Genetically Evolved Trading Strategies with PonyGE2

Eoin Carroll

16202781

Abstract

The finance industry is often quick to adopt new technology and the application of natural computing techniques is a contemporary research area. This project aimed to derive and test genetically evolved trading strategies with a grammatical evolution algorithm. Four S&P 500 stocks were modelled over a 17-year period and trading strategies were generated. The results were marginally more profitable than a benchmark buy and hold strategy.

1 Introduction

The financial markets have been transformed with the use of technology from online trading platforms to comprehensive analytics tools. This paper reports a project involving the application of advances in machine learning to an interesting finance problem. The introduction will introduce natural computing, evolutionary computation, genetic programming (GP) and trading strategies along with covering previous applications of GP to trading, motivation and objectives. A methodology section describes the approach taken, an experimental section captures experimentation and the paper is completed with a discussion and conclusion.

1.1 Natural Computing

Natural computing is a branch of computer science inspired by nature. One area of natural computing involves utilising materials other than silicone to carry out computational tasks and a great example of this is in biological computing where

DNA has been used to successfully store information. Another area of natural computing is in the machine learning domain where algorithms are derived from key characteristics of processes found in nature and are deployed to solve difficult problems (Brabazon & O'Neill, 2009). Examples of natural computing algorithms include exploration of shortest paths using ant colony optimisation or game strategy development using artificial neural networks where components of a human brain are mimicked in software. The algorithm implemented in this project was inspired by natural evolution.

1.2 Evolutionary Computation

The concept of machine learning dates back as far as Turing and has been the subject of sci-fi since the beginning of the digital revolution (Turing, 1950). Friedberg (1958) hypothesised that computers could perform new tasks if the computer could create a new program through trial and error. The Darwinian evolution theory is the evolution of species through slight variations in each generation where traits that strengthen the species are passed on with a random but probabilistic degree. One means of facilitating a trial and error methodology in computation is through evolutionary computation where an algorithm is designed to generate and test numerous candidate solutions to a problem. This algorithm is a metaphoric implementation of evolution and the trial and error simulation mimics the survival of the fittest aspect of nature (Beyer, 2002).

1.3 Genetic Programming

Genetic programming (GP) works by randomly creating an initial population of programs from a defined set of programming features. This random process means it never guarantees convergence on an optimal solution however it does introduce the benefit of producing unexpected programs that a human may not have considered an option. For each generation of population, parent programs pass on traits to children programs stochastically through two main mechanisms (PolR et al. 2008):

- Crossover: Randomly selected traits from two parents are preserved.
- Mutation: A parent is randomly changed.

This process continues until a convergence criteria or iteration limit has been reached. The final output program is the program producing the best fitness result, where the fitness criteria is defined for that specific problem. A graphic showing the information flow in genetic programming is shown in Figure 1 below.

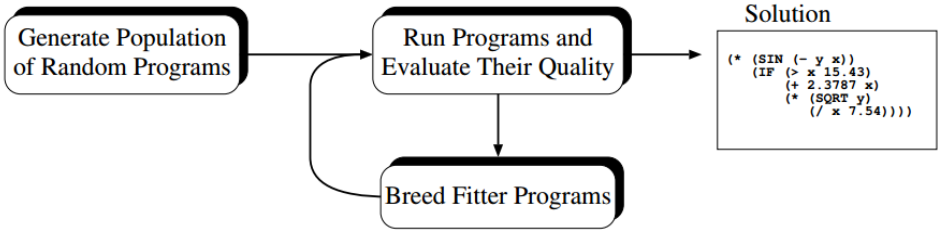

Figure 1 - GP Control Flow (PolR et al. 2008)

1.4 Trading Strategies

In financial markets, a trading strategy is an execution plan of when to invest in or sell an asset. Most strategies aim to maximise return on an initial investment while minimising variation in the expected range of return. Strategies vary greatly depending on the asset class and time frame. A key performance metric of a trading strategy is a comparison of return on investment in contrast with that of the return from simply buying and holding the asset instead.

Financial markets are perceived as non-deterministic and it has been demonstrated that trading strategies often perform extremely poorly in the real world. This poor performance has been attributed to the noisy, constantly changing arena that is the financial markets, however this environment could be well suited to natural computing (Brabazon & O'Neill, 2009).

