# Co-evolved Genetic Program for Stock Market Trading

by

Jason Frederick Nicholls

Submitted in partial fulfillment of the requirements for the degree
Magister Scientia (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

September 2018

Publication data:

Jason Frederick Nicholls. Co-evolved Genetic Program for Stock Market Trading. Master's dissertation, University of Pretoria, Department of Computer Science, Pretoria, South Africa, September 2018.

Electronic, hyperlinked versions of this thesis are available online, as Adobe PDF files, at:

http://cirg.cs.up.ac.za/

https://repository.up.ac.za/

**Co-evolved Genetic Program for Stock Market Trading**

by

Jason Frederick Nicholls

## Abstract

This thesis compares the profitability of trading rules evolved by a single population genetic program (GP), a co-operative co-evolved GP, and a competitive co-evolved GP. Profitability was determined by trading thirteen listed shares on the Johannesburg Stock Exchange (JSE) over a period of April 2003 to June 2008. The GP parameters were optimised using a response surface methodology known as $2^k r$ factorial design. A compound excess return over the buy-and-hold strategy was determined as the preferred fitness function via an empirical process. Various selection strategies to select individuals for the crossover and mutation operators were compared. It was found rank selection was the preferred strategy. The optimised GPs were tested on market data using a real world fee structure. The results were compared to a buy-and-hold strategy and a random-walk. The results of this thesis show that the co-operative co-evolved GP generates trading rules that perform significantly worse than a single population GP and a competitively co-evolved GP. The results also show that a competitive co-evolved GP and the single population GP produce similar trading rules. The evolved trading rules significantly outperform the buy-and-hold strategy when the market, including fees, was trending downwards. No significant difference was found between the buy-and-hold strategy, the competitive co-evolved GP, and single population GP when the market (including fees) was trending upwards.

**Keywords:** Co-evolution, Evolution, Genetic Algorithm, Genetic Program, Stock Market, Shares, Trading Rule Optimisation

**Supervisor:** Prof A. P. Engelbrecht
**Department:** Department of Computer Science
**Degree:** Master of Science

The most exciting phrase to hear in science, the one that heralds new discoveries, is not "Eureka!" (I found it) but rather "Hmmm. . . That's funny!"

Isaac Asimov

Dedicated to my amazing daughter Katherine Lynne Nicholls,
I hope you continue to enjoy learning new things as much as I do,
and your enthusiasm for life never fades...

# Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor, Prof. Andries Engelbrecht, for his continuous support of my MSc study and related research, for his patience, motivation, and immense knowledge. I have been extremely lucky to have a supervisor who continuously pushed me to finish, steered me back in the right direction when I jumped to the next interesting research topic, and who cared so much for my work. I really appreciate the amount of time Prof. Engelbrecht spent on validating my research, correcting my numerous first drafts, and teaching me how to follow through. His enthusiasm and passion for computational intelligence is contagious and kept me going even when I lost all faith in continuing.

Secondly, I would like to thank Katherine Malan, for the countless reviews of my first draft and the suggestions she made towards this thesis.

Finally, I must express my very profound gratitude to my Mom and Dad for their sacrifices and support so that I could get an education. I would like to thank my parents, Shay Pretorius, and Gareth Nicholls for providing me with unfailing support and continuous encouragement throughout my MSc. This accomplishment would not have been possible without them. Thank you.

# Contents

# List of Figures

# List of Algorithms

# List of Tables

# Chapter 1

# Introduction

Like alchemy, finding the perfect algorithm to forecast the stock market is the allure of many. While accurately forecasting the stock market is improbable, determining the current market direction is possible. Knowing the market direction and determining if the direction will continue or change can be used to turn a profit. Market traders fall into one of two camps: either a *fundamental trader* or a *technical trader*. The fundamental trader studies a company's fundamentals such as budget, financial books, management, and demand, to name a few. The fundamental trader wants to understand how the company turns a profit, and if the profit will continue or change. The fundamental trader knows that profit results in higher share prices, and potentially dividends.

The technical trader studies the historic share price, and understands that the knowledge the fundamental trader has, determines the market price, and therefore that the market price reflects all known information about a share. The technical trader employs statistical techniques, known as technical analysis, to determine the market trend and potential change in the trend. A technical analysis function is generally used in conjunction with other technical analysis functions. Each technical analysis function has a set of parameters that must be set for the given share. The problem for a technical trader is selecting which technical analysis functions to use, and the values of the parameters required by the technical analysis functions selected.

The combination of technical analysis functions and parameters is known as a *trading rule*. Finding a trading rule for a given share is an optimisation problem. Computers are good at solving optimisation problems. However, testing the sheer number of technical analysis functions and parameters is a mammoth task that can, even for a computer, take a very long time. Instead of comparing every combination of technical analysis functions and parameters to each other to find the best possible solution, a smarter approach is needed.

One such approach is to use a meta-heuristic algorithm to find an optimal combination of technical analysis functions and parameters. Meta-heuristic algorithms search the solution space by removing sections of the search space that produce poor solutions. The result is an approximate solution. One such class of meta-heuristic algorithms is evolutionary algorithms [52, 59, 103, 111].

Evolutionary algorithms use the concept of simulated evolution to produce many solutions to the problem stochastically. Each generation probabilistically selects the best solutions to produce similar solutions that lead to the best approximate solution, in this case a selection of technical analysis functions and parameters. The process leading to the best approximate solution is referred to as evolution.

The problem is still which technical analysis functions should be tested to form a trading rule. While many functions exist, many new functions are created each year. To find an optimal combination of technical analysis functions requires a comprehensive list of all the possible technical analysis functions. A comprehensive list is not feasible and therefore either a sub-set of technical analysis functions are used, or an evolutionary algorithm could evolve trading rules from basic mathematical operations. A special evolutionary algorithm exists that can do just that, called genetic programming [52, 59, 103, 149].

Originally developed by Koza [149] in the late 1980s to evolve computer programs, a genetic program (GP) can evolve trading rules using basic mathematical operations. Allen and Karjalainen [43, 44] used a GP to evolve trading rules. Their approach used a single population of individuals to evolve a trading rule over a fixed number of generations. Allen and Karjalainen noted that, while their approach produces trading rules that generate profitable returns, their GP was relatively simple and the parameters were not optimised [43, 44].

This thesis extends the work of Allen and Karjalainen by empirically optimising the evolutionary operators and parameters of their GP. The empirical process is based on the central limit theorem [40] and uses the statistical tests proposed by García *et al.* [122, 123, 124], and Demšar [96]. The complete empirical process is presented in Chapter 5, Section 5.2.1.

The GP parameters were optimised using a response surface methodology known as $2^k r$ factorial design [93]. The results of parameter sensitivity analysis are presented in Chapter 5, Section 5.5.

In nature, populations rarely evolve in isolation. Instead, populations evolve in co-operation with other populations or in competition with other populations [185]. This thesis examines the performance of trading rules evolved through co-operative and competitive co-evolution by comparing trading rules evolved by co-evolution to trading rules derived by the single population GP implemented by Allen and Karjalainen [43, 44].

The results of this thesis show that a co-operative co-evolved GP generates trading rules that perform

significantly worse than a single population GP and a competitively co-evolved GP. The results also show that a competitive co-evolved GP and a single population GP produce similar trading rules. The evolved trading rules significantly outperform the buy-and-hold strategy when the market, including fees, was trending downwards. No significant difference was found between the buy-and-hold strategy, the competitive co-evolved GP, and the single population GP, when the market including fees was trending upwards.

This chapter serves as an introduction. Section 1.1 outlines the motivation for this thesis. The motivation is followed by the objectives of the thesis in Section 1.2, the methodology used in Section 1.3, and contributions made in Section 1.4. The introduction chapter ends with an outline of the chapters that follow.

## 1.1 Motivation

A stock market is a registrar of purchases and sales of commodities, securities, or equity. A registrar is an institution, often a bank or trust company, responsible for keeping records of bondholders and shareholders after an issuer offers securities to the public. Investors list offers to buy or sell shares on the market, and a broker matches the transactions to facilitate the trade. Deciding when to buy or sell a share is a personal process. Some investors trade on feelings, some on market news. There are those that make use of economic indicators such as profit sharing, currency value, gold reef grade, or competent managers. The process of using economic indicators for market forecasting is known as fundamental analysis [173]. Some traders believe that the share prices have already absorbed all the feelings, news, and market sector information. This belief is referred to as the efficient market hypothesis [106]. Using the share price as a means of determining when to buy and sell is known as technical analysis [173]. Edwards [101] defined technical analysis as the study of the action of the market itself as opposed to the study of the goods the market deals.

Charles Dow [85, 101, 147] studied market action, and proposed the first scientific stock movement theory known as Dow theory [69, 101]. Dow theory states that a market is either in an upward trend, downward trend, or continuing in the same direction [69, 101, 182]. Dow theory proposes that each trend has three phases [69, 101, 182].

In the accumulation phase, investors close to the company start selling stock to the public or buying back stock from the public. In the public participation phase, the public begins to sell stock to other members of the public or buy stock back from the public. During the distribution phase the market is mixed, and no definite trend has formed.

Technical analysis techniques use the historic share price to find which of the three Dow trends the market is in. Technical analysis then determines which phase the trend is in and if the trend will change. This thesis explores just ten of the many technical analysis functions available [42, 53, 71, 84, 101, 173, 178, 197, 203, 206].

Each technical analysis function requires a set of parameters that must be set. Technical analysis functions are used in combination with one another.

Selecting which technical analysis functions to use, and determining the correct parameters is an optimisation problem. Fixing the set of technical analysis functions to size $k$ and the parameters to a set of size $n$ takes $O(n^k)$ time [104] to compute on a Turing machine [36]. This class of complex problems is known as NP-complete problems [36, 104]. Searching through all the combinations of technical analysis functions and parameters is not computationally feasible. Meta-heuristics offer an alternative approach to searching through all the combinations of functions and parameters. Meta-heuristics search the solution space for the optimal solution by removing sections of the search space that produce poor solutions.

Bauer [63] was one of the earliest researchers to use a class of meta-heuristics known as genetic algorithms [135] to optimise the combination of technical analysis functions and parameters to maximise return on a set of stock trades.

A genetic algorithm is based on simulated evolution. The theory of evolution, first proposed 2 600 years ago by the Greek philosophers Anaximander [81, 100] and Empedocles [41, 127], proposes that all individuals are derived from a set of common ancestors created during the big birth.

Comte de Buffon [76] believed that creatures morph adapting to their environment, a belief shared by Lamarck [94] and documented in his book “Philosophie Zoologique”. Lamarck believed that individuals lose characteristics they do not require and develop those which are useful.

Darwin, E. [90] rejected the morphing hypothesis of Lamarck and Buffon, and proposed that each individual has slightly different traits to others and their traits are passed on their offspring. Darwin, C. [53, 68, 89, 111, 201] and Wallace [89, 208] introduced the scientific theory of evolution. They proposed that all species of life have descended over time from common ancestors. The branching of species resulted from a process that Darwin called natural selection, in which the struggle for existence within different environments favours different traits.

The German evolutionary biologist, Weismann [209] refined evolution by introducing heredity. Weismann found that individuals pass genetic information on to their children via germ cells, and that the information can mutate.

Mendel [53], an Augustinian priest and scientist, conducted hybridisation experiments on garden

peas (*Pisum sativum*) and discovered that the information passed on alters the traits of the offspring. Therefore, the mutated genetic information postulated by Weismann resulted in new characteristics within the garden pea.

An American geneticist and embryologist, Morgan [111, 168], studied a small fly (*drosophila*), and determined that a chromosome stores the traits of an individual as genes, and that these genes are selected and passed on to offspring. The genes may undergo mutation before the offspring forms.

Neo-Darwinism is a theory that combines natural selection, germ cells, inheritance, and genetics [111]. Neo-Darwinism consists of four operations, namely reproduction, mutation, competition, and selection [111].

A genetic algorithm simulates the four operations of Neo-Darwinism by creating a population of candidate solutions to an optimisation problem. Candidate solutions are called individuals. Each individual has a set of properties defined as a chromosome. Each property is known as a gene.

Chromosomes are altered through simulated mutation and reproduction to form new individuals referred to as offspring. Selection samples individuals from the population, forming the next generation. Typically, selection is biased towards the best performing individuals.

A fitness function quantifies an individual's performance [103]. Performance is generally determined by how well the candidate solution completes the problem.

The process of selection, reproduction, mutation, and competition is repeated until an adequate solution is found or a fixed number of generations have passed [111, 112].

A genetic algorithm requires a chromosome representation of the solution domain and a fitness function to evaluate the individual within the domain.

Friedman and Fraser [53, 112, 121] are recognised as the first to experiment with genetic algorithms. However, it was Holland [53, 135, 136, 137] that formalised the first version of a genetic algorithm. Holland's emulation of evolution used a fixed length binary string representation of a chromosome.

This thesis defines the optimisation problem as finding a trading rule that can reliably return a profit for a given share. If possible the trading rule should return a higher profit than a buy-and-hold trading strategy. The solution domain is then defined as a domain containing every possible trading rule.

The bit string chromosome representation of a trading rule implies that each bit corresponds to a technical analysis function and fixed parameters. The bit enables or disables the technical analysis function. The combination of enabled technical analysis functions represents a trading rule.

Replacing a bit string vector chromosome structure with a mixed data-type vector facilitates the storage of technical analysis parameters within the chromosome, allowing the genetic algorithm to evolve the parameters to an enabled function.

To evolve the technical analysis functions from basic mathematical operators requires a variable length chromosome structure that defines a relationship between genes within the chromosome. The chromosome structure must cater for binary, logical and numerical functions. Koza [150] experimented with an alternative version of a genetic algorithm called a GP. A GP replaces the fixed length vector structure used in a genetic algorithm with a random length, non-linear, hierarchical abstract data structure, known as a tree.

The tree structure naturally accommodates a trading rule. A tree is not restricted to a fixed length. A tree is a natural structure to represent hierarchical decisions rules, and a tree can store both binary, logical, and numeric functions within its nodes. Logical functions allow the GP to evolve conditional execution of technical analysis functions not easily implemented using a fixed length vector.

Allen and Karjalainen [43, 44] were one of the earliest implementers of GPs in stock market trading. Allen and Karjalainen used a GP to evolve trading rules from basic mathematical, and logical operators. They showed that the trading rules generated by their GP are generally beneficial when the market falls or when it is stable. Although their GPs were profitable, the excess returns were not dramatically higher than that of a buy-and-hold strategy. Allen and Karjalainen noted that, while their approach produces trading rules that generate profitable returns their GP was relatively simple and the parameters were not optimised [43, 44].

Variations of Allen and Karjalainen's implementation of a GP were reimplemented by Neely *et al.* [170], Telbany [102], Mahfoud and Mani [160, 161], Li and Tsang [204, 205], and Potvina *et al.* [177] with varying degrees of success.

This thesis extends the work of Allen and Karjalainen by empirically optimising the evolutionary operators and parameters of their GP.

