Genetically Evolved Trading Strategies with PonyGE2

# Abstract

Contents

[Abstract 1](#_Toc486372542)

[Introduction 2](#_Toc486372543)

[Natural Computing 2](#_Toc486372544)

[Evolutionary Computation 2](#_Toc486372545)

[Genetic Programming 2](#_Toc486372546)

[Trading Strategies 3](#_Toc486372547)

[Application of Genetic Evolution to Trading 4](#_Toc486372548)

[Motivation and Objective 4](#_Toc486372549)

[Methodology 5](#_Toc486372550)

[Problem Description 5](#_Toc486372551)

[PonyGE2 5](#_Toc486372552)

[Assumptions and Simplifications 5](#_Toc486372553)

[Experimentation 6](#_Toc486372554)

[Dataset 6](#_Toc486372555)

[Grammar 6](#_Toc486372556)

[Fitness Function 6](#_Toc486372557)

[Experiment Results 6](#_Toc486372558)

[Discussion 6](#_Toc486372559)

[Conclusion 6](#_Toc486372560)

[References 8](#_Toc486372561)

[Appendix 8](#_Toc486372562)

# Introduction

Since the 1990s, the financial markets have been transformed with the use of technology from online trading platforms to comprehensive analytics tools. This paper captures the results of a project involving the application of recent advances in machine learning to an interesting finance problem. The first few sections of this introduction will bring a computer software student or IT professional who has not researched natural computing up to a base knowledge level where they will understand the project objective and outcome. The introduction section covers natural computing, evolutionary computation, genetic programming (GP), trading strategies, previous application of GP to trading, motivation and objectives. The experimental section . . . discussion and closing remarks. For those initiated in the natural computing area, they may skip forward to the Trading Strategies section within the introduction.

## 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 (Church et al. 2012). 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). The application of a computing algorithm inspired by a nature forms the basis of this paper while the Natural Computing journal provides further reading.

Examples of natural computing algorithms include exploration of shortest paths using swarm techniques such as ant colony optimisation or finding optimal points by modelling bird flocking behaviour (Banks et al. 2007). Another popular research area is artificial neural networks where the components and workings of a human brain are mimicked with software to artificially learn the solution to a range of problems such as developing game strategies. The algorithm implemented in this project was inspired by the process of natural evolution and a brief overview is presented in the next few sections.

## Evolutionary Computation

The concept of machine learning dates back as far as Turing and has been the subject of many sci-fi works 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 widely accepted 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 programming 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).

## Genetic Programming

Genetic programming (GP) works by randomly creating an initial population of programs from a defined set of programming features and constraints such as code sections, inputs, outputs and transformations. This random process means we are never guaranteed to converge on an optimal solution however it does introduce the benefit of producing unexpected programs that a human may not have considered as an option. For each generation of population, parent programs pass on traits to children program stochastically through two main mechanisms (PolR et al. 2008):

* Crossover: Randomly selected traits from two parents are preserved.
* Mutation: A parent is randomly changed to produce a child.

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

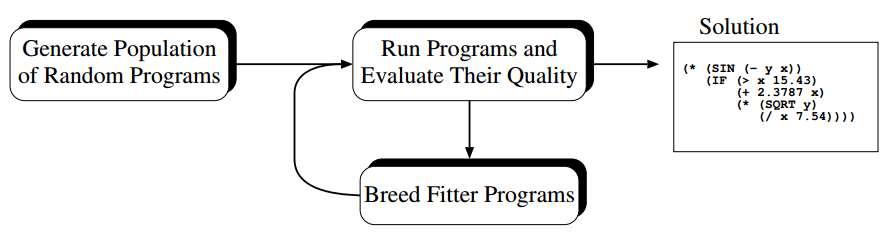


Figure - GP Control Flow (PolR et al. 2008)

As Koza (2009) notes, the usefulness of genetically evolved programs has been proportional to improvements in computing power due to the computation required in this domain. Genetic programming has successfully produced software that rivals that produced by humans in a wide variety of fields such as software creation, game theory where strategies have been developed, electronic design where digital and analog circuits have been designed and mechanical engineering for systems design (Koza, 2009).

Lastly and most importantly for this project, the finance industry has employed genetic programming in many specific problem areas. Some examples of evolutionary computation in finance include asset allocation in pension schemes, strategies for risk-optimal portfolios and stochastic portfolio optimisation (Brabazon & O'Neill, 2009). There are also many examples of notable trading strategies being developed with genetic algorithms. Allen & Karjalainen (1999) used S&P 500 stock price data and a genetic algorithm to generate trading rules which closely matches the aims of this project.