A moving average crossover is when a stock price rises above or falls below a certain moving average which can signify a change in trend. This historic indicator is a very simple method of determining the direction a stock price may move and is still used on trading floors today although will not necessarily forecast future prices (Ready, 2002).

1.5 Application of Genetic Evolution to Trading

There have been many examples of genetically evolved trading strategies published and most appear to report inconclusive results and recommend further research. Allen & Karjalainen (1999) implemented a genetic algorithm to find technical trading rules but the authors were quick to point out that numerous papers have shown that technical trading rules are unreliable and will not guarantee profits. They go on to say that it is possible that historic technical rules may not be optimal but that genetic algorithms present an opportunity to derive optimal solutions. The authors applied their genetic algorithm to the S&P

500 composite index and derived some interesting trading rules. While giving informative insights into the data, the rules did not earn additional returns in comparison to buying and holding. This paper also brought together previous work that had not been formally analysed up to this point.

O'Neill et al. (2001) presented promising results by deriving trading rules with Grammatical Evolution which is a method of genetic evolutionary programming. Their work was focused on the UK FTSE 100 stock index in contrast to most other papers concerned with the US stock market.

Sceptics of genetic programming induced decision making, Chen & Navet (2007) question the usefulness of genetic programming due to the inconclusive results that seem prominent in publications. They propose a method of testing the market and derived strategies to help conclude further research and determine if additional research in GP is worthwhile.

1.6 Motivation and Objective

The objective of this project is to generate a trading strategy using genetic evolution using the PonyGE2 python library to build on previous works and validate their results. Specifically, grammatical evolution like O'Neill et al. (2001) will be implemented on individual US stocks. A successful outcome of the project would be a genetically evolved trading strategy that provides a return on both training and test data that exceeds a return from a simple buy and hold technique. A successful strategy would need to determine technical trading rules such as the moving average crossover but furthermore, that technical trading rule needs to be profitable in the test dataset and this is the fundamental theory behind genetically evolved trading strategies.

Evolutionary algorithms have shown promise in past research and have the benefit of human readable output in comparison to neural networks. This is attractive for investors who may not trust a hidden trading strategy. The project will use three basic stock indicators, the historic daily open, high and low prices. Since the input data is purely technical in nature, the output strategy will be a set of technical rules. While it is unlikely that it will uncover ground-breaking indicators from the project, any interesting correlations would be significant in the fields of natural computing and trading strategies.

2 Methodology

The methodology selected to generate trading strategies is loosely based on O'Neill et al. (2001) with some structural simplifications, a wider search space and different datasets. The algorithm outputs will be trading rules which will be implemented on test data with the outcomes benchmarked against a buy and hold strategy.

2.1 PonyGE2

PonyGE2 is the second version of PonyGE which was developed as an easy to implement genetic algorithm with the use of grammatical evolution (GE). GE is a concept from molecular biology where a grammar is defined to store the possible components of candidate programs in a text file format called Backus-Naur Form (.BNF). A genotype is a randomly generated list containing integer numbers which are mapped to the grammar and each number will select a component from the grammar options. The result of this mapping is referred to as a phenotype which is a candidate solution. In this way, many different grammars can be easily testing using PonyGE2 (Fenton et al. 2017).

2.2 Simplifications

In contrast to O'Neill et al. (2001), this project will focus on one share at a time instead of an index. The fitness function has been simplified to look purely at profits over the training time period and the data will not be normalised so that the output is relatable to current market prices. The investment strategy will be to invest all or sell all at any point instead of holding and smaller orders. Additionally, there is no cost of placing a trade included, the program can buy partial shares and the returns omit dividends.

2.3 Search Space

O'Neill et al. (2001) note that their project focussed mainly on moving average crossover indicators and so it can be assumed that their grammar was strongly typed to promote this as a trading rule. The grammar created for this project is purposefully left more open to allow for alternative technical rule generation.

3 Experimentation

3.1 Dataset

The dataset for experimenetation was obtained from the Bloomberg suite where the open, high and low stock price for all S&P 500 companies on each trading day for the years 2000 to 2017 was downloaded. This data was then parsed into individual .csv files for analyses. A preprocessing step removed a small number of trading days with missing data points. Four stocks were selected across a range of industries – Apple (AAPL), Boeing (BA), Bank of America (BAC) and Coca Cola (KO).