To be profitable when trading shares, the selling value less the buying value must be greater than the transaction cost. Because trading costs are generally high, Allen and Karjalainen noted that only large institutional investors with lower trading costs could implement evolutionary algorithms and be profitable [43, 44]. Allen and Karjalainen used a percentage of the share price as a cost, which is not the same cost structure employed by brokers. This thesis differs from Allen and Karjalainen's implementation by incorporating the share trading fees charged by the Standard Bank's online share trading platform (OST) [32, 43, 44]. The OST is a publicly accessible share trading platform.

As discussed previously, meta-heuristics search the solution space for an optimal solution by removing sections of the solution space that produce poor solutions. Sometimes meta-heuristics fail to search particular areas of the solution space, or remove areas of the search space inadvertently. When this happens, the meta-heuristics return sub-optimal solutions.

Evolutionary algorithms use the concept of simulated evolution to produce many solutions to the problem stochastically. The best solutions are selected to produce similar solutions that lead to the best approximate solution. Over time the individuals in a generation become more and more similar. When they are so similar that all the solutions produce the same result, the solutions are said to have converged.

Premature convergence results in inadvertently ignoring parts of the solution search space. Mutation introduces new genetic material, therefore introducing parts of the ignored search space. However, sometimes mutation alone is not enough to stop premature convergence.

Convergence and premature convergence is a result of selection. Selection is based on the evaluation of the performance of individual solutions with respect to the performance of others. Individuals with the highest performance have a higher probability of being selected as individuals for the next generation than individuals with a lower performance.

Allen and Karjalainen implemented a single population GP. In nature, populations rarely evolve in isolation. Instead populations co-evolve in co-operation with other populations or in competition with other populations.

Co-evoled populations evolve in isolation ensuring that each population can explore the solution space independently [111]. Co-operative co-evolution [111, 132] results in populations evolving parts of the solution independently. The performance of individuals are based on the performance of all the populations. Competitive co-evolution [132] results in populations competing against each other. The performance of individuals in one population are indirectly proportional to the performance of individuals in other populations.

This thesis extends the single population GP of Allen and Karjalainen to two populations and compares the effect co-operative and competitive co-evolution had on evolving trading rules.

The motivation behind the introduction of co-evolution is to minimise both over-fitting and premature convergence.

## 1.2 Objectives

The main objective of this thesis is to study the effectiveness of an evolved trading rule on real-world market conditions. The primary objectives of this thesis are summarised as follows:

- to implement the standard GP of Allen and Karjalainen;
- to empirically optimise the parameters used by the GP;

- to determine a preferred selection operator for the GP;
- to determine a preferred fitness function for the GP;
- to implement a co-operative co-evolved version of the GP;
- to implement a competitively co-evolved version of the GP;
- to compare trading rules evolved using co-evolution to single population evolution;
- to determine the profitability of evolved trading rules on the Johannesburg stock exchange (JSE) against chance and a buy-and-hold strategy;

## 1.3 Methodology

The stock market is presented and discussed, with exploration of different trading approaches, namely fundamental and technical analysis. The origins of technical analysis is presented, with a review of the most popular technical analysis functions. Evolution and simulated evolution is presented and discussed; more specifically, genetic programming is explored as a meta-heuristic search that has been shown to be successful in optimising solutions.

Research in using genetic algorithms to find optimal technical analysis functions and parameters is presented. The work of Allen and Karjalainen [43, 44] is explored. A set of parameter sensitivity experiments were run to find a good GP parameter configuration for the GP implemented.

Co-evolution is explored as a multi-population extension of the GP. The optimised single population genetic GP was implemented as both a co-operative and a competitive co-evolution GP. The GP parameters were optimised using a response surface methodology known as $2^k r$ factorial design [93]. The results of parameter sensitivity analysis are presented in Chapter 5, Section 5.5. Each configuration of a GP constitutes a simulation. A simulation was run 30 times to comply with the central limit theorem [40]. The null-hypothesis was assumed when comparing the results of various GP configurations. The null-hypothesis assumes the results to be the same, unless statistically shown to be significantly different. Significance was determined at a 95% confidence level using statistical tests proposed by García *et al.* [122, 123, 124], and Demšar [96].

Only configurations shown to invalidate the null-hypothesis were investigated further. Investigation compared the mean profit returned by the simulation runs to determine which configuration was preferred. The complete empirical process is presented in Chapter 5, Section 5.2.1.

All comparisons were performed using real world financial data. Each simulation was repeated for each of the financial datasets. The financial datasets comprised of thirteen different shares across various market segments selected from the JSE. To compare the performance of the evolved trading rules against chance, the profitability of the trading rules were compared to four simple trading strategies. The first was a buy-and-hold strategy that bought shares on the first day of trade and sold them on the last day of trade. The other three trading strategies were implemented as variations of a random-walk. The random walk determines a buy and sell action randomly.

## 1.4 Contributions

The contributions offered by this thesis are:

- Empirically optimised GP for stock market trading.
- Empirical comparison of co-evolution and a single population GP using real work stock market data.
- Empirical evidence that a GP can evolve trading rules to outperform the buy-and-hold when the market is trending down.
- Results that show trading rules generated by a GP perform significantly better than chance.

The following list of published articles support the main contributions of this thesis.

- J.F. Nicholls, K.M. Malan, and A.P Engelbrecht. Comparison of trade decision strategies in an equity market GA trader. *In Proceedings of the IEEE Symposium on Computational Intelligence for Financial Engineering and Economics* 2011.
- J.F. Nicholls, K.M. Malan, and A.P Engelbrecht. Evaluation of Fitness Functions for Evolved Stock Market Forecasting. *In Proceedings of the WSPC Symposium on Artificial Intelligence for Economics Research Centre* 2008.

## 1.5 Thesis Outline

This thesis is organised as follows:

- **Chapter 2** describes share trading and technical analysis. The chapter begins with an introduction to stock markets, and a discussion on fundamental and technical analysis. Technical analysis is then described in more detail by briefly covering its history, followed by exploration of the most common technical analysis functions, their use, and the formulae used to implement them. The chapter introduces simple trading strategies and a transaction cost model used by Standard Bank.

- **Chapter 3** covers the history of evolutionary theory and introduces the basic evolutionary operators. The history of natural evolution is followed by an introduction to evolutionary algorithms and the history of genetic programming. The chapter concludes with a section on co-evolution and co-evolved GPs. This is followed by a discussion on evolutionary algorithms, its history, and the various implementations thereof, including genetic algorithms, genetic programming, evolutionary programming, evolutionary strategies, and differential evolution. Genetic programming is covered in more detail within its own section, which is followed by a section on co-evolution.

- **Chapter 4** combines technical analysis and evolutionary algorithms by introducing the history of evolutionary algorithms applied to automated stock market trading. The history begins with research in using genetic algorithms to select technical analysis functions, which is followed by evolving trading decision trees using GPs.

- **Chapter 5** builds on the work presented in Chapter 4 by presenting the GP used in the empirical study. The presentation includes an outline of the GP algorithm as pseudo code. A discussion on various aspects of the algorithms and the decisions made. The decisions include the selection criterion, mutation operators, crossover operators, and fitness functions. The various operator parameters are explored by running parameter sensitivity analysis. The results are presented and discussed. This chapter introduces the pre-requisites of the empirical study including stock market data used, the simulation set up, empirical process, and empirical study configuration.

- **Chapter 6** extends the single population GP presented in Chapter 5 by defining two co-evolution versions. The first version is a co-operative co-evolution extension, and the second a competitive co-evolution extension. The basic algorithms are defined as well as the fitness functions.

- **Chapter 7** presents the empirical study of this thesis, beginning by confirming the GP configuration used. The configuration is followed by the study consisting of three sub sections. The first subsection compares the results obtained by the three GPs across various market shares without fees. The second subsection discuses the results with the inclusion of fees. The third subsection

compares the profit results obtained by the standard GP and the competitive co-evolved GP to four random trading rules.

- **Chapter 8** highlights the conclusions of this thesis and presents ideas relating to possible future work.

- **Appendix A** examines the data used throughout this thesis. The market conditions are discussed as well as the reasons for selecting the dataset.

- **Appendix B** provides a list of the important acronyms used or newly defined in the course of this work, as well as their associated definitions.

- **Appendix C** lists the publications derived from this work.

# Chapter 2

# The Stock Market

*One of the funny things about the stock market is that every time one person buys, another sells, and both think they are astute.*

*-William Feather (American Publisher and Author)*

The purpose of this chapter is to explain share trading and technical analysis. Section 2.1 describes stock markets and how supply and demand can drive share transactions. Section 2.1 introduces two analytical techniques for understanding the forces of supply and demand: fundamental analysis and technical analysis. Section 2.2 presents the history of technical analysis, while Section 2.3 highlights modern technical analysis techniques. Section 2.4 discusses different trading platforms and the transaction cost structure used in the empirical study. Section 2.5 offers a summary of the chapter.

## 2.1 Introduction to Stock Markets

A stock market or equity market is a gathering of people for the purchase and sale of commodities, securities, or equity.

A commodity is a physical item that can be bought or sold, for example, rice or gold. A security is something used as a guarantee for the undertaking of a loan, also known as collateral, surety, or bond. An equity is a stake or share in a company. A share refers to the ownership of a specific company, while stock refers to shares in many companies.

The process of buying and selling these items is known as trading. The stock exchange manages the market and regulates trading.

The market facilitates the trade in equity, providing a place for a company to raise capital. The company gains capital by selling a share of ownership to investors, who in return, for cash, gain a percentage of ownership, but more importantly, a right to the profit in the form of dividends. Management of dividends, company information and shareholder information is the function of the exchange. Listing of new equities, commodities or securities is the function of a market maker. A broker matches the transactions of buyers and sellers.

Before the invention of the computer, stock markets were lively markets. Brokers would arrange a trade between the market maker and the buyer or seller through shouting, pushing and shoving within the trading area. Once visually or verbally acknowledged, the broker informs the exchange via a sliver of paper and the exchange acknowledges the transaction in the share register. Furthermore, a market maker selling from its inventory of stock, tries to offset the sale with a purchase of another share. The physical trading process is known as a trading pit or open floor trading system.

The New York stock exchange (NYSE) still operates on this open floor trading system, while the London stock exchange (LSE) and the JSE went digital in the 1980's and 1996 respectively [5, 48]. A digital exchange replaces paper and books with databases, and human brokers with digital transaction matching brokers.

Today the market is an interconnected web of computers, which buy and sell at the speed of light. Although this market seems to be ahead of its time, the same principles of market maker, willing buyer, and willing seller exist, and the same fears and excitement drives the price.

The following illustration explains the stock market process:

Consider a company that needs money to open a new mine. This company would list a finite number of shares with an appropriate market maker based at an exchange such as the LSE or the JSE. The exchange lists the share on behalf of the company at a pre-determined price. The market maker considers the share demand and adjusts the price accordingly. Investors buy the shares at the market price and the company raises the money needed. The new shareholders gain rights in the company which includes a vote on the running of the company, and a share of the profit.

Investors buy shares in a company for two reasons:

- to obtain a share of the future profit, and
- to speculate and sell the share at a higher price than what they paid for it.

Investors sell shares for several reasons: it may be that the shareholder needs money, or that the share value is dropping to a level that they have incurred a loss, or that the perceived profit is less than expected, or perhaps the profit has reached an all-time high and is likely to start diminishing. The perception of future profit or share price is based on beliefs or truths in factors that affect the market and the company at a point in time. Such factors include the state of the economy, the management of the company, and press releases related to the company or the sector that the company operates in.

If an investor wants to sell a share in the company, the share is listed on the market at a selling price determined by the investor. A prospective investor lists with the market an interest in buying a share in the company at a price decided by the prospective investor. If the seller's price matches the buyer's price, the broker fulfils the transaction, with the seller receiving money and the buyer a share in the company. The process of buying and selling is known as trading. Once the transaction takes place, a new market price is set and reflected against the company's trading code. Trading incurs levies and costs that are determined by the exchange and the country that the exchange resides in.

More buyers than sellers create a higher demand with little supply. High demand means that the seller decides the price, and so the offer price increases. Fewer buyers than sellers create a lower demand with a higher supply. Low demand pushes the price down to attract buyers. The fluctuation of price based on supply is known as the law of supply and demand [48].

Determining when to buy or sell a share is a personal process. Some investors trade on feelings, some on news. There are those that make use of economic indicators such as profit sharing, currency

value, gold reef grade, or competent managers. The process of using economic indicators for market forecasting is known as fundamental analysis [101]. Fundamental analysis requires a deep understanding of the specific sector of the market traded in. Some traders believe that the share price has already absorbed all the feelings, news, and market sector information. These traders believe that patterns in the share price can decide the future market value of the share. This type of forecasting is known as technical analysis [101]. Technical analysis makes use of numeric data readily available. This study focuses on technical analysis used in stock market trading. The next section briefly covers the history of technical analysis.

## 2.2 History of Technical Analysis

Edwards [101] defines technical analysis as "the study of the action of the market itself as opposed to the study of the goods the market deals". Legend has it that the first technical analyst was Munehisa Homma [162]. Homma was a rice merchant from Sakata, Japan, trading in the future price of rice. Through the years, he learnt that the emotion of traders had a significant effect on the price of the rice. His understanding of this market psychology was published in his book "San-en Kinsen Hiroku" in 1755. Using his knowledge of the market and its psychology, Homma became legendary in the forecasting of the rice price. To aid his forecast, Homma used a charting technique that displayed the opening, closing, high, and low rice price for a given trading day. This charting technique was so efficient and simple that it became widely used as a method to determine future demand, based on patterns within the chart. This form of charting became known as candlestick charting [180].

Candlestick charting is a visual tool that requires interpretation, which is subject to human subjectivity and not very scientific. One hundred years later the first editor and founder of the Wall Street Journal, Charles Dow [85, 101, 147], proposed the first scientific stock movement theory that would later become the bases for all technical analysis techniques [69, 101].

Dow proposed the following principles:

- Markets are either in an upward trend, downward trend, or markets continue in the same direction. In the up-trend, the market continues to close higher than the previous high. In the down-trend the market continues to close lower than the previous low. Markets can continue in the same direction, neither higher nor lower, but oscillate between the two.
- Each trend has three phases: an accumulation phase, a public participation phase, and a distribution phase. In the accumulation phase, investors close to the company start off-loading stock

to the public or buying back stock from the public. In the public participation phase, the public catches on and starts to off-load stock to other members of the public or buy stock back from the public. In the distribution phase the market is mixed, and no definite trend has formed. The distribution phase is also known as the speculation phase.

- The stock market absorbs all new information. This is the same as the efficient market hypothesis (EMH) as stated by Fama [106, 107]. The market also absorbs information related to future share prices as forecast by any trader.

William Hamilton [69], the fourth editor of the Wall Street Journal [85], added the following to the Dow theory [69, 182]:

- The primary trend of a share can not be manipulated.
- Stock market averages must confirm each other. This is a very strong premise. In short, if one industry is reliant on another industry, for example, the farming and bread industries, it must hold that if the bread industry is doing well, then the farming industry supplying the wheat should be doing equally well. The transport industry transporting all the bread and wheat should also be benefiting from the success in the farming and bread industries. Change in market trend will occur if there is no correlation in performance between the industries that rely on each other.
- Price trends must be confirmed by volume (the masses must agree with the price through a large number of trades).
- Trends will continue despite market noise until the signals above indicate that the market trend has changed.

