Genetical Evolution in Trading Strategies

# Abstract

# Introduction

This paper aims to capture the output 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. 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. 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 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 a 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 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.

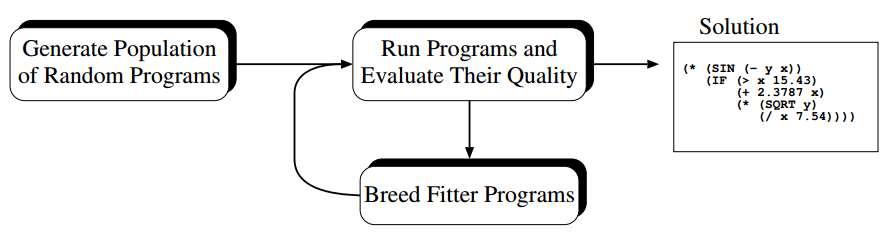


Figure 1 - GP Control Flow (PolR et al. 2008)

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 also 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 is the same theme as the objective 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. A short time frame is less than three-month period while a long time frame is anything greater than 12 months (Conrad & Kaul, 1998). 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 to return on investing and holding a particular asset instead of buying and selling that asset.

The derivation of a trading strategy is an interesting topic since financial markets are perceived as non-deterministic and Conrad & Kaul (1998) demonstrated that 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).

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.

## Application of Genetic Evolution to Trading

As mentioned, Allen & Karjalainen (1999) implemented a genetic failed to generate additional returns

Fabian Kostadinov – Article

Main benefits of GP is random unexpected programs which can lead to insights in different fields. Kostadinov points out that it is extremely unlikely that a we will uncover ground-breaking technical indicators from our datasets.

Failure of Genetic-Programming Induced Trading Strategies: Distinguishing between Efficient Markets and Inefficient Algorithms

More background – earliest mention?

Try find a successful one

Controversy – overfitting training data

Many inconclusive papers

Other NC attempts? Neural network will not show workings - intended to be a continually developing AI rather than a single use program. Intended as a continually evolving strategy that uses latest data rather than outputting a strategy to be analysed and implemented. Designed to be left alone - Argue that it can be left alone. Neural network selected as it may be more advanced . . .

## 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. An optimistic result from the project would be a genetically evolved version of the moving average crossover strategy however this seems unlikely based on former works in the area.

What algorithm and why? (Finance can be a noisy environment just like nature – MO book)

Question – could a simple PonyGE2 GE beat the stock market

Based on the theory that there is some trend/exploitable aspect

This has been tried before here and here . . . mine is the same/different in this way/etc

Continuing advances in computing power available to not just finance professionals but also consumers

What is the problem

Question being asked - would it be feasible

Objective – beat return on if you had bought one share and held for 1 years

An extension of the classic portfolio optimisation problem – without variance

Morgan Stanley using AI – becoming more and more popular

problem description, proposed variation

What role would a ge or NC alg play in bus an

Significance of the problem . . . and in bus an

Why that algorithm

Problem statement/objective

Method/Alg/Variation

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.

## PonyGE2

Particular algorithm – positives, negatives and limitations of this one

Specific GE algorithm that Pony uses?

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

## Paper Layout

# Experimentation

Strategy will be a list of rules or almost descision tree

Question being asked - would it be feasible

Just need to code enough to show promise

Assumptions is that S&P companies do not change over time (they are updated . . .)

Fitness function

Simple is better

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

Grammar

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.

Frist experiment . . . . random buy/sell with probability 50% . .. . . Cash after 16 years:

Final result, cash remaining after 10000: 49376.2923

Site run times as a constraint and reasoning for simplifying

## Assumptions and Simplifications

We only look back one year of data

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

We do not need to show workings to execute . . . some banks may not like this

Single objective – max profit ignoring variance

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

Shut down strategy ommited

## Problem Description

## Experiment Results

# Discussion

Analyses of results

Crossover – might totally ruin good parts by combining . . .

# 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

# References

1. Church, G.M., Gao, Y. & Kosuri, S. 2012, "Next-Generation Digital Information Storage in DNA", Science, vol. 337, no. 6102, pp. 1628.
2. Brabazon, A. & O'Neill, M. 2009, Natural computing in computational finance, Springer-Verlag Berlin and Heidelberg GmbH & Co. K, Berlin.
3. Banks, A., Vincent, J. & Anyakoha, C. 2007, "A review of particle swarm optimization. Part I: background and development", Natural Computing, vol. 6, no. 4, pp. 467-484.
4. Turing, A.M. 1950, "Computing Machinery and Intelligence", Mind, vol. 59, no. 236, pp. 433-460.
5. Friedberg, R.M., 1958. A learning machine: Part I. IBM Journal of Research and Development, 2(1), pp.2-13.
6. Beyer, H. & Schwefel, H. 2002, "Evolution strategies – A comprehensive introduction", Natural Computing, vol. 1, no. 1, pp. 3-52.
7. PolR, Langdon W B, McPhee N F (2008) A Field Guide to Genetic Programming. <http://www.gp-field-guide.org.uk>
8. Koza, J.R., 2010. Human-competitive results produced by genetic programming. Genetic Programming and Evolvable Machines, 11(3-4), pp.251-284.
9. Conrad, J. and Kaul, G., 1998. An anatomy of trading strategies. The Review of Financial Studies, 11(3), pp.489-519.
10. Allen, F. & Karjalainen, R. 1999, "Using genetic algorithms to find technical trading rules", Journal of Financial Economics, vol. 51, no. 2, pp. 245-271.

# Appendix

1. Special thanks to Michael Fenton for his support with PonyGE2.
2. All code can be seen at the following Github repository:

<https://github.com/eoincUCD/NC_GE_Evolved_Trading_Strategy>

1. The PonyGE2 library can be seen at the following Github repository:

<https://github.com/jmmcd/PonyGE2>

1. Interesting web links:
   1. <http://fabian-kostadinov.github.io/2014/09/01/evolving-trading-strategies-with-genetic-programming-an-overview/>
   2. <http://jonathankinlay.com/2014/06/developing-trading-strategies-with-genetic-programming/>