## 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. Time frame classification can vary but for this project, short time frame is less than three-month period while a long time frame is anything greater than 12 months. A medium-term strategy falls between the two categories and is the focus of this project. Furthermore, the almost limitless asset classes have been restricted to stocks for this project. A key performance metric of a trading strategy is a comparison of the strategies return on investment with that of the return if we have simply bought and held the asset instead.

Financial markets are perceived as non-deterministic and Conrad & Kaul (1998) 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).

The efficient market hypothesis is a theory that states that no trading strategy will return excess profits in a well-developed market as the stocks will always be priced correctly. If this theory holds then no evolutionary algorithm would be able to derive a strategy based on market inefficiency.

One final concept that the reader should be aware of is a moving average crossover strategy. If a stock price rises above or falls below a certain moving average, this signifies 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 predict the future (Ready, 2002).

## Application of Genetic Evolution to Trading

There have been many examples of GE trading strategies published over the years. Most appear to report inconclusive results and recommend further research while very few present definitive successful or unsuccessful strategies. Other related works have been performed with the aim of optimising a portfolio or predicting stock prices however these are outside the scope of this project.

As mentioned, 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 additional profits. They go on to say that it is possible that historic technical rules may not be optimal but that genetic algorithms may 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.

There are countless instances of claimed successful GP trading strategies but we should analyse their results with scepticism. 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.

Further additions to GP strategies are the use of graph and network structure to improve results (Izumi et al. 2006) or the use of Sarsa learning (Chen et al. 2007). Critics 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 program to help conclude further research and determine if additional research in GP is worthwhile. As we have seen from past works, genetic evolution can derive very interesting and novel strategies due to its random nature but it’s usefulness is questionable.

## Motivation and Objective

The objective of this works 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 positive 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.

PonyGE2 is a Python implementation of a grammar based evolutionary algorithm. Evolutionary algorithm has shown promise in past research and has the benefit of human readable output in comparison to the likes of 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 a we will uncover ground-breaking indicators from the project, any interesting correlations would be significant in the fields of natural computing and trading strategies.

# Methodology

## Problem Description

Overall similar to Oneil 2001 but simplified in x,y,z.

Allows more rules to be generated than moving average

Aim of experiments

Strategy will be a list of rules or almost decision tree. Most strategies appear to use genetic programming to induce a set of trading rules. This approach will be applied in this project.

## PonyGE2

How GE works: Fitness, codon, genome, grammar etc mutation, crossover (match in experiment section)

Particular algorithm – positives, negatives and limitations of this one

Specific GE algorithm that Pony uses?

Grammatical evolution , , , , highlighted in O’Neill et al. (2001)

## Assumptions and Simplifications

We only look back one year of data

Cover the point that model does not consider global/outside indicators and influencers

Single objective – max profit ignoring variance

Technical indicators only (ie stock price etc. and not corporate and political news)

Shut down strategy omitted

One stock, 3 prices per day

Ignore dividends and stock splits already accounted for

No cost of placing a trade

Not including variance but this could be included

my trades do not influence market

We can buy partial shares

Trade on opening price everyday

Removed NaN cells from dataset

DONE TO HERE

DONE TO HERE

DONE TO HERE

DONE TO HERE

DONE TO HERE

DONE TO HERE

DONE TO HERE

# Experimentation

## Grammar

Strongly typed for faster convergence

Allows moving average and so moving average crossover system could be used. N is set

N also compares like with like

Is set up to create trees that could offset eachother

Set up to ensure we cannot see into the future

Setup and results

Can use any input dataset, can add on other variables such as high/low/recommendations etc.

# Discussion

Analyses of results

Crossover – might totally ruin good parts by combining . . . “Deriving Trading Rules Using Gene Expression Programming” - Overcomes gene crossover issue

Difficult to lose money with AAPL

Debate over speculative trading being detrimental to markets but it props up and keeps commodities constant with futures etc.

efficient market hypothesis

Positives - strategy is in the form of a program string and so can be analysed, Novel strategies that a trader may not have come up with, We can see the output in if statements etc

Limitations/Drawbacks – Overfitting (minimise with train and test), time consuming (reduces with moores law), random so strategy could be completely different every time. Input data could be biased . . . we wouldn’t look at a failing or bust companies data. Limited to technical indicators.

Site run times as a constraint and reasoning for simplifying – would be interesting to increase generations.

# Conclusion

Restate objective

Further research – add other variables, add ability to buy/sell portions not all, add hold option, two objectives – minimise variance, punish complexity

Fitness function should look to give a good return on average and not just max for that dataset

Refer back to objective . . . was it achieved