In 1921, Rhea [162] tried to disprove the Dow theory only to successfully forecast the bottom of the market in 1932 and the peak in 1937. Building on the success of Rhea [182], Schaefer [187] developed additional technical analysis models to forecast trend changes. Schaefer published these techniques in the book titled “How I Helped More than 10,000 Investors to Profit in Stocks” [187].

Although fundamental analysts such as Fama [43, 106, 107, 125, 147, 162, 189, 204] claim that the market follows a random-walk, technical analysts have shown that patterns do exist within the market. The American Statistical Association (ASA) [147] held a meeting on April 24, 1934 to discuss the effectiveness of technical analysis. It was at this meeting that three analysts presented papers. The following outlines a summary of some key points from the meeting:

Ellsworth [147], the editor of the "Annalist", stated that stock market prices are different from cumulative chance in that:

- unlike chance data, stock prices can not go below zero, and
- the fact that stock prices are tied to earnings, limits the altitude to which they can climb.

Macaulay [147] commented on the relationship between chance and the market by stating that if a slightly loaded dice (i.e. not balanced correctly) is thrown and the results are plotted on a graph, the plot will not be uniform. This is similar to the results seen in a market. The shorter the term, the more the market looks like a balanced dice; while the longer the sample or term, the more distorted the results.

Supporters of Fama's EMH claim that the market is efficient, and the share price reflects all the traders' feelings and assumptions of the market price. The trading price absorbs all known market and stock information as soon as the information is known. Gartley [147] argued that this is not possible, because Dow showed that market trends enter three distinct phases. Gartley insisted that the trading price does not change at once, but rather gradually with each trade. Price performance on a specific day could indicate the probable changes that would occur the next day, as the market moved through the three phases.

Schabacker [147], the financial editor of Forbes Magazine in 1934, was also present at the meeting. He put forth his thesis on the logic of the technical approach and presented the following simple truths:

- the stock market moves in trends,
- more people buying than selling defines an upward trend,
- more people selling than buying defines a downward trend, and
- while the majority of people buying or selling is not important, it is important to know when the majority of people are buying or selling and to do the same.

The American Statistical Association meeting showed that there is a basis in believing the market data may offer foresight into the future market price. Based on this belief, several technical analysis techniques have been developed to analyse the market data and to determine the current trend. The technical analysis techniques are also used to predict a change in the trend. Knowing the trend, and if the trend is changing, gives guidance as to when a trader should buy or sell a share.

## 2.3 Technical Analysis Techniques

Technical analysts have — through the years — plotted the share price on a special type of graph called a candlestick graph. The candlestick graph, invented by Homma [180], is illustrated in Figures 2.1 and 2.2. This graph shows the opening, closing, high, and low stock market values for a given day with wide vertical bars and lines called shadows [173]. Analysts then added the number of individual stock trades made within the day to the graph. The total number of trades made within a period is known as the volume.

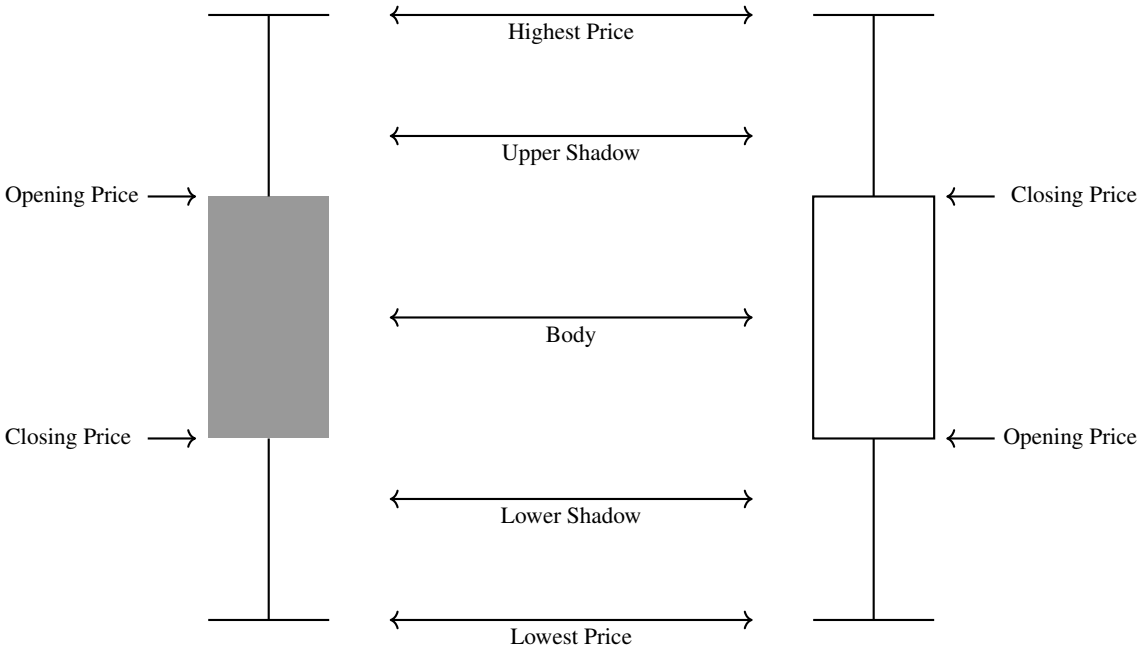



**Figure 2.1:** The format of candle stick used in the candle stick graph invented by Homma. An open block represents a share trending upwards, where the opening price is lower than the closing price. Conversely, a filled block represents a share trending downwards, where the closing price is lower than the opening price. The high and low whiskers represent the highest and lowest recorded price of the trading period.

Through careful observation of the candlestick graphs, analysts found different techniques for reading the graph. As noted by Macaulay [147] in the meeting of the American Statistical Association [147], like a loaded dice the short-term market appears random, but over a longer period the dice appears "biased" and the market appears to follow a trend. At first, the graph appears random. However, many analysts employ visual patterns to determine the trend. These patterns are subjective human interpretations of the data. The reader of the graph attempts to use the patterns to decipher the market trend within the perceived randomness of the market [173].

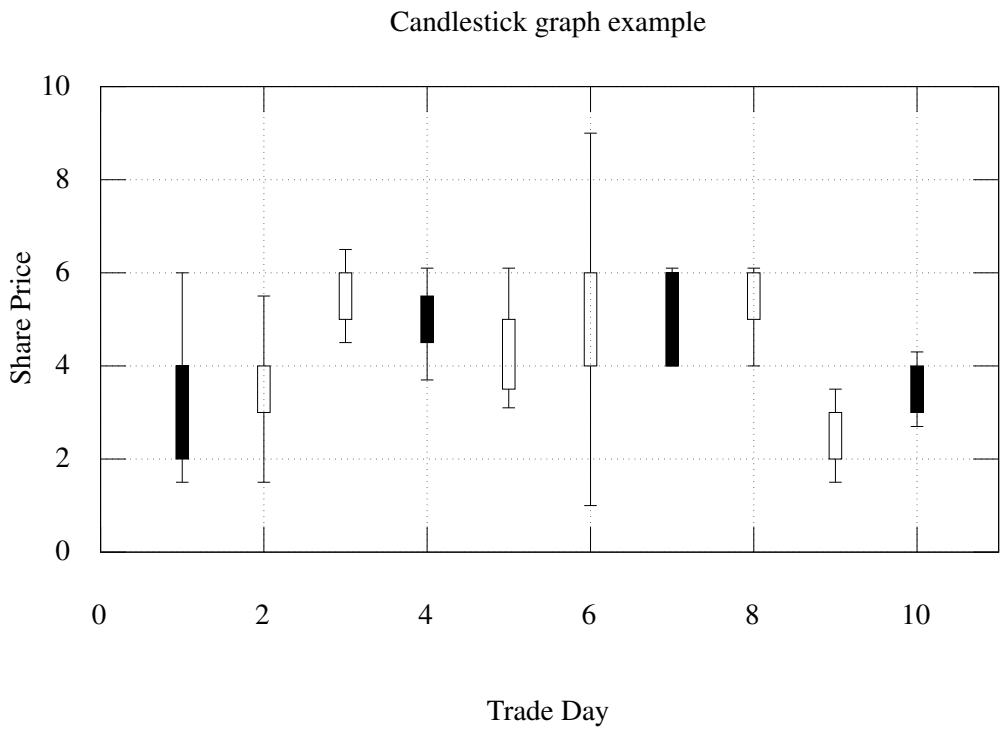



**Figure 2.2:** An example of the candle stick graph invented by Homma.

To better detect the trend, technical analysts began to employ correlations between the volume and the trading price, low and high boundaries, and increased deviations from the established trend. The volume of the day represents the total number of trades captured for that day. The volume for a given day $t$ is represented as $Volume_t$.

The trading price is either the opening price, closing price, the highest recorded price, or the lowest recorded price of the trading day $t$ and is represented as $Price_t$.

The market bias is difficult to detect over the short-term market, because the market is full of noise in the form of short-term or daily fluctuations. To improve forecasting, technical analysts smooth out the noise and view the market averages over a longer period than a day. A moving daily average function is an example of a simple noise reduction function.

The simple moving average (SMA) has its foundation in Dow's theory [69, 147]. Dow believed that if the share price is on the rise, the probability of a continued rise is greater than a fall and vice versa [85, 143, 147]. Consider a graph of the daily closing share price illustrated in Figure 2.3. The

share price could be up in value on one day, but down in value the following day. The daily share price is jagged and unpredictable. However, consider a graph that plots the average price of $x$ days before the trade day. This graph appears smoother, and the trend becomes visible [84]. Increasing the number of days ($n$) used to calculate the average results in a smoother graph. Moving averages are the most common methods employed by technical analysts. Many variations of the moving average exist. The three most common variations include:

- the SMA, expressed as:

$$SMA(n) = \frac{Price_t + Price_{t-1} + \cdots + Price_{t-n+1}}{n}$$

- the weighted moving average (WMA), expressed as:

$$WMA(n) = \frac{\omega Price_t + (\omega - 1)Price_{t-1} + \cdots + 2Price_{t-n+2} + Price_{t-n+1}}{\omega + (\omega - 1) + \cdots + 2 + 1}$$

  where $\omega$ is a weight. In this example, $\omega$ is equal to $n$.

- the exponential moving average (EMA), expressed as:

$$\alpha = 1 - \frac{2}{n+1}$$

$$EMA(n) = \frac{\alpha^n Price_t + \alpha^{n-1} Price_{t-1} + \cdots + \alpha^2 Price_{t-n+2} + \alpha Price_{t-n+1}}{\alpha^n + \alpha^{n-1} + \cdots + \alpha^2 + \alpha}$$

  where $\alpha = 1 - \frac{2}{n+1}$ and $\alpha$ is the weight.

A comparison of different moving average trends can reinforce the trend certainty. For example, if the 10-day SMA follows the same direction as the 50-day SMA, then it can be assumed that the trend is set. If the two moving averages do not follow the same direction, it could signify a trend reversal. A common technique used to determine the trend using two different moving averages is to subtract the longer (i.e. slower) moving average from the shorter (i.e. faster) one. A positive result means the trend is upwards, a negative result means the trend is downwards, and a zero result is a moment of uncertainty. In the 1960s, Appel [42, 53, 203] subtracted a 26-day EMA from a 12-day EMA, and called this technique moving average convergence or divergence (MACD) [42, 203]. A 9-day EMA of the MACD is known as the signal line. The signal line is plotted on top of the MACD, functioning as a trigger for buy and sell signals. When the MACD falls below the signal line it suggests time to sell. When the MACD is above the signal line it suggests time to buy. Generally, the MACD and signal lines

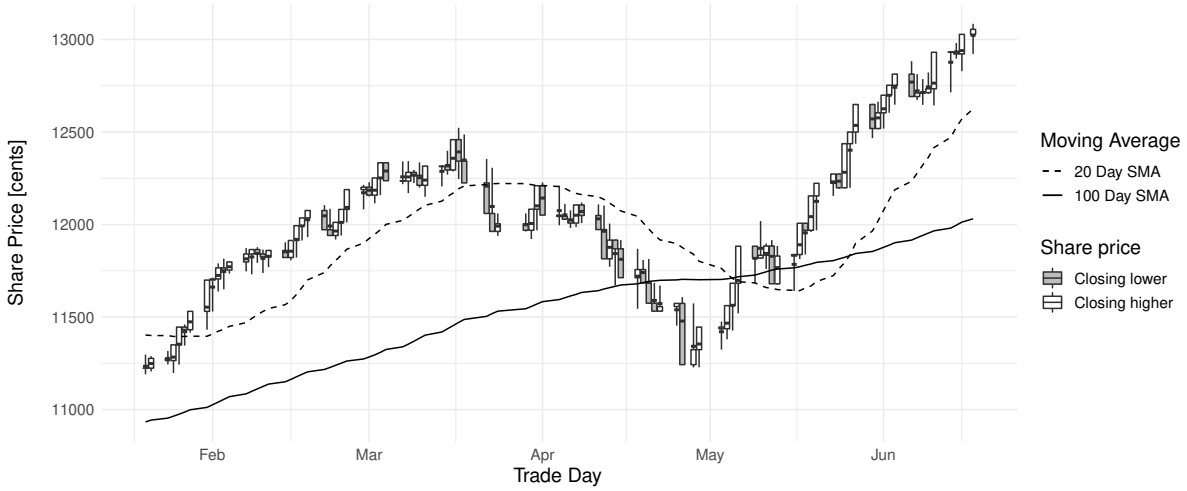



**Figure 2.3:** Illustration of a 20-day and 100-day simple moving average function plotted against the ALSI40 closing price from February 2005 - July 2005.

are plotted as a line graph, and the difference between the MACD and the signal line as a bar graph. A positive difference signals an upward trend and a negative difference signals a downward trend. An example of the MACD is depicted in Figure 2.4. The MACD is expressed as follows:

$$MACD = EMA(12) - EMA(26)$$

Similarly, Chaikin [101] implemented the same subtraction strategy but used a 10-day EMA and a 3-day EMA. His technique is known as the Chaikin oscillator (CO). The CO is expressed as follows:

$$CO = EMA(3) - EMA(10)$$

The CO and MACD result in a number that oscillates above and below zero. Naturally, a positive MACD / CO result means that the faster moving average is greater than the slower moving average and so the market is more positive, or more bullish. This is reinforcement in an upward share price trend. A negative MACD / CO result means that the faster moving average is lower than the slower moving average, and therefore the market is more negative or bearish and supports a downward trend. The MACD / CO may be positive, negative or zero. The further the MACD / CO is from zero, the greater the change in the trend. The MACD and CO are examples of trend following indicators. A trend following indicator is also known as a lagging indicator. Lagging indicators confirm a trend, e.g. is the market going up, down, or remaining the same. Lagging indicators help to reduce risk, as they indicate when the market is bearish even if the share price is high [203].

