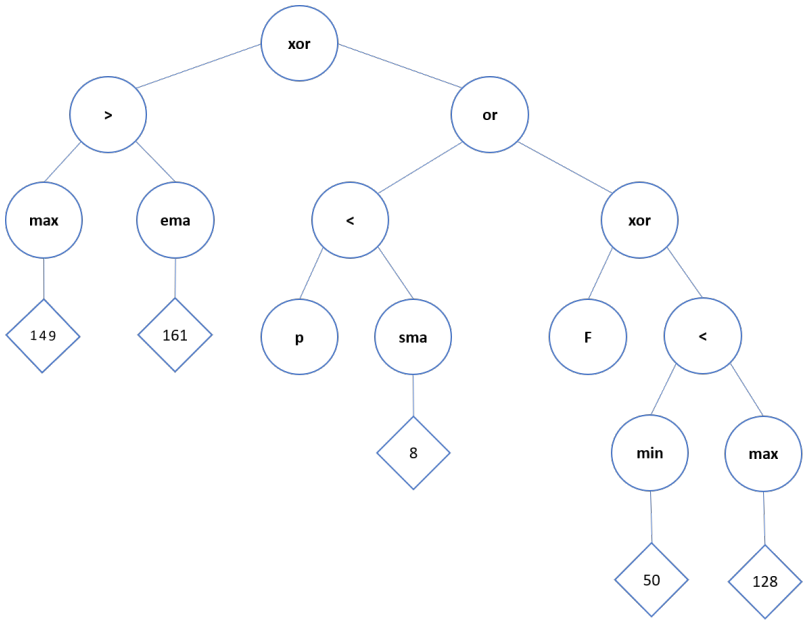



Figure 7: A complex tree structure (long indicator)

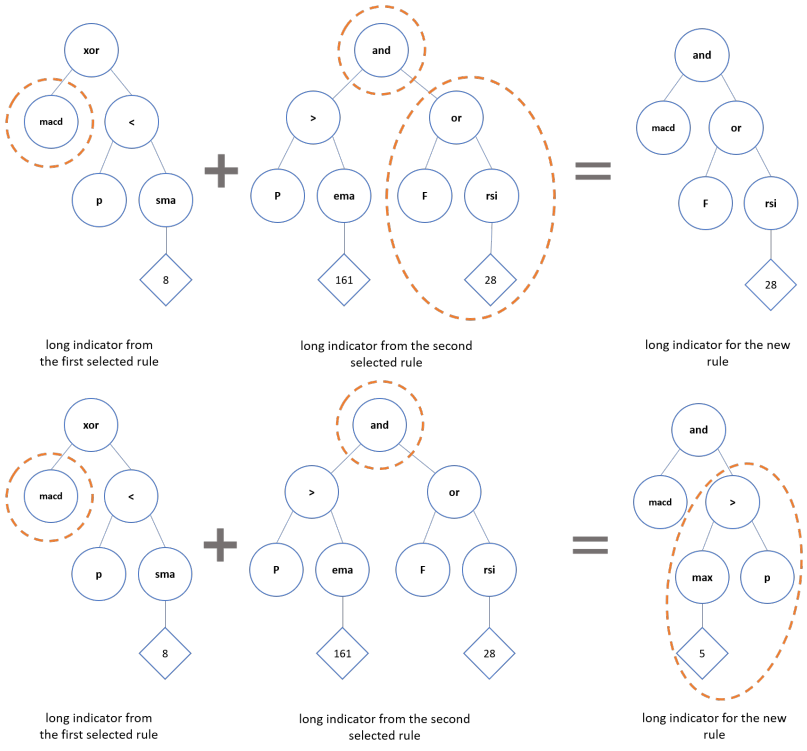
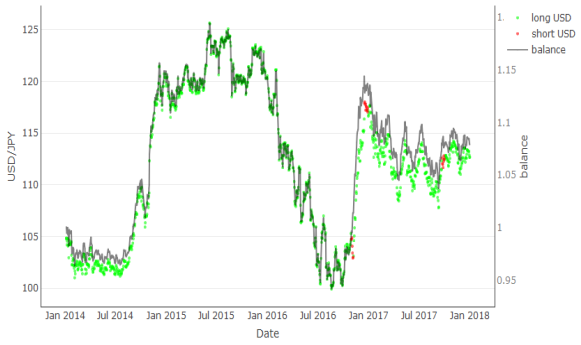
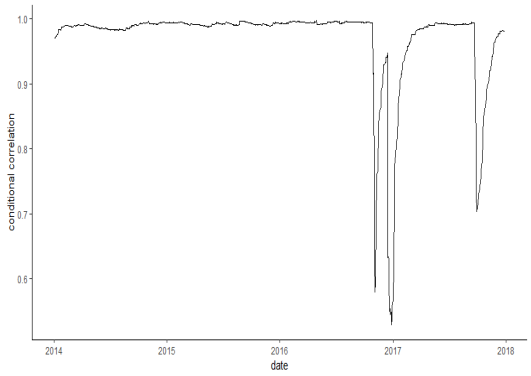