3.2 Grammar

The grammar was created to generate a series of if statements which represent trading rules. Firstly, it allows for moving average calculation and single point comparisons across any data points with a bias towards yesterday's prices. Secondly, it allows for any of combination of data points and averages to be evaluated which opens the search space for any technical indicator. The grammar will generate a positive or negative number which indicates a recommendation to buy or sell. A sample of the grammar is shown below and the full grammar can be found at the at the Github link in the appendix. The vertical bar indicates a selection which is made by the genome.

```
<if> ::= if <data> <eval> <data>:{:x = <expr>:}
<data> ::= <moving_average> | <data_point>
<eval> ::= > | <
<expr> ::= <const> | (x + <const>)
<const> ::= 0.1 | 0.2 | 0.3
```

3.3 Fitness Function

The fitness function is a program that executes the phenotype every trading day for a given date range. Specifically, the training fitness is executed from 2000 to 2016 while the test fitness is executed from 2016 to 2017. Note that only the last year of data is passed into the fitness function and so the trading rules will only consider this amount of information. This is important to stop irrelevant historic or unavailable future data being included. Pseudo code for the fitness function is shown below and the full Python implementation can be found at the Github link in the appendix.

```
Cash, shares = $10,000, 0
for i in range(start, end):
```

```
data = df[start-246: end-246] # Today – 246 trading days
execute(phenotype)
if recommendation > 0:
       invest_all_cash()
else:
       sell_all_shares()
sell_all_shares()
return cash
```

The output of this program is cash remaining after executing the candidate program each day over the time period.

3.4 PonyGE2 Parameters

The parameters within PonyGE2 were trialled and the most promising setup was selected for the experiment. The main parameters used were a population of 200 candidates for 30 generations and a crossover probability of 75%. A full set of parameters used can be found at the Github link in the appendix.

3.5 Experiment Results

The trading rules were generated and the strategy was implemented over both training and test periods. A comparison of the returns from a simple buy and hold strategy in comparison to the derived trading rules over the training and test periods is shown in Table 1.

	Stock	**AAPL**	**BA**	**BAC**	**KO**
Train period: 2000/2016	Investment	$10,000	$10,000	$10,000	$10,000
	Buy&Hold	$563,324	$20,788	$5,257	$18,842
	GE Rules	$2,334,625	$80,493	$59,980	$36,617
Test period: 2016/2017	Investment	$10,000	$10,000	$10,000	$10,000
	Buy&Hold	$15,488	$14,118	$15,978	$10,007
	GE Rules	$15,488	$13,091	$17,152	$11,121

Table 1 – Training and testing period returns comparison

The fitness function output is the cash after each period so the fitness per generation can be examined in context. The plots in Figure 2 show the train fitness improving over time. More generations may have improved results.