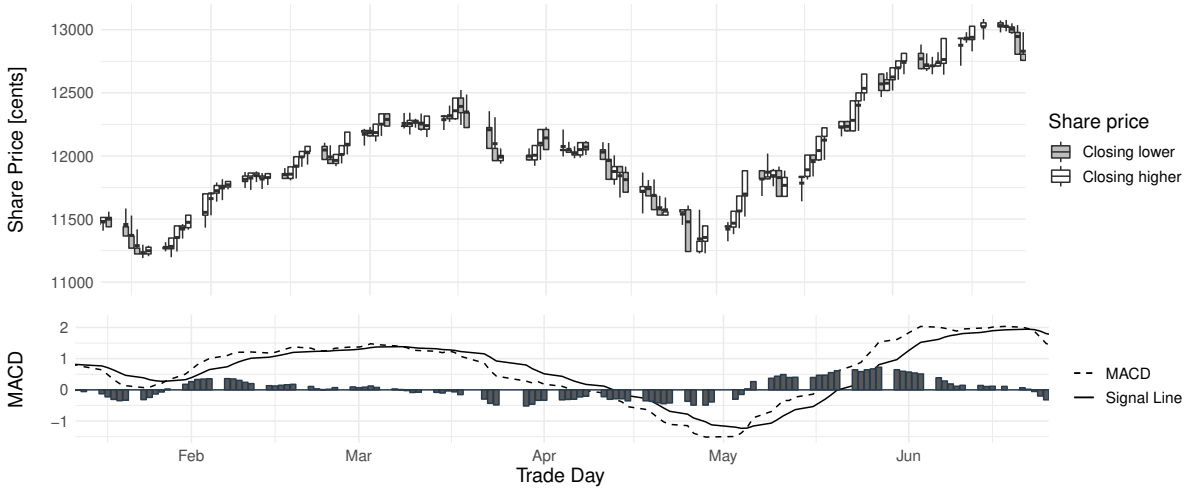



**Figure 2.4:** Illustration of a moving average convergence divergence plotted, lagging indicator function against the ALSI40 closing price from February 2005 - July 2005.

Moving averages confirm the current underlying trend by removing noise. Technical analysts also want to know if this trend will continue, or if the trend will change. Change can be detected by comparing the current price change to the historic mean price deviation. In statistics, the deviation is the difference between the current value and the average or mean value observed over a dataset. In technical analysis, the deviation refers to the average value of the share price over time, also known as the SMA less the current share price [84]. The mean deviation of the moving average (MD) is therefore the mean of the calculated deviation over the share price. Generally, the MD is shown as $\theta$. The MD is expressed as follows:

$$\theta_t = \frac{\sum_{t=1}^{n} \mid SMA_t - Price_t \mid}{n}$$

Technical analysts plot the MD of the moving average on a graph as two lines. The first line is the moving average minus the MD at that point and the other is the MD plus the moving average at that point. This graph was introduced by Bollinger [42, 71, 101] in 1980 and is known as Bollinger bands (BBs). Figure 2.5 illustrates an example of a BB graph. Should the share price represented by the centre line break the upper band, the shares are said to be overbought and trend reversal is due. Breaking the lower band results in an indication of being oversold and the trend may reverse. Another technique to forecast a probable change in the trend, is to measure the rate of divergence between the price trend and the current share value. An oscillating index measures the rate of divergence.

An oscillating index tries to measure the difference between the current established trend and the

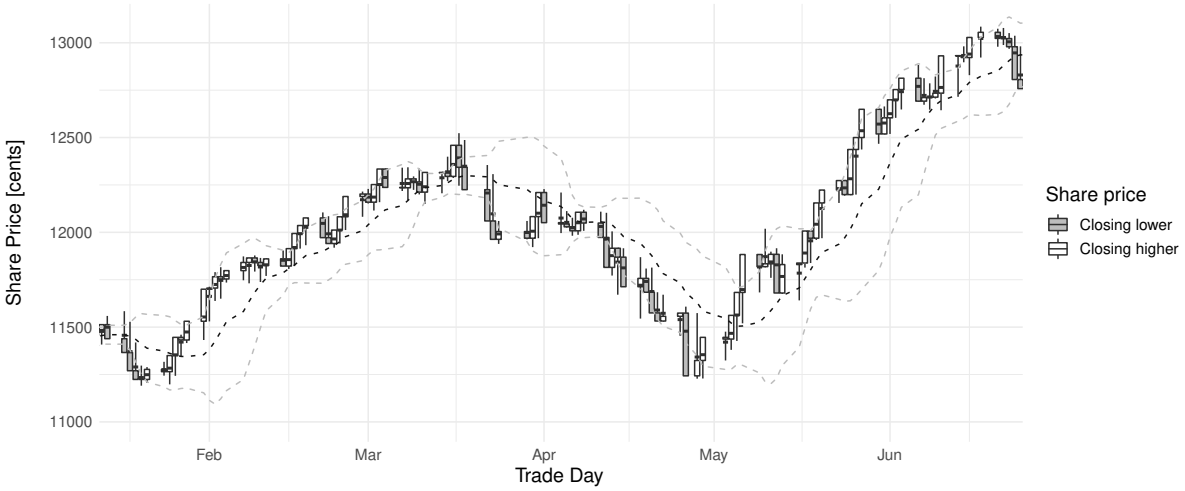



**Figure 2.5:** Illustration of a 10-Day Bollinger bands function plotted against the ALSI40 closing price and volume from 2007–2008.

current share price. Like the BBs, the oscillator attempts to determine if the shares are overbought or oversold, which increases the probability of a trend reversal. The three most widely used oscillating indices are:

- relative strength index (RSI),
- commodity channel index (CCI), and
- money flow index (MFI).

The RSI, developed in 1978 by Wilder [42, 101], shows the strength of the share price relative to the market within a range of 1 to 100 [173]. The closer the RSI is to 100, the more overbought the shares are; and the closer it is to 0, the more oversold the shares are. Wilder recommended a period of 14 days for the calculation and using index threshold values of 70 and 30 to determine overbought and oversold indicators, respectively.

The RSI is expressed as follows:

$$TotalGains_t = \sum_{t=1}^{n} (Price_{t+1} - Price_t) \text{ for all } t \text{ where } Price_{t+1} - Price_t > 0$$

$$TotalLosses_t = \sum_{t=1}^{n} (Price_t - Price_{t+1}) \text{ for all } t \text{ where } Price_t - Price_{t+1} > 0$$

$$AverageGain_t = TotalGains_t / n$$

$$AveragLoss_t = TotalLoss_t/n$$
$$RelativeStrength_t = \frac{AverageGain_t}{AverageLoss_t}$$
$$RSI_t = 100 - \frac{100}{1 + RelativeStrength_t}$$

Lambert created the CCI in October 1980 [197]. While the RSI returns a value between 0 and 100, the CCI returns a number that is greater or less than 0. Under normal trading the CCI should return a value between +100 and -100. If the returned value is above 100, it means that the market has overbought; and value less than -100 means that the market has oversold. To reduce fluctuations, an average of the daily closing, high, and low price is used to calculate the CCI. This daily average price is known as the typical earning price (TEP). The TEP is expressed as follows:

$$TEP_t = \frac{ClosePrice_t + HighPrice_t + LowPrice_t}{3}$$

where $ClosePrice_t, HighPrice_t, LowPrice_t$ is the closing price, highest price, and lowest price respectively for a given day $t$.

The CCI is expressed as follows:

$$CCI_t = \frac{1}{0.015} \times \frac{TEP_t - SMA_t}{\theta}$$

During each day of trade, money flows in and out of the stock market. If the average amount of money traded on a given day is less than that of the following day, the market has gained money (i.e. inflow). If the average amount of money traded on a given day is more than that of the following day, the market has lost money (i.e. outflow). Multiplication of the TEP and the volume of the day returns the average amount of money traded on the day.

Quong and Soudack [178] converted the average amount of money traded between two days into an oscillating index known as MFI. The MFI results in a number between 0 and 100, where anything greater than 80 and less than 20, is considered overbought and oversold respectively. The MFI is expressed as follows:

$$RMF_t = TEP_t \times Volume_t$$
$$PositiveMoneyFlow_t = \sum_{t=1}^{n} (RMF_{t+1} - RMF_t) \text{ for all } t \text{ where } RMF_{t+1} - RMF_t > 0$$
$$NegativeMoneyFlow_t = \sum_{t=1}^{n} (RMF_{t+1} - RMF_t) \text{ for all } t \text{ where } RMF_{t+1} - RMF_t < 0$$

$$MoneyFlowRatio_t = \frac{14 - PositiveMoneyFlow_t}{14 - NegativeMoneyFlow_t}$$

$$MFI_t = 100 - \frac{100}{1 + MoneyFlowRatio_t}$$

The MFI is an example of a technical analysis technique that combines the share price and the volume to measure the cumulative flow of money into and out of a share. Chaikin [42] built on the MFI and developed what he called the cumulative money flow line, today referred to as the accumulation/distribution Index (ADI) [84]. Instead of using the TEP, Chaikin used a proportion of the closing price against the day's high and low price. Chaikin called this the money flow multiplier (MFM), expressed as follows:

$$MoneyFlowMultiplier_t = \frac{(ClosePrice_t - LowPrice_t) - (HighPrice_t - Close_t)}{(HighPrice_t - LowPrice_t)}$$

The MFM fluctuates between -1 and 1, where -1 means that the closing price is the same as the lowest price of the day and 1 means that the closing price is the same as the highest price of the day. The MFM is multiplied by the volume of the day and is known as the ADI. Because the ADI is an accumulation, the ADI is added to the previous day's ADI. ADI is expressed as follows:

$$ADIt = ADI_{t-1} + (volume_t \times MoneyFlowMultiplier_t)$$

A high MFM combined with a high volume shows strong buying pressure (inflow of capital) and pushes the ADI higher. Conversely, a low negative MFM combined with a high volume reflects strong selling pressure (outflow of capital) and drops the ADI. ADI is highly geared to level-off the close relative to the high-low range of the period. For example, a share price could fall dramatically and close significantly lower, but the ADI would still rise if the close is above the midpoint of the high and low share price of the period. Chaikin ignores the change in value from one day to the next.

The on balance volume (OBV) is a money flow index that focuses on the change in value from one day to the next. In 1946, Woods and Vignolia [206] created the OBV statistic. Granville [84] named the statistic OBV. The OBV gauges the market trend by observing the cumulative direction of the volume traded. Like the MFI, and ADI, the OBV also attempts to determine the flow of money in and out of the market. The OBV keeps a running total, and the volume is added or subtracted from this total depending on whether the daily share price is greater or less than that of the previous day's share price. If the running total is positive, the market is moving in an upward trend, with an inflow of cash. If the running total is negative, the market is moving in a downward trend, with an outflow of cash. The OBV

is expressed as follows:

$$OBV_t = OBV_{t-1} + \begin{cases} volume & \text{if } Price_t > Price_{t-1} \\ 0 & \text{if } Price_t = Price_{t-1} \\ -volume & \text{if } Price_t < Price_{t-1} \end{cases}$$

In addition to the inflow and outflow of money, technical analysts need to know the rate at which the share price is changing. The rate of change (ROC) is commonly referred to as a *momentum indicator*. The ROC is expressed as a ratio between a change in share price on a given day relative to the change in a share price of the previous day. The ROC is expressed as follows:

$$ROC_t = \frac{Price_t - Price_{t-n}}{Price_{t-n}} * 100$$

A share with a high ROC has high momentum and should outperform the market in the short-term. Conversely, a share with a negative or low ROC, and closing price below its SMA, is more likely to continue the negative trend and decline in value.

Each technical analysis function has several parameters that must be optimised by the analyst. Furthermore, each function must be used in conjunction with other functions to confirm the trend or change in the trend. The formulae above depict a sub-set of the available technical analysis functions. "Technical Analysis of Stock Trends" by Edwards, Magee and Bassetti [101] gives a detailed explanation of technical analysis functions used by traders.

Technical analysis is performed on stock data, currency data, or individual company share data. The technical analysis results are used to trade various trading products provided by exchanges. Each type of trading product has different levels of risk and different levels of cost. The next section explores three different trading products and the cost structures used during the empirical study.

## 2.4 Trading and Cost

This thesis focuses on stock listed on the JSE. The JSE provides access to its trading platform via partner companies. Partner companies expose different trading packages to their clients. One such trading partner is Standard Bank Group Ltd (SBK). SBK offers a simple, yet cheap investment package called Autoshare Invest [31]. Autoshare Invest offers investors a cheap alternative to direct online trading. Autoshare Invest exposes a sub-set of the most traded shares listed on the JSE. Trading is limited to specific days, the prices are quoted as the previous days' close, and purchasing is performed in bulk. The bulk purchase allows a trader to buy a percentage of a share, while another trader or the bank buys the rest. The bulk trading reduces fees. The reduced fees, fixed market price, and access to

a percentage of a share makes Autoshare Invest a simple, cheap platform for investors looking to hold onto shares for the long term. Because Autoshare Invest regulates the price, and fixes trading to specific days, daily trading is not possible.

The OST is a more advanced trading platform giving investors direct access to the JSE to buy and sell shares. All JSE listed shares are available to the investor, and an investor trades whenever the exchange is open. Share prices are quoted in real-time. The platform displays the current offers to buy and sell. Making the OST the ideal platform for daily trading. To facilitate daily trading, the online platform makes various products available to traders. The trading products include currencies, commodities, shares, and derivatives.

A derivative is a financial security with a value that is reliant upon or derived from an underlying currency, commodity, or share. The types of derivatives offered are dependent on the trading platform. Single stock futures, commonly referred to as *futures*, is an example of a derivative. Futures are contracts between two parties based upon the underlying stock or asset. One party agrees to buy the stock now and sell it to another party at a later date. The party buying the stock now is said to take the long position, while the party buying it later is said to take the short position. Generally, futures are purchased on credit and settled at the end. A futures price is the value paid to take part in the transaction.

To illustrate a futures contract, consider a share trading at R1 per share. A traditional share trader would spend R1 to purchase the share. Suppose that the share price increases in value by 10% and that the trader sells the share. Excluding costs, the trader would make 10c profit. If the share price dropped by 10%, the trader would have lost 10c, excluding costs. A futures contract returns far greater losses and gains. Assume that the future requires the payment of 1% of the underlying stock value to take part in the contract. The future consists of 100 shares. The total share value is R100. The cost to take part is R1.

Suppose Party A takes the long position and Party B takes the short position. Both Party A and Party B pay R1 to enter into the contract. Suppose that at the agreed contract termination date the share price has increased by 10% to R1.10. Party B must purchase the 100 shares from Party A for R110. Party A makes a profit of R9. Party B now owns the futures and paid R111 for stock worth R110.

Suppose that instead of increasing in value, the share price dropped by 10% to R0.90. Party B buys the 100 shares for R90, while Party A must now cover the credit and pay the difference making a total loss of R11. Party B paid R91 for shares worth R90. In both scenarios Party B paid more for stock than it was worth. However, Party B entered into an agreement to buy the shares at a later date, and therefore could have sold the shares not yet purchased at the start of the derivative contract. If Party B sold the shares not yet owned at R100 and bought them later at R90, Party B could have made a profit of R9.

The illustration shows that a trader can pay a fraction of the share cost and receive greater returns. This effect is known as *gearing*. Gearing is the ratio between a stock price and the stock value. Gearing is achieved through credit. Derivative products determine the gearing, the trade duration, the asset traded and the trading limits. Derivatives such as single stock futures are superior to regular share trading, because shares can be bought or sold for profit while the share price is going up or down. Derivatives are the preferred trading mechanism for daily market traders.