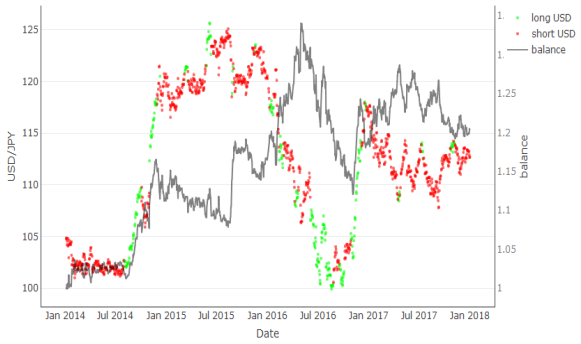

Figure 8: Crossover (up), Mutation (down)



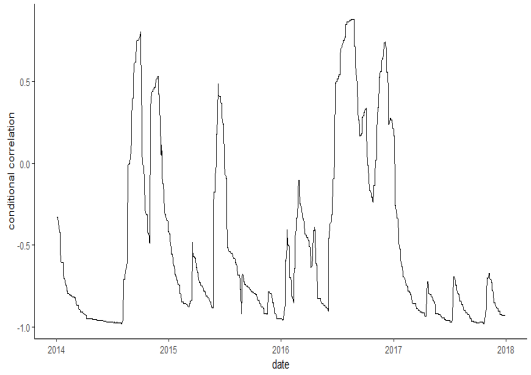

(a) top 5 (4)



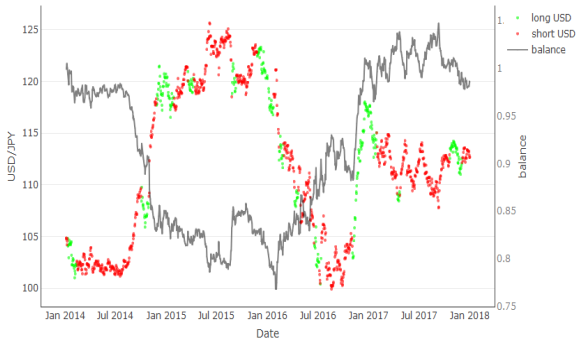

(b) Coniditional correlation



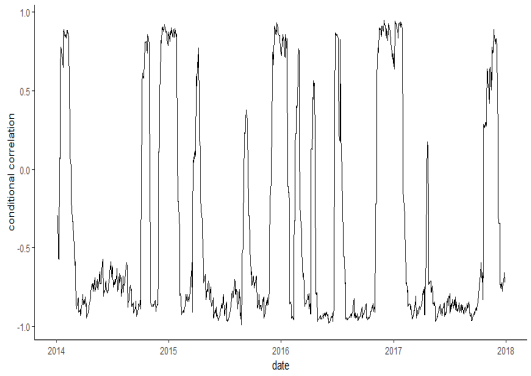

(c) top 5 (3)



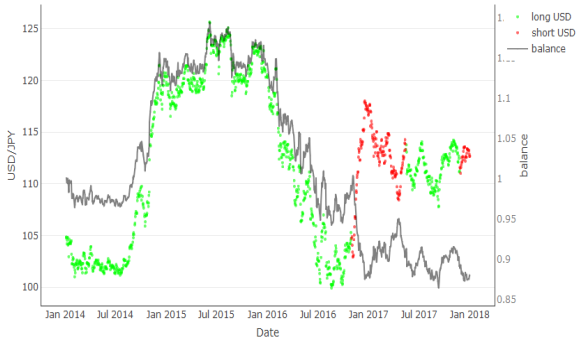

(d) Coniditional correlation



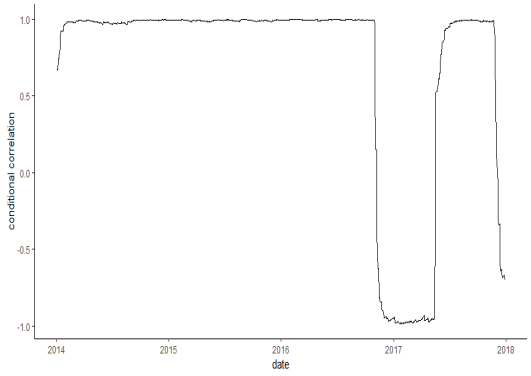

(e) top 5 (2)