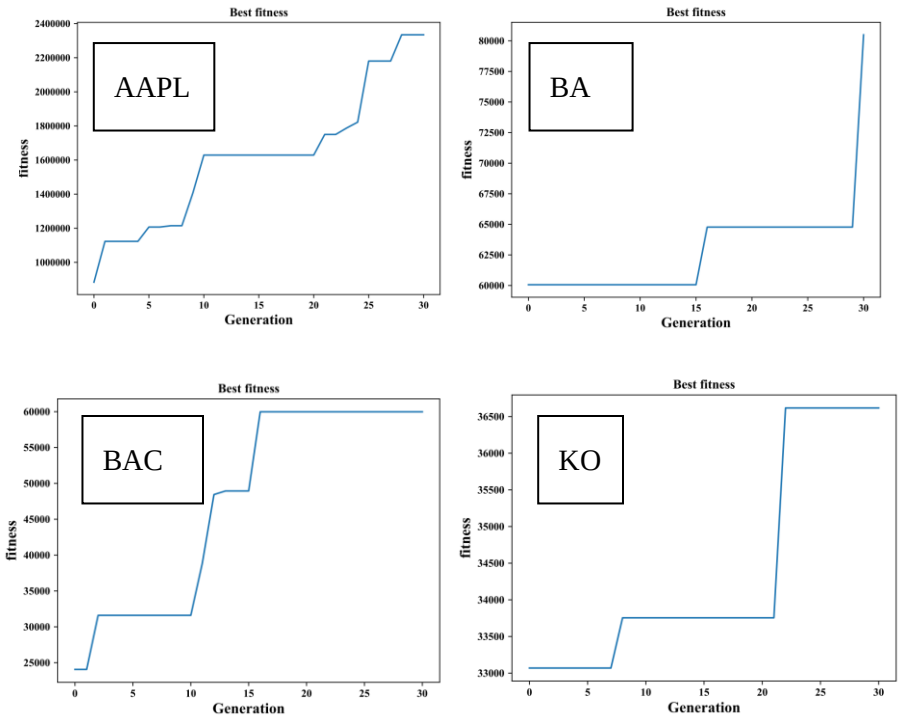

Figure 2 - Fitness results per generation of strategy

All four trading strategies produced interesting programs and the full results can be found at the Github link in the appendix. Some redundant or meaningless outputs were visible and unlikely to provide insights, however below is one of the more useful rules.

```
If yesterday's high > 6 month moving average high
   increase_buy_signal()
```

4 Discussion

The genetic algorithm fit a strategy to work well with that specific stocks training data. When that strategy was applied to the same stocks test data, mixed results were found. There was no change to AAPL, BA saw a loss while BAC and KO showed an increase in return. Overall additional profits of $1,261 were found over the 1 year test period from $10,00 investment compared to a buy and hold benchmark.

The selection of stocks for this project was somewhat arbitrary and is not representative of the market as a whole. In general, the stocks that with most data available are stocks that have performed well historically and so care should be taken when generalising any conclusions. Likewise, the strategies were derived for the four sample stocks and are unlikely to generalise well.

Positive attributes of genetic programming for trading strategies include the strategy output in the form of a string which can be easily analysed, novel strategy generation that a trader may not have come up with and lastly, promising experimentation results in this project and previous literature.

Some limitations and drawbacks identified include susceptibility to overfitting, lack of guarantee of convergence on a useful solution, sensitivity to modelling causing very different results and a limitation of focus on technical indicators.

The project was created with scalability in mind and so any additional data points can be added to the fitness function for inclusion in strategy generation. Further study could include more quantitative data such as price to earnings ratios and a more detailed comparison of results with different parameters would also be interesting to see. Lastly, programming improvements within the fitness function could be implemented to account for some of the simplifications made in this implementation such as omitting variance and risk as minimisation objectives.

5 Conclusion

The objective of this project was to generate a trading strategy using genetic evolution. To this end, the program can create trading strategies. A success measure was defined as having the genetic algorithm output moving average as a technical indicator and the strategies did incorporate this into the trading rules.

Overall, the trading strategies generated performed extremely well on the training data and with mixed results on the test data which is a classic case of overfitting. The feasibility of using genetically evolved trading strategies remains inconclusive.

6 References

Allen, F. & Karjalainen, R. 1999, "Using genetic algorithms to find technical trading rules", Journal of Financial Economics, vol. 51, no. 2, pp. 245-271.

Beyer, H. & Schwefel, H. 2002, "Evolution strategies – A comprehensive introduction", Natural Computing, vol. 1, no. 1, pp. 3-52.

Brabazon, A. & O'Neill, M. 2009, Natural computing in computational finance, Springer-Verlag Berlin and Heidelberg GmbH & Co. K, Berlin.

Chen, S.H. and Navet, N., 2007. Failure of genetic-programming induced trading strategies: Distinguishing between efficient markets and inefficient algorithms. Computational Intelligence in Economics and Finance, 2, 169-182.

Fenton, M., McDermott, J., Fagan, D., Forstenlechner, S., O'Neill, M. & Hemberg, E. 2017, "PonyGE2: Grammatical Evolution in Python", .

Friedberg, R.M., 1958. A learning machine: Part I. IBM Journal of Research and Development, 2(1), pp.2-13.

O'neill, M., Brabazon, A., Ryan, C. & Collins, J.J. 2001, "Evolving market index trading rules using grammatical evolution", Applications of Evolutionary Computing. EvoWorkshops 2001. Lecture Notes in Computer Science, vol 2037. Springer, Berlin, Heidelberg, pp. 343.

PolR, Langdon W B, McPhee N F (2008) A Field Guide to Genetic Programming. http://www.gp-field-guide.org.uk

Ready, M.J. 2002, "Profits from Technical Trading Rules", Financial Management, vol. 31, no. 3, pp. 43-61

Turing, A.M. 1950, "Computing Machinery and Intelligence", Mind, vol. 59, no. 236, pp. 433-460.

7 Appendix

All code can be found at the following Github repository: https://github.com/eoincUCD/NC_GE_Evolved_Trading_Strategy