Derivative trading requires more than just a hold, buy or sell signal. It requires the holding, buying, or selling of a long or short position over a period. Due to the complexities of derivative trading, the traditional buy and sell share trading is used in this thesis.

Traditional share trading has the following advantages over derivatives:

- simple trading process of buy and sell,
- simple costing model, and
- no expiry of trades.

Traditional trading requires a trader to have the cash on hand to purchase a share. A trader can only sell a share if the trader owns the share. A trader is required to pay a number of fees when buying or selling shares. Trading fees in South Africa are determined by the South African Government, the JSE, and the trading platform provider.

Fees presented here are based on the 2008 fees of OST, which include:

- a headline brokerage fee of 0.5%, with a minimum of R80,
- security transfer tax of 0.25%,
- strate tax of 0.005787%, with a minimum strate of R11.58 and a maximum of R57.87,
- value added tax (VAT) of 14% (the study was done before the recent VAT increase to 15%)
- a Financial Services Board (FSB) fee of 0.0002%.

Buy and sell transactions incur the same fees, except for the security transfer tax which is payable on purchases and not sales. The headline brokerage fee is a minimum of R80, making it highly unlikely to return a profit on one share trading at R20 over the short term. A minimum number of shares must be traded to cover the transaction costs. Figures 2.6 and 2.7 illustrate the difference in trading fees when trading 100 or 1 000 000 shares in SBK respectively. The fee cost per share when buying 100 shares is R1.28, while the fee cost per share when buying 1 000 000 shares is R0.72.

Purchase and Sale of SBK Shares

| | | | |
|---|---|---|---|
| Number of Shares | 100 | | |
| Share Price | R84.50 | | |
| | | **Bought SBK Shares** | **Sold SBK Shares** |
| Share Value | | R8 450.00 | R8 450.00 |
| Brokerage Fee of 0.5% with a minimum of R80 | | R80.00 | R80.00 |
| Security Transfer Tax of 0.25% | | R21.13 | R0.00 |
| Strate Fee of 0.005787%, minimum of R11.58 and a maximum of R57.87 | | R11.58 | R11.58 |
| Insider Trading Levy (FSB) of 0.0002% | | R0.02 | R0.02 |
| Total Charges | | R112.72 | R91.60 |
| Value Added Tax (VAT) of 14% | | R15.78 | R12.82 |
| Total cost of the transaction | | R128.50 | R104.42 |

**Figure 2.6:** Cost structure of trading 100 SBK shares on the Standard Bank Trading Platform. The study was done before the recent VAT increase to 15%.

Purchase and Sale of SBK Shares

| | | | |
|---|---|---|---|
| Number of Shares | 1000000 | | |
| Share Price | R84.50 | | |
| | | **Bought SBK Shares** | **Sold SBK Shares** |
| Share Value | | R84 500 000.00 | R84 500 000.00 |
| Brokerage Fee of 0.5% with a minimum of R80 | | R422 500.00 | R422 500.00 |
| Security Transfer Tax of 0.25% | | R211 250.00 | R0.00 |
| Strate Fee of 0.005787%, minimum of R11.58 and a maximum of R57.87 | | R57.87 | R57.87 |
| Insider Trading Levy (FSB) of 0.0002% | | R169.00 | R169.00 |
| Total Charges | | R633 976.87 | R422 726.87 |
| Value Added Tax (VAT) of 14% | | R88 756.76 | R59 181.76 |
| Total cost of the transaction | | R722 733.63 | R481 908.63 |

**Figure 2.7:** Cost structure of trading 1 000 000 SBK shares on the Standard Bank Trading Platform. The study was done before the recent VAT increase to 15%.

## 2.5 Summary

This chapter explained stock market trading, stock market trends, and how supply and demand can drive the stock trend. Technical analysis was introduced as set of functions to determine the current and future market trend, based on current and historic stock market data. Technical analysts combine the results of some or all the functions described in this chapter into trading rules. Each trading rule is tailored to a specific share or market.

A trading rule is used to determine when to buy or sell a share. This study uses evolutionary algorithm (EA)s to evolve trading rules similar to the functions described in this chapter. The next chapter focuses on the concept of evolution, and simulated evolution known as *evolutionary computation*. Chapter 4 investigates how researchers used EAs to determine which of the technical analysis functions and parameters above are best suited for a specific share.

# Chapter 3

# Evolutionary Computation

*A guy said to me, "Yes, but the whole theory of evolution is based on a tautology: that which survives, survives" This is tautological; therefore it doesn't mean anything. I thought about that for a while and it finally occurred to me that a tautology is something that if it means nothing, not only that no information has gone into it but that no consequence has come out of it. So, we may have accidentally stumbled upon the ultimate answer; it's the only thing, the only force, arguably the most powerful of which we are aware, which requires no other input, no other support from any other place, is self-evident, hence tautological, but nevertheless astonishingly powerful in its effects." [sic]*

*-Douglas Adams (Science Fiction Author), 1998*

This chapter explores the concept of evolutionary computation. The chapter begins with Section 3.1 covering the history of evolutionary theory, more specifically the works of Darwin, Weismann, Mendel, and Morgan postulating on natural selection, germ theory, inheritance, and genetics respectively. Section 3.2 follows, describing the history of evolutionary computation and how researchers used the processes theorised by early evolutionary theorists to build computational models to solve optimisation problems. From the ideas defined in Section 3.2 grew a specific form of evolutionary computation known as genetic programming, which is described in Section 3.3. Section 3.4 describes multi-population evolution to enhance learning through a process known as co-evolution. The chapter concludes with a summary in Section 3.5.

## 3.1 Theory of Evolution

The Greek philosopher Anaximander [81, 100], lived in Miletus (Turkey) between c. 610–546 B.C. Anaximander noticed that the foetus of a human is similar to that of a fish. He reasoned that a fish could grow into all kinds of animals including a human. He stated that humans were born from a combination of different animals. Empedocles [41, 127] a Greek philosopher born in Akragas on the south-east coast of Sicily, Italy in 490 B.C. documented the similarity of species and concluded that all species came from the earth, and shared common traits. Anaximander and Empedocles concluded that in the beginning a small group of species gave birth to all the species on earth and overtime some species became extinct.

The French naturalist, mathematician, cosmologist Georges-Louis Leclerc, Comte de Buffon [76] believed the world was a few thousand years old and that species were created separately and organized into an unchanging hierarchy, with humans positioned just below the angels. Buffon believed that as creatures migrate the supply of "organic" particles that made up the creatures changed and so the creature would change, adapting to its environment. The proposal of ever-changing generations was shared by the French naturalist Jean-Baptiste Lamarck in his book "Philosophie Zoologique" [94].

Lamarck [94] proposed a systematic theoretical framework for understanding evolution where fluids etch out organs from tissue leading to ever more complex structures. Lamarck proposed that disused organs would transform or morph into more useful organs, and useful organs would continue to be enhanced. Overtime the organism would become more adapted to its environment.

Baron Cuvier [207] commonly known as the father of palaeontology rejected the morphology theory of Lamarck. Cuvier believed that function determined if a species continues to survive and that environmental events such as floods can kill off an entire species. While Cuvier did not believe in

evolution, his book "Le Règne Animal" [87], which compared the anatomy of the entire known animal kingdom provided convincing evidence for evolutionary change.

The English physician and poet Erasmus Darwin believed in evolutionary change. Darwin proposed in the chapter "Generation" of his book "Zoonomia" [90] that a species changed over time, and that each generation of the species is slightly different from the previous generation.

In 1837, the grandson of Erasmus Darwin, Charles Darwin [53, 68, 89, 111, 201] travelled to South America on board the H.M.S. "Beagle" to investigate the idea of a generation. It was on this voyage that Darwin realised that there are divisions in the distribution of the inhabitants of South America; those inhabitants on the east coast are different from those on the west coast.

Darwin realised that the past inhabitants of both the west coast and east coast share similarities with the present inhabitants. These observations led Darwin to formulate the concepts of modern evolutionary theory [89]. According to Darwin, Alfred Russel Wallace [111, 208], whom had been studying the Malay Archipelago had also arrived at the same conclusion [89].

Charles Darwin, in his book "On the Origins of Species" [89], alludes to a book on creationism titled "Vestiges of Creation". According to Darwin, the author of this book argues that animals, plants, and the like would just appear for no reason and be perfect. Perfectly appearing species did not, according to Darwin, explain the similar traits shared between species in the embryo, geographic distribution, and geological succession, first noted by Anaximander [81, 100] 2400 years earlier. Darwin states that species had not been created independently, but rather descended.

Darwin and Wallace [111] presented their theory of evolution to the Linnean Society of London in 1858 [89]. Darwin outlines the theory presented at the Linnean Society in Chapter IV of his book, "On the Origins of Species", which is as follows: Suppose that a species gives birth to several individuals and suppose that one individual has an advantage over the other individuals in the group. This advantage may make this individual smarter, faster, or stronger than any other individual in the group. Because this individual has an advantage, the conclusion is that this individual would live longer, and thus have more opportunity to reproduce. An individual that reproduces more, would naturally have more offspring; more offspring inheriting the advantage of the parent. It may be concluded that this new advanced offspring would live longer and have more children than the individuals that did not inherit the advantage.

As each generation develops, the individuals with the advantage become more, and the individuals without the advantage become less. Eventually, over several generations, all individuals in the population will have inherited the advantage. Darwin called this the theory of natural selection. Today, it is known as Darwinism.

The German evolutionary biologist, Friedrich Leopold August Weismann [209], confirmed the speculation of heredity and small changes to individuals proposed by Darwin and Wallace. Weismann published a paper in 1893 called "The Germ-Plasm. A theory of heredity". In his paper, Weismann proposed that genetic information is passed on through *germ cells*.

A germ cell contains half the number of genetic material of a regular cell. Regular cells are referred to as *Somatic cells*. A germ cell unites with a germ cell from the opposite sex to form a new individual. A germ cell from a specific sex is referred to as a *gamete*. A gamete from a male is known as *sperm*, and a gamete from a female is known as an *egg*.

Genetic material contains genetic information in the form of genes. Genes consist of nucleic acids and proteins that string together to form a thread-like structure referred to as a *chromosome*.

Weismann concluded that germ cells encode physical characteristics and not behaviour. When an organism produces germ cells, mutation takes place. This mutation is carried over to the somatic cells that are created by the genetic code within the germ cells. Weismann referred to the propagation of genetic information from germ cells to somatic cells as *heredity*.

Building on the concept of heredity, Gregor Johann Mendel [53], an Augustinian priest and scientist, conducted hybridisation experiments on garden peas (*Pisum sativum*). Mendel postulated two generalisations, which became known as Mendelian inheritance. Mendel's generalisations are the law of segregation and the law of independent assortment. Mendel described his laws in a paper, which he presented to the Natural History Society in 1865 called "Experiments on Plant Hybridisation" [167]. The first law of segregation says that "an individual produces gametes". Gametes contain one copy of an allele. An allele is a variant of a gene. Gametes are segregated from the parent organism. The second law is that of "independent assortment" or the inheritance law. When two gametes come together (egg and sperm) to form a new individual, the dominant phenotypes are selected with a higher probability than the recessive ones, and traits are inherited independently of each other. A phenotype is a set of observable characteristics of an individual.

An American geneticist and embryologist, Thomas Hunt Morgan [111, 168], aimed to discredit the works of Darwin, Weismann, and Mendel. However, through his studies of a small fly (*drosophila*), he confirmed the theories of selection and heredity. Morgan was awarded "The Nobel Prize in Physiology" in 1933 for the role that the chromosome plays in heredity [181].

Combining Darwinism, Weismann's theory of germ cells, Mendel's theory of inheritance, and Morgan's theory of genetics, the bases for a set of arguments known as the Neo-Darwinism are formed [111].

Neo-Darwinism is based on the premise that all life evolved from a primitive single cell organism.

This single cell organism evolved into complex plants and animals through a few processes, namely reproduction, mutation, competition, and selection [111].

The first of the processes in Neo-Darwinism, also known as *Darwinian evolution*, is reproduction. Reproduction is the process of replicating traits of a parent, or two parents, to their offspring. Reproduction is not replication. Only molecules can replicate, cells reproduce. Cells reproduce by replicating the molecules that make up the cell. Often, not all the molecules are replicated. This could be due to many factors, for example, there are not enough atoms present to make the molecule, the instructions to make the molecules are not complete, or the environment may not be conducive to the formation of the molecule. For whatever reason, it is important that reproduction is not replication. Mutation is the result of errors introduced during reproduction. Mutation is the second evolutionary process. Mutation is an important process as it introduces diversification into the structure of the cell. This diversification can affect the traits and functions of the cell.

Reproduction transfers the traits of the parents to their offspring, while mutation, a process during reproduction, can alter the trait in the offspring. The traits of an individual directly affect its probability of survival and reproduction. A finite set of resources such as food, or the right to reproduce exists. Organisms are required to compete for these resources. Traits of an individual affect the outcome of competition. If the food is fast, the fastest individuals catch more food. If the food is high, the tallest individuals reach more food. If the food is out at night, individuals that see the best at night find more food. If the predator of an individual has excellent eyesight, prey that blend into the surroundings the best, survive the longest. The longer an individual survives, the higher the probability that the individual will reproduce. Reproduction passes genetic material from parents to their offspring. Natural selection is the process of selecting parents for reproduction. Selection is the last of the four processes, and defines the rules or pressures for reproduction [111].

A trait is the key to the processes of reproduction, mutation, competition, and selection. Mendel [53] found that genes of an individual store traits or characteristics of the individual. A gene is the most basic unit of heredity. Weismann [209] proposed that germ cells store genes. During the creation of germ cells, genes are copied from the parent to the new germ cell. The copy is not always correct, resulting in altered genetic material or mutations. Because a gene is responsible for a specific trait, the altered gene results in a new or different trait. A new individual is formed by combining an egg cell and a sperm cell from the mother and father respectively. The newly formed individual has inherited traits from its parents, and different traits through mutation. Nature tests the traits; if the traits perform well within the individual's environment, the probability of the traits being selected for the next generation is much higher than that of traits that do not perform well. An organism's performance is a product of

the environment. Performance could be the length of time an organism lives for, the amount of food it consumes, or the appearance or smell of the organism. The performance influences the amount of reproduction an organism undertakes. The probability that the traits are selected for the next generation depends on the efficiency of reproduction, or the number of times that an individual reproduces.

## 3.2 The History of Evolutionary Computation

The previous section explored the works of Darwin, Wallace, Weismann, Mendel, and Morgan. These scientists pioneered our understanding of the evolutionary process in nature; a process that starts with a population of individuals, that create offspring, either through mutation of their own genetic material, or through re-combination of genetic material with another individual's genetic material, or both. The new offspring compete for survival and the right to become the parents of the next generation. The process continues with each generation becoming slightly better (based on the criterion of selection) than the previous generation [111, 112].

The idea that the process of evolution could be used to solve optimisation problems started about 60 years ago and has around 3000 papers published annually [112].