(f) Coniditional correlation




(g) top 5 (1)




(h) Coniditional correlation

Figure 9: Second-phase Top 5 rules performance and Conditional correlation with buy-and-hold strategy

## 6.2 Tables

| function | input number | input type | param range | output type | notes on operation |
|---|---|---|---|---|---|
| AND | 2 | Boolean | - | Boolean | 1 if TRUE, 0 if FALSE |
| OR | 2 | Boolean | - | Boolean | 1 if TRUE, 0 if FALSE |
| XOR | 2 | Boolean | - | Boolean | 1 if TRUE, 0 if FALSE |
| TRUE | - | - | - | Boolean | always 1 |
| FALSE | - | - | - | Boolean | always 0 |
| $>$ | 2 | Numeric values | - | Boolean | 1 if TRUE, 0 if FALSE |
| $<$ | 2 | Numeric values | - | Boolean | 1 if TRUE, 0 if FALSE |
| MAX | 1 | Numeric series | $n = (1, 150)$ | Numeric | Maximum in $n$ periods |
| MIN | 1 | Numeric series | $n = (1, 150)$ | Numeric | Minimum in $n$ periods |
| SMA | 1 | Numeric series | $n = (1, 150)$ | Numeric | Simple moving average in $n$ periods |
| EMA | 1 | Numeric | $n = (1, 150)$ | Numeric | Exponential moving average in $n$ periods |
| LAG | 1 | Numeric | $n = (1, 150)$ | Numeric | Lag in $n$ periods |
| MACD | 1 | Close price | (26,12,9) | Boolean | 1 if MACD>signal, 0 otherwise for buy |
| | | | | | 1 if MACD<signal, 0 otherwise for sell |
| RSI | 1 | Close price | $l = (20, 40)$,$h = (60, 80)$ | Boolean | 1 if RSI<l, 0 otherwise for buy |
| | | | | | 1 if RSI>h, 0 otherwise for buy |
| STO | 1 | Close price | $l = (20, 40)$,$h = (60, 80)$ | Boolean | 1 if STO>signal, 0 otherwise for buy |
| | | | | | 1 if STO<signal, 0 otherwise for sell |
| Close | - | - | - | Numeric series | Close price |
| High | - | - | - | Numeric series | Price High |
| Low | - | - | - | Numeric series | Price Low |

Table 4: Definition of nodes

### 6.3 Function explanation

Functions can be categorised into six sets based on their properties. The first set is $\{\text{AND}, \text{OR}, \text{XOR}\}$. Functions take Boolean inputs and return Boolean outputs. These functions are essential for level 1 node, but can also appear in other levels. [6] The second set is $\{\text{TRUE}, \text{FALSE}\}$, which takes no inputs and always produces true and false respectively. The third set includes functions $>$ and $<$ for numeric value comparison, which takes two numeric inputs and returns a Boolean output. The fourth set is $\{\text{MAX}, \text{MIN}, \text{SMA}, \text{EMA}, \text{LAG}\}$, each has an associated parameter $n$ as an integer randomly selected from range $(2, 150)$. These functions take a time series of numeric values. MAX function outputs the maximum of input series within the $n$ periods and MIN function finds the minimum in a similar manner. SMA and EMA output the simple and exponential moving average of a series in the past $n$ periods. LAG function outputs the lag $n$ value for the input.

The fifth set consists of popular technical trading indicators, the Relative Strength Index (RSI), the Moving Average Convergence and Divergence (MACD) and the Stochastic Oscillator (STO). Traders use RSI's lower ($l$) and upper ($h$) bound to determine market oversold and overbought. For RSI node working in a long indicator, if RSI is lower than lower bound $l$, the function $RSI$ outputs a True Boolean. It works oppositely if RSI appears in a short indicator. For MACD and STO, traders use crossovers of moving average and signal lines to indicate trading decisions. If these functions appear in long indicator and when the moving average line rises above its signal line, they indicate a buy signal, which is a TRUE Boolean. They act oppositely if it appears in a short indicator. The Boolean outputs are then inserted into connected functions as inputs. Although MACD and STO have parameters that can adjust, pre-defined parameters are used since they are widely accepted benchmark parameters.

Different from existing literature that only uses close price as the data source. We use three price data: the close, the high and the low of the day. The sixth function set $\{\text{CLOSE}, \text{HIGH}, \text{LOW}\}$ take no input and produce the price data correspondingly. They are the main input stream for the third and fourth set, while the fifth set only uses the close price.

[6] XOR function, as exclusive or, returns True if two Boolean inputs are not the same, and return False if inputs are identical.