In 1932, Walter Bradford Cannon, an American physiologist, noted in his book, "The wisdom of the body", that evolution was a learning process and made a direct comparison to individual learning [112]. The famous computer scientist, crypto-analyst, logician, and mathematician, Alan Turing [112], once said that, "[there is an] obvious connection between [machine learning] and evolution".

Much of the early work in evolutionary computation has been forgotten or lost to science. Unlike the history of chemistry, physics, or mathematics, not much is known about the foundations of evolutionary computation [53, 112]. The earliest documented use of using a computer to understand evolution, was that of Barricelli [113]. In 1954, Barricelli performed computational experiments in evolution on an IBM704 at the Institute for Advanced Study in Princeton. Barricelli described his work in the paper, titled "Numerical testing of evolution theories: Part I Theoretical introduction and basic tests", which was originally published in Italian in 1954 and re-published in English in 1957 [62, 113].

Friedman and Fraser [53, 112, 121] performed some of the earliest evolutionary computation experiments. Friedman [121] published a 158-page Master's thesis in 1956 entitled "Selective Feedback Computers for Engineering Synthesis and Nervous System Analogy". An academic paper on the descriptions of a computational evolutionary process by Alex Fraser in 1957 [120], followed Friedman's thesis. The paper appeared in the Australian Journal of Biological Sciences, volume 10, titled "Simulation of Genetic Systems by Automatic Digital Computers. I. Introduction." [110, 112, 120]. Fraser

presented the case of diploid organisms represented by binary strings of a given length. Each bit in the string represented a gene. Fraser proposed the process of single-point crossover for binary string reproduction [110].

Single-point crossover selects an arbitrary point within the binary string. All data beyond that point in either organism's string is swapped between the two parent organisms [103, 112]. The resulting organisms are the offspring. Fraser extended the single-point crossover to $n$-point crossover in which each position along an organism's genetic string was assigned a probability of breaking for re-combination [112].

To maintain genetic diversity from one generation to the next, Friedman and Fraser introduced mutation. Mutation alters one or more genes within a chromosome from its initial state. A set of probability values determines if a gene undergoes mutation. Fraser studied varying effects of linkage, epistasis (interaction of genes that are not alleles), rates of reproduction, and additional factors on the rates of advance under selection, as well as the genetic variability of a population and other statistics [110]. Fraser's work was followed by articles by Friedberg and Friedberg *et al.* in 1958 and 1959, respectively [53]. Friedberg's work represents the earliest work in machine learning and describes the use of an EA for automatic programming.

In 1958, Bremermann [97, 112] presented a similar evolutionary computation model to that of Friedman and Fraser. Individuals were encoded as binary strings that were processed by reproduction, mutation, and selection to form offspring. Bremermann introduced the one-max problem, where fitness is determined by the sum of the number of 1$s$ within the binary string. Bremermann derived the generalised optimum probability of mutation for the one-max problem. In 1962, Bremermann [74] extended his original model to become a generalised function optimisation method.

American computer scientist John Henry Holland [135, 136, 137] and his colleagues at the University of Michigan formalised the EAs originally implemented by Bremermann, Fraser, Friedman and Friedberg *et al.* [53, 97]. Holland's emulation of evolution is now known as a genetic algorithm (GA), made up of a fixed length binary representation of a chromosome, similar to the work of Fraser and Bremermann. Holland and Fraser introduced the evolutionary operator of reproduction through the formation of a process known as *crossover*. Selection is done by selecting the top performing percentage of individuals. Quantification of individual performance is done by a mathematical function known as a *fitness function*. Holland published this complex adaptive system in 1975 in a book titled "Adaptation in Natural and Artificial Systems" [135], followed by two more books, "Hidden Order: How Adaptation Builds Complexity" [136] and "Emergence: From Chaos to Order" [137].

By the mid-1960s GAs had been used in many applications, such as tuning sets of weights in the

evaluation of fitness functions of game-playing programs [54, 60], evolution of chemical concentrations of biochemical systems [54, 184], pattern recognition [54, 80], amongst many others.

By the early 1970s, researchers turned their focus to better understand the behaviours of implementable GAs. Alternative selection and encoding strategies were experimented with, including elitism, rates for adaptive crossover and mutation [54, 80], and alternate selection and mating schemes [54, 139]. Aspects of GAs were documented at length, including the effects of genetic drift [118], the effects of population size, crossover, and mutation [53, 91]. The first workshop on GAs, the "Adaptive Systems Workshop", was organised in 1976 by several universities including the University of Michigan, the University of Pittsburgh and the University of Alberta [53]. By the 1990s variations of GAs had been developed to evolve complex, non-linear, variable-length structures such as rule sets, neural networks, and LISP code [53].

Fraser, Bremermann, and Holland simulated abstracted chromosomes and the modification thereof. Selection criteria focused on the chromosome, to determine the parents of successive generations. The focus on the hereditary information is known as *genotypic evolution*. Lawrence Fogel [116] began researching alternative methods for simulating evolution as a phenotypic process. A phenotype is an organism's actual observed properties such as behaviour (actions), development (regeneration), or morphology (structure).

In 1962, Fogel began experimenting with simulated evolution of finite-state machines [53, 116]. A finite-state machine is an abstract machine that can be in exactly one of a finite number of states at any given time. A transition moves a finite state machine from one state to another. Each transition requires external inputs and conditions for the transition to occur. A finite-state machine is a mathematical model of computation. Fogel focused on the behavioural link between parents and offspring, as opposed to the genetic link. Fogel experimented on mutation of finite state machines to forecast non-stationary time series data.

The finite-state machine had to predict a 4-symbol output sequence of a Markov process driven by random noise. A Markov process is defined as a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event [35]. In Fogel's simulation, a randomly mutated copy of a parent machine becomes an offspring machine. A probability distribution determines the rate of mutation. A fitness function measures the error rate of the finite-state machine in predicting the Markov sequence. The parents and offspring are ranked by their error measure from the least errors to the most errors. The top half performing machines are retained as parents to generate children. The population size remains constant. Fogel called this process *evolutionary programming* [115, 116].

In 1964, Fogel published a dissertation titled "On the organisation of intellect" [114]. In 1966, a book, "Artificial Intelligence through Simulated Evolution" [116], co-authored by Alvin Owens and Michael Walsh followed Fogel's dissertation. "Artificial Intelligence through Simulated Evolution" is regarded as the first book published on evolutionary computation. In 1970, Fraser and Burnell published the second book on evolutionary computation titled "Computer models in genetics" [112, 119]. Additional experiments by Lutter and Huntsinger [159], Burgin [77], Atmar [49], Dearholt [95], and Takeuchi [202] followed the approach of Fogel. By the mid-1980s, the evolutionary programming (EP) process was extended to include alternative representations such as ordered lists, vectors, and neural networks. In 1992, the first formal conference on EP was held. The conference included applications such as path planning [53, 109, 156, 172], robotics [45, 53, 165], artificial neural network (ANN) design and training [53, 164, 176, 192], automatic control, and other fields [53].

At around the same time that Fogel was experimenting with simulated evolution, researchers in Berlin were also experimenting with simulated evolution, focusing on phenotypic evolution. Bienert, Rechenberg, and Schwefel [53, 68] began research on a robot that performed a series of experiments on a flexible slender three-dimensional body in a wind tunnel to minimise its drag. At first Binert *et al.* tried two techniques manually. The first was manually optimising one variable at a time, the second approach was manually optimising all the variables using a mathematical optimisation technique known as *discrete gradient*. Both approaches failed, because variables that affected the shape of the object were linked. Changing one variable affected the other variables.

Rechenberg decided to change the variables randomly. The first version of their algorithm was called $(1+1)$ and it was a type of EA referred to as evolutionary strategies (ES). One parent and one descendant per generation were tested for the lowest drag. The best individual was selected as the parent for the next generation. Binomially distributed mutations updated the variables of a parent to form a child. While the $(1+1)$-ES performed better than the manual optimisations, the strategy became stuck prematurely. The strategy could not evolve a better solution beyond the stuck solution. Schwefel changed the algorithm from binomially distributed mutations to normal distributed mutations, which performed significantly better than the previous implementation. In 1973, Rechenberg extended the $(1+1)$-ES to include multiple individuals as a population, and called the algorithm $(\mu+1)$-ES. Until now, the mutation step sizes were fixed. If the step size was too large, the evolutionary process would update the variables to a number greater or less than the optimal. If the step sizes were too small, the $(\mu+1)$-ES could prematurely converge. Rechenberg and Schwefel changed their algorithm to include the ability to evolve the mutation step size. This new self-adapting $(\mu+1)$-ES was named $(\mu+\lambda)$-ES. Schwefel used a $(\mu+\lambda)$-ES to optimise the shape of a three-dimensional convergent-divergent nozzle

for maximum energy efficiency. The technique was improved by Klockgether and Schwefel in 1970 [53]. In 1975, Schwefel [191] introduced multi-cellular individuals into an ES for binary optimisation. This process introduced several sub-populations and niching mechanisms for global optimisation. A wide range of ESs exists.

From the 1960s to the late 1980s, the three branches of evolutionary computation evolved independently of each other [53]. Comparisons between the three branches, ESs, GAs and EPs are documented by Bäck *et al.* [52, 53, 54, 55, 56, 57, 58, 59]. It was Bäck [52] and Bäck *et al.* [53] who introduced a common algorithmic schema for all brands of EAs.

In 1991, an international workshop titled "Parallel Problem Solving from Nature" at Dortmund attempted to provide interaction amongst the various evolutionary computing research groups [53]. More co-operation and interaction workshops followed this, namely:

- the International Conference on GAs in San Diego, CA, USA, July 1991;
- the Evolutionary Programming Conference in La Jolla, San Diego, CA, USA, 1992; and
- Parallel Problem Solving from Nature in Brussels, Belgium, September 1992.

The increased interaction led to a consensus for the name of this new field, namely evolutionary computation (EC). In 1993, a journal with the same name was established and published by MIT Press. The first formal conference on EC was held from the 27$^{th}$ to the 29$^{th}$ June 1994 in Orlando, Florida, USA. The Institute of Electrical and Electronics Engineers (IEEE) held three dedicated streams at the "IEEE World Congress on Computational Intelligence" in 1994 namely fuzzy systems, ANNs and EC. The IEEE conference led to the publishing of an organised EC handbook to provide a more cohesive view of EC: the "Handbook of Evolutionary Computation" was first published in 1997 by IOP Publishing Ltd and Oxford University Press [53].

The Genetic and Evolutionary Computation Conference (GECCO) was the first conference to focus solely on EC. GECCO was held in 1999 as a combination of the International Conference on Genetic Algorithms (ICGA) and the Annual Genetic Programming Conference (AGPC). GECCO is the main annual conference of the special interest group on genetic and evolutionary computation, which is a special interest group of the Association for Computing Machinery (ACM).

## 3.3 Introduction to Genetic Programming

Holland formalised GAs by defining that a GA is a meta-heuristic algorithm that simulates the Neo-Darwinism process of evolution and natural selection. A GA has four evolutionary operations, namely

reproduction, mutation, competition, and selection [111]. This evolutionary process is described in the pseudo code listing in Algorithm 1.

**Algorithm 1:** Basic evolutionary algorithm process [103, 111].

$g = 0$;
Initialise the initial population $C_g$;
Evaluate the fitness of each individual $C_{g,n}$ in population $C_g$;
**while** not converged **do**
    $g = g + 1$;
    Select parents from $C_{g-1}$;
    Re-combine selected parents through crossover to form offspring,
        which is added to $O_g$;
    Mutate the offspring in $O_g$;
    Evaluate each individual in $O_g$;
    Select a new population from $C_{g-1}$ and $O_g$ to form $C_g$;
**end while**
**return** the best individual in $C_g$ as the solution;

The initial population of a GA is created stochastically. A population is made up of individuals, with each individual having a "chromosome". Holland defined the chromosome as a binary string known as the *genotypic instruction set*. The chromosome tells the genetic algorithm how this individual will behave within its programmed environment, referred to as the *individual's phenotypic behaviour*. The phenotypic behaviour of the individuals within the population are quantified using a defined fitness function.

Each gene within the chromosome represents a value within the problem's solution space. Genes are sampled from the problem's solution space during initialisation. The problem's solution space is also known as the *search space*. Each individual represented by a chromosome defines one possible solution within the search space. Initialisation is done as a uniform sampling over the search space to ensure that the solutions are diverse and cover a greater area of the solution space. The more individuals, the more solutions representing more of the search space. A fitness function is used to quantify the performance of a solution within the search space.

Parents are probabilistically sampled from the population of individuals. The selected parents undergo reproduction using a crossover operator to form offspring. Offspring combine genetic material from both parents. The offspring are evaluated by the fitness function. A selection function based on a

probability proportional to the individual's fitness value determines which of the individuals from the previous generation and offspring are selected as individuals of the next generation.

The offspring are made from combinations of genetic material derived from the parental group, and the parental group is smaller than the population size. For these reasons, each individual within each new generation becomes more similar to each other than the previous generation's individuals are to each other. Once a certain number of individuals are similar, it is said that the process has *converged*.

The smaller the population size, the quicker the convergence speed. A smaller population has a smaller sample of the search space, meaning that the most optimal solution has a higher probability of being missed. To find the optimal solution, a smaller population might require more generations, but convergence may cause the population to be so similar sooner, that the most optimal solutions are missed. When this happens, it is said that the population has converged on a local optimum.

Mutation changes genes within a chromosome. Altering the genes within an individual results in more diversity, because the individual is no longer similar to its parents. Mutation reduces convergence speed by introducing diversity. Offspring undergo mutation by applying a mutation operator probabilistically. The higher the probability of gene mutation, the more genes are altered and the greater the difference between the offspring and their parents. The lower the mutation probability, the more similar the offspring are to their parents.

The new offspring carry some traits inherited through reproduction, and some offspring have new traits generated through mutation. A new generation is selected from the parents and the offspring. The process continues until a given number of generations have elapsed, a sufficient solution to the optimisation problem is found, or until all the members of the population are so similar that no further generations would yield a better result.

Several implementations of a GA exist. The original GA implementation, called the canonical genetic algorithm (CGA) [53] as defined by Holland [53], had distinct characteristics: a chromosome is represented by a fixed and finite length bit string. Each bit in the string represents a gene. A gene is one of two alleles, a 0 or a 1. Holland used a 50% probability of randomly selecting individuals and copying them to a parent pool. Individuals within the parent pool are randomly paired as parents. Single-point crossover is used as a reproductive method [103, 126]. A crossover point is randomly chosen and the strings are swapped with respect to the crossover point between the two parents to form offspring. After crossover, a probability distribution determines which of the bits within the chromosome of the offspring are flipped. The random alteration of bits within the individual simulates mutation. Holland used a proportional selection operator, known as *rank selection*, to sample individuals from the previous generation and offspring as individuals of the new generation. Proportional selection samples

individuals with a probability proportional to the individual's fitness [103, 126].

A GP is a specialisation of a GA where the individual's chromosome is encoded using a *tree* structure. A tree structure is a random length, non-linear, hierarchical abstract data structure. A tree structure is able to encode logical formulae, arithmetic formulae, or computer programs. Each tree represents a symbolic expression defined by a tree grammar.

The tree consists of *branch nodes* and *leaf nodes*. Leaf nodes have no children and implement a number or constant from a *terminal set*. A leaf node is also known as a *terminal node*. A branch node is connected to other branch nodes or leaf nodes and implement an operator from a *functional set*. A branch node is also known as a *non-terminal node*.

The first documented use of a GP was in 1981 by Richard Forsyth [53, 117]. Forsyth did not call his algorithm a GP but rather biological evolutionary algorithm generating logical expressions (BEAGLE). Forsyth screened client medical data to determine the probability of patient survival [117]. Patients were measured against 18 variables. BEAGLE had to evolve a rule, using a bespoke language created by Forsyth. The language had 13 operators, including arithmetic equality, arithmetic inequality, greater than, less than, greater than or equal to, less than or equal to, logical disjunction, logical conjunction, negation, addition, subtraction, multiplication, and division [117]. A combination of the operators determined the rule. A rule results in a boolean value where `true` represents survival and `false` represents death. Forsyth did not know in advance what the length of the optimal rule would be. Forsyth used a tree structure to store the evolved logical expression [117]. The interpretation of the tree structure was critical to the success of BEAGLE. Interpretation of a tree requires two considerations, namely tree traversal and a tree grammar.

Forsyth used an in-order traversal of the tree [117]. The order of operation is important to BEAGLE and to illustrate the order, each walk is encapsulated with brackets. Using the example in Figure 3.1 and adding brackets to an in-order traversal yields the following result: ((4 (2) 5) 1 (3)). Addition of brackets maintains the structure of the tree. Examine node (2), which is encapsulated by node (4) and node (5). This is the same as what the tree represents: (4) and (5) are children of (2).

The second important consideration of a tree structure is its grammar. Forsyth created his own language of 13 operators [117]. The operators together with the patient variables form a rule. Forsyth's operators require variable input. Some operators required numerical input, while others accepted boolean input. This constraint meant that terminal nodes of the tree, (3), (4), and (5) in the example could not be an operator, and had to be a variable.

The variables are one of 18 measured patient variables. If the parent node required a boolean variable, then the child node had to be a boolean type. Forsyth's operators resulted in different output types.

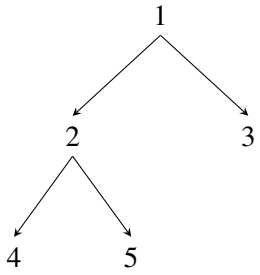



**Figure 3.1:** Simple binary tree structure, depicted here to illustrate a tree traversal. A pre-order traversal returns 1 2 4 5 3. A post-order traversal results in 4 5 2 3 1. An in-order traversal results in 4 2 5 1 3. A top down level order traversal results in 1 2 3 4 5. A bottom up level order traversal results in 4 5 2 3 1.

For example, the operator multiplication requires two numerical inputs, and the result is a numerical output. The operator, arithmetic inequality requires two numerical inputs and the result is a boolean output. In the example, ((4 (2) 5) 1 (3)), the operator stored in node (2) is applied to the operands stored in nodes (4) and (5), resulting in an operand for node (2). The operator (1) is applied to the result of (2) and the value of (3). From the example, the root node is the last node to be executed. Because the rule requires a boolean result, the root node must result in a boolean value.

The tree encoding requires rules, defined as syntax. Syntax defines the combinations of symbols that represent operators and variables. Syntax does not provide any information about the meaning of the rule or the results of executing that rule. The semantics focus on the meaning of the rule, its operators, and the results of executing the rule. Together, the syntax and semantics are defined as the grammar. Grammar is application specific.

Forsyth's implementation used a bespoke language as the grammar in BEAGLE and he suggested that LISP be used in future [155]. LISP was created by John McCarthy [163] in 1958. LISP is the second-oldest high-level programming language. In 1985, Nichael Cramer experimented with GPs, and also used a bespoke grammar [86]. In 1992, John Koza [150] used LISP as a grammar for his experimentation in evolving programs with GPs.

Koza [53, 111, 112, 150] used GPs for a number of very diverse optimisation problems, some of which included symbolic integration, discovery of reusable programs, automated synthesis of analog electrical circuits, automated synthesis of metabolic pathways, optimisation of antennas, and evolving a controller to balance a broom on moving carts [148, 149, 150, 151, 152, 153].

Forsyth, McCarthy, and Koza used GPs to evolve solutions to a variety of problems. Each problem space requires a defined grammar and tree traversal. To illustrate why tree traversal and a grammar are required, suppose a GP is tasked to evolve a set of instructions for a simple robot to complete a task,

and another GP is tasked to find a mathematical formula to a problem. The same EA is implemented. However, the tree traversal and tree grammar are different.

Consider the optimal program to control the robot as listed in Algorithm 2. The operation goForward has a dependency on the operation multiply. The result of multiply is used by the operation goForward. The operation turnRight is executed once the operation goForward is complete. Each operation is executed in sequence, and an operation may consist of sub operations. The output of a sub operation is the input of an operation.

To encode this simple program into a tree structure requires two considerations: a grammar and tree traversal. The robot controller grammar is a bespoke language invented to execute robot commands such as go forward or turn right. The tree traversal is dependent on the tree structure, and how the program is extracted from the tree to ensure that the grammar is correct. Figure 3.1 depicts a binary tree, where each sub-tree has at most two children. Algorithm 2 is transcribed to a binary tree structure in Figure 3.2. To correctly extract the program from the binary tree, a pre-order traversal is used. First, the root node is visited and then goForward, multiply, 4, 100, turnRight, 90, goForward, 200 and then the grammar is required, so that the executor of the program understands that the operator multiply has two variables, 4 and 100, and that the result of multiply is the parameter of goForward.

For the example tree in Figure 3.2, the operators are encoded such that any node to the left is a child, while any node to the right is its sibling. The nodes (2) and (4) are siblings, and are children of node multiply. Multiply is a child of goForward. GoForward, turnRight and goForward are siblings. An alternative approach to a binary tree is to use a multi-way or k-ary-tree. A k-ary-tree is a tree with an unlimited number of direct descendants. Figure 3.3 is an example of the algorithm in Algorithm 2 transcribed into a k-ary-tree. In this example, each level denotes a sibling. The tree is walked in a pre-order traversal. The robot control program is different from the Forsyth's rule problem described previously, in that operations must be executed in sequence.

**Algorithm 2:** An example of simple procedural robot instructions.

```
Start;
    goForward(multiply(4,100));
    turnRight(90);
    goForward(200);
```

An alternative to the robot program example, is that of a mathematical function. Mathematical operations are executed using the defined mathematical order of operations, namely brackets, exponents,

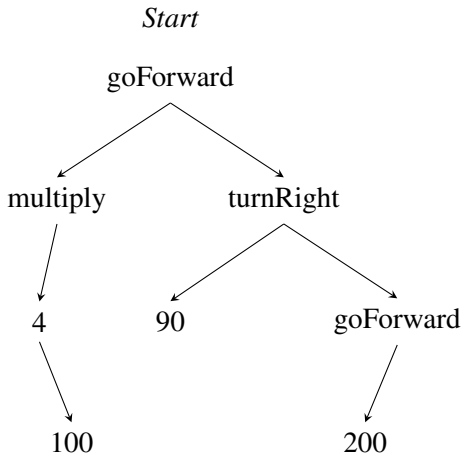



**Figure 3.2:** Simple procedural robot instructions encoded as a tree. A pre-order traversal is used to correctly extract the instructions such that order is preserved.

roots, multiplication, division, addition, subtraction, and left to right. The grammar of a mathematical expression is defined by the laws of mathematics. The expression is encoded within a tree structure, and the tree must be traversed in a way that the expression remains grammatically correct. The mathematical expression $(x+1)-4$ is encoded as shown in Figure 3.4. The first node in the tree represents the binary operation subtraction, its children are addition, and the number four respectively. The number of children a node has corresponds to the arity of the operand/operator. The addition binary operator has the unary operands $x$ and 1, while 4 has no children. An in-order traversal extracts the expression from the tree structure. Wrapping each extraction with brackets preserves the execution order. Traversing the tree in Figure 3.4, using an in-order traversal and adding brackets to each traversal results in the following mathematical expression: $(((x)(+)(1))(-)4)$; the result is simplified to form $(x+1)-4$.

From the two examples it is clear that the problem determines the type of tree structure, the grammar, and the traversal path. A GP is a GA with the exception that the internal chromosome structure is different. Because the structure of a GP is different, the crossover and mutation operators have been modified to work for tree-based representations.

Holland's GA uses single-point crossover as a reproduction operator [135]. Single-point crossover selects an arbitrary point within the binary string chromosome of each parent. All data beyond that point in either parent individuals' string is swapped between the two parent individuals [103, 112]. This works when applied to a binary string, as there is no semantic meaning associated with genes. All genes share the same possible variations, namely a 1 or a 0.

A GP uses a tree structure that conforms to a grammar. Swapping nodes arbitrarily can invalidate the

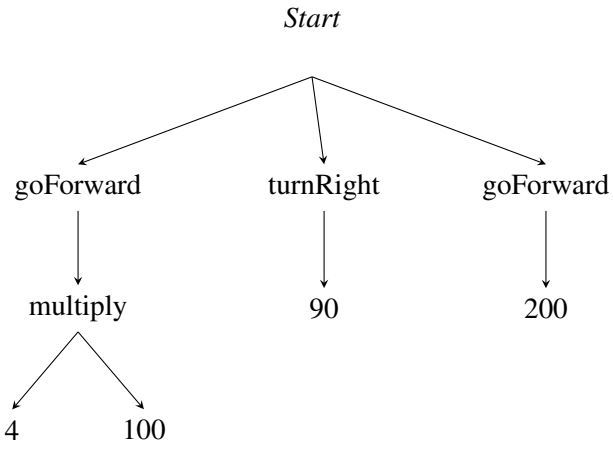



**Figure 3.3:** Simple procedural robot instructions encoded as a k-ary-tree. A pre-order traversal is used to correctly extract the instructions such that order is preserved.

syntax and semantics. Consider the mathematical expression $(x+1)-4$, the number 4 can be replaced by any other operand, or complete operation that results in an operand. Thus, 4 can be replaced by a number, for example 3, because $((x+1)-3)$ is valid. The number 4 can not be replaced by an operation, for example $\times$ because $((x+1)-\times)$ is invalid. However, 4 can be replaced by a complete operation, i.e. $3\times 2$ because $((x+1)-(3\times 2))$ is valid. Alterations within a chromosome must comply with the grammar defined for the problem solution.

To perform crossover between two trees, a node (A1) is selected at random from the first tree. A sub-set of nodes are selected from the second tree, such that the nodes selected are valid grammatical replacements for the node selected in the first tree. A random node is then selected from this sub-set. The sub-trees of the selected nodes are then swapped. An example of crossover using two trees is illustrated in Figure 3.5. Node (4) from the left parent is selected at random. Node $(-)$ from the right parent is selected at random. The nodes and their sub-trees are then swapped to form two offspring trees.

The GA implemented by Holland used a binary string [135]. Each element in the string is either a 1 or a 0. Mutation is performed by flipping a bit from one state to the other. Holland [135] assigned a small mutation probability to each gene within a chromosome. A small probability results in a few altered genes or none at all. Assigning a mutation probability to individual genes is referred to as micro-mutation [145].

Micro-mutation on a fixed length structure results in equal mutation probability across all individuals. A variable length structure such as a tree results in a higher probability of larger trees undergoing

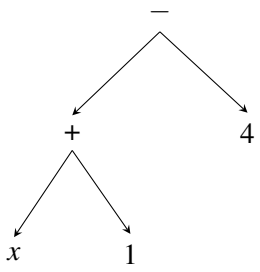



**Figure 3.4:** Example of a simple mathematical expression encoded as a binary tree. Using an in-order traversal and adding brackets to each traversal results in the mathematical expression: $(((x)(+)(1))(-)4)$, which is simplified to $(x+1)-4$.

mutation.

Jones [46, 145] introduced a macro-mutation called headless chicken. Headless chicken generates offspring by recombining one parent individual with a new randomly generated individual. Jones assigned the probability of mutation to the entire individual and not specific genes. Macro-mutation ensures that all individuals in the population have the same probability of undergoing mutation regardless of chromosome size.

Like crossover, mutation must ensure that altering a gene results in a grammatically correct tree. Several mutation operators have been developed for GPs. Mutation operators are dependent on the representation grammar. The most frequently used mutation operators are discussed below with reference to Figure 3.6 [103] (Figure 3.6(a) illustrates the original individual before mutation):

- **Root node mutation**: Depending on the tree grammar, the root node might require a different node to a function or terminal node. Root node mutation changes the root node to a randomly selected valid node. Figure 3.6(b) illustrates that the node $+$ is replaced with a node $\times$.
- **Function node mutation**: A functional node is randomly selected and replaced with a randomly selected functional node of the same arity. Figure 3.6(c) illustrates that function node $-$ is replaced with function node $\div$.
- **Terminal node mutation**: A terminal node is randomly selected and replaced with a randomly selected terminal node. Figure 3.6(d) illustrates that terminal node $z$ is replaced with terminal node $y$.
- **Swapping mutation**: A functional node, or root node is randomly selected. The children of the randomly selected node are swapped, provided the result is semantically correct, e.g. division by

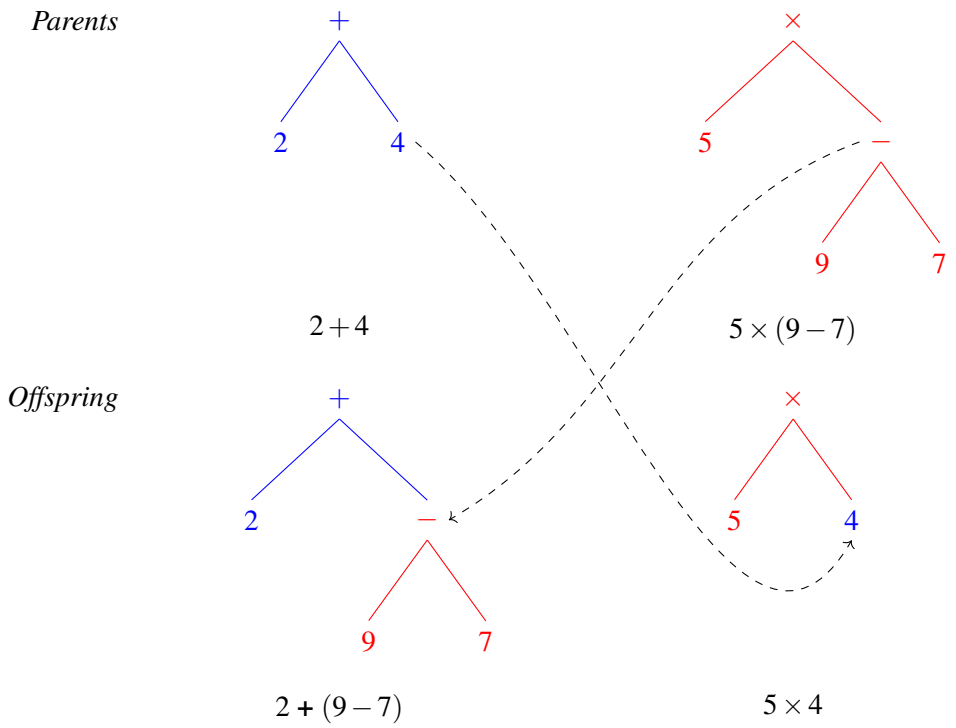



**Figure 3.5:** An example of GP crossover using two trees. Crossover is dependent on the grammar of the tree.

zero is not allowed. Figure 3.6(e) illustrates that function node $-$ is selected and its children are swapped.

- **Grow mutation**: A terminal, or functional node is randomly selected and replaced by a randomly generated tree. Figure 3.6(f) illustrates that function node 2 is replaced by a new sub-tree.
- **Gaussian mutation**: A terminal node which represents a constant is randomly selected and mutated by adding Gaussian noise to that constant. Figure 3.6(g) illustrates Gaussian mutation
- **Truncation mutation**: A functional node is randomly selected and replaced by a randomly selected terminal node. Figure 3.6(g) illustrates that function node $-$ is replaced with terminal node $a$.
- **Headless chicken mutation**: A parent is selected and recombined with a randomly generated tree. Recombination is done using crossover.

The examples of tree structures, traversals, mutation and crossover presented are problem specific. Each problem has a discrete search space defined by the grammar. The grammar determines the possible

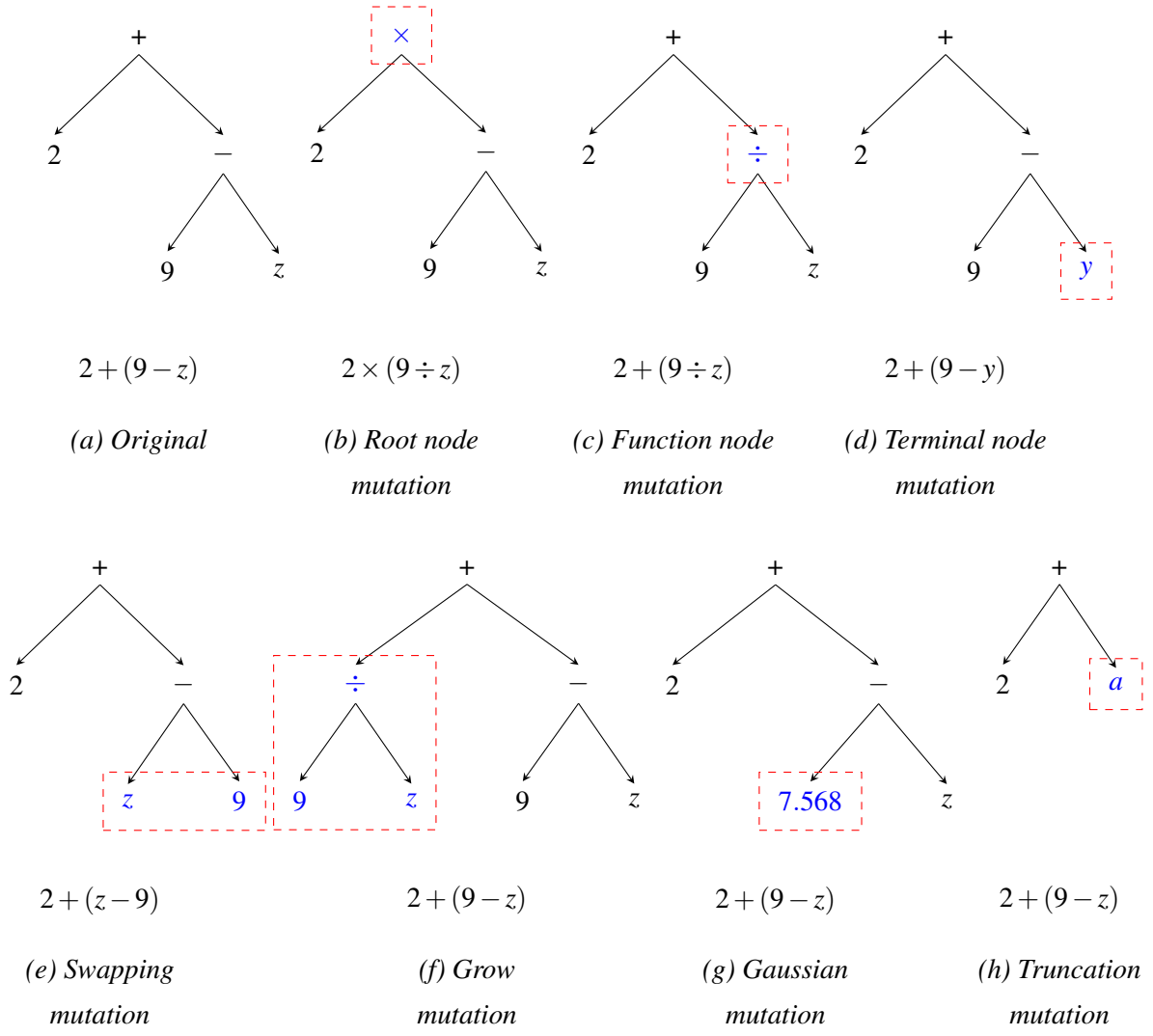



**Figure 3.6:** Various examples of GP mutation strategies are depicted. Mutation is dependent on the grammar used.

tree structures, and how the tree is traversed. The grammar determines possible mutations and valid crossover points.

Before the evolutionary process can begin, an initial population must be created. Initialisation of the first generation is done by stochastically generating individuals. Population initialisation for a GA is simple, because the fixed length binary string is initialised to random values of 0 or 1. The initialisation of a GP is problem specific and is dependent on the grammar. A valid root node is required, followed

by valid children nodes. The process is repeated until either the tree is complete with valid terminal nodes created stochastically, or a desired maximum tree depth is reached and terminal nodes are forced. This leads to variable tree depths. Tree size is also problem specific.

The tree represents a model of the observed dataset. If the model corresponds too closely or exactly to the dataset, the model may fail to reliably predict future observations; the model is said to over-fit [26]. The opposite of over-fitting is under-fitting. Under-fitting results in failure to model observed data. A tree must contain enough nodes to accurately model generalised observations in the data. If the tree is too small it will not have enough nodes to model the observations and will under-fit. If the tree is too large it will begin to model all the observations and begin to over-fit. Limiting the size of the trees must be considered.

The crossover operator can exponentially grow a tree in a GP. For example, a child connected to the root with a sub-tree of 10 descendants could swap with the lowest node in another individual of the same size, doubling the depth of one offspring and decreasing the depth of the other. The crossover operator could be altered to only crossover small sub-trees, slowing down the growth of the tree. Reducing the tree size of the individuals in the initial generation can delay the generation of large offspring through crossover. Neither approach stops the generation of large offspring it only delays the creation of large offspring.

The mutation operator grows a tree by selecting a terminal node and replacing it with a new sub-tree. The rate of growth is dependent on the size of the new sub-tree. Reducing the size of the newly generated sub-tree reduces the growth rate of the mutated tree. Decreasing the probability of replacing a terminal node with a new sub-tree reduces the growth rate. The prune mutation operator replaces a non-terminal node with a terminal node, decreasing the tree size. Increasing the probability of pruning delays the creation of large offspring.

Mutation and crossover inevitably produce more complex models as evolution continues leading to greater probability of over-fitting. Several approaches have been proposed to avoid over-fitting. These include reserving part of the in-sample data set as a validation set on which to test the performance of the solutions, increasing the amount of in-sample data, dynamically decreasing the probability of mutation, and penalising model complexity [43, 130].

The selection pressure determines which individuals become parents and which individuals are carried to the next generation. Selection pressure is determined by a probability based on a fitness measure. The fitness measure is determined by a fitness function which can include a penalty function that penalises larger trees.

Three degrees of penalty functions exist [196]: barrier penalty functions in which no solution is

considered that exceeds the maximum tree size, partial penalty functions in which a penalty is applied close to a maximum tree size, and global penalty functions which are applied across all individuals. A barrier penalty function results in a fitness measure that returns a probability of zero, while the partial and global penalty functions decrease the probability as the tree size increases. Increasing selection pressure on smaller trees could result in premature convergence on a suboptimal solution. Care must be taken to ensure that the penalty value is large enough to favour smaller trees, but not too large to inhibit exploration.

Holland's CGA randomly selected individuals to become parents of the next generation [53]. The individuals that comprise the next generation are sampled from both the parents and the offspring. A selection operator determines which individuals are carried over and which are discarded.

CGA used rank selection to sample individuals from the previous generation and the offspring to form the new generation. Rank selection uses a relative fitness value derived from the absolute fitness. The absolute fitness of the previous generation and offspring are calculated using the fitness function. The previous generation and offspring are then ordered by their fitness value. Once sorted, a rank is assigned in ascending order. The worst individual is assigned 1, the second worst 2, and so on until the best individual has the rank $n$, where $n$ is the total number of individuals. A probability of selection is assigned to each individual using:

$$p_i = \frac{Rank_i}{\sum_{t=1}^{n} Rank_t}$$

where $i$ is the individual.

During the initial generations, absolute fitness values between individuals can differ greatly, which results in a select few individuals dominating the selection process. Reusing the same dominant individuals for crossover results in premature convergence. Because rank selection assigns a rank as a relative fitness, the difference in absolute fitness has no bearing on the selection probability, thereby reducing selection pressure. Therefore, rank selection avoids premature convergence by relaxing the selection probability on individuals that have greater fitness than others, and increasing the selection probability for individuals with low fitness. Conversely, when fitness is similar amongst individuals, which occurs in later generations, small differences in fitness are amplified. To illustrate the amplification effect absolute fitness has on selection probability, consider five individuals with fitness values of 80, 6, 4, 2, and 1. After ranking, the fitness values become 5, 4, 3, 2, and 1 respectively. The selection probabilities are 0.33, 0.266, 0.2, 0.13, and 0.06. Should the fitness of a later generation of individuals become 80, 79.998, 79.988, 79.786, and 79.635, the selection probabilities remain the same at 0.33, 0.266, 0.2, 0.13, and 0.06, because there was no change in the ranking.

Since the initial experiments of Holland, alternative approaches to selection have been implemented for EAs. The following is a sub-set of the most frequently used sampling strategies:

- Random selection assigns equal selection probability to all the individuals regardless of their fitness value [103].

- Proportional selection is similar to rank selection, except that the absolute fitness value is used and not the relative fitness. In proportional selection, the probability of an individual being selected is proportional to the absolute fitness value. Many proportional selection algorithms exist.

  A common proportional selection algorithm is known as roulette wheel selection [103]. Roulette wheel sampling normalises the fitness values, usually by dividing each fitness value by the maximum fitness value [103]. The analogy to a roulette wheel can be envisaged by imagining a roulette wheel in which each individual represents a pocket on the wheel; the size of the pocket is proportional to the probability of selection of that individual. While the wheel is spinning, the ball has a greater probability of landing in a larger pocket than a smaller one [103].

  Stochastic universal sampling (SUS) is an alternative proportional selection strategy introduced by Baker [61]. Each individual is placed next to each other on the circumference of an imaginary circle. Individuals are placed in order of their fitness value. The amount of space an individual takes up on the circumference is directly proportional to the individual's fitness value. The greater the individual's fitness, the greater the amount of space allocated to the individual. SUS uses a random number to determine a position on the circumference. A fixed distance equal to the circumference divided by the number of individuals is used to mark positions on the circumference. Each individual occupying a marked position is selected. Individuals can be selected multiple times.

- Tournament selection randomly selects $k$ individuals from the population. The selected individuals take part in a tournament and the individual with the best fitness value is selected [103]. An advantage of tournament selection is that the worst individual of the population will never be selected for reproduction provided that $k$ is greater than one. If $k$ is equal to one, then the selection process is random and the worst performing individual could be selected. If $k$ is the same as the population size, the best individual will always be selected [103].

- Elitism selects the best $k$ individuals. An advantage of this approach is that the worst individuals are never selected. However, the best individuals dominate the selection process leading to rapid

convergence. Elitism is the same process as tournament selection with a $k$ value equal to the population size.

As per the pseudo code listing in Algorithm 1, the CGA randomly selects individuals as parents and then sampled individuals from the previous generation and offspring to form the next generation. This approach is preferred, because individuals from the previous generation that represent good solutions have a probability of moving to the next generation. However, depending on the sampling strategy, good solutions may dominate and lead to rapid convergence. An alternative approach is to carry only the offspring to the next generation, discarding the previous generation entirely. This approach slows down convergence, but good solutions are lost. A hall-of-fame can store the best individuals from each generation to preserve good solutions.

This thesis uses the approach of Holland, using both the previous generation and the offspring as a source for the next generation. A hall-of-fame is revisited in the next section.

## 3.4 Co-evolution

Generations of species rarely evolve in isolation, but rather co-evolve with other species. The word "co-evolution" was coined by Ehrlick and Raven in 1964 [185] in their description of the probable influences that plants, herbivores, and insects have had on each other's evolution. Brooks and McLennan [185] further subdivided co-evolution into co-speciation, referring to "mutual phylogenetic association", and co-adaptation, referring to "mutual modification". Co-speciation is the process whereby one population evolves in response to, and in concert with another, and is a consequence of the associate's dependence on its host for its survival [33]. Co-speciation is referred to as co-operative/symbiotic co-evolution. Co-adaptation is the reciprocal adaptation of two or more genetically determined features through natural selection [12]. Co-adaptation can occur between interacting genes or structures within an organism or between two or more interacting species. Co-adaptation is referred to as competitive co-evolution.

Competitive co-evolution is when two or more species compete against each other, such that when one species benefits, the other does not. Consider a certain species of plant living in an environment containing insects that eat the plant. To survive, the plant needs to evolve mechanisms to defend itself against the insects. The insects must evolve mechanisms to overcome the plant's defences to survive. This fight for survival drives an arms race. For example, the plant may develop a tough exterior, but then the insect develops stronger jaws. The plant may evolve a poison to kill the insect, but then the insect may evolve an enzyme to counteract the poison. The plant may develop hairs to make it difficult for the insect to climb, and the insect could develop longer legs.

Another classic example of competitive co-evolution is that of viruses and the immune system. In order to replicate, a virus must infect its host; replication requires resources that the host needs to function. The host evolves means to stop infection, while the virus evolves means to infect. Competitive co-evolution ensures that each generation is better at its offensive or defensive role.

In nature, organisms evolve at the expense of individuals within the same species as well as individuals in different species. Co-evolution is not always destructive; mutual co-evolution also known as co-adaptation or co-operative co-evolution, is mutually beneficial to all the parties involved. In 1981, a paper by political scientist Robert Axelrod and evolutionary biologist William Hamilton explored the evolution of co-adaptation [50, 51]. They argued that co-adaptation between species can emerge through evolution and persist through numbers of generations. As an example, consider a bee and a flower: the flowers need the bees for pollination and the bees need the nectar for food.

Co-evolution is a process where one species evolve because of the influence of another species. In simulated evolution, competitive co-evolution requires an inverse of fitness between two or more populations. A win for one population results in a loss for the other population, driving a simulated arms race. To survive, the losing population must adapt, becoming the winning population. In a simulated co-operative/symbiotic co-evolution, two or more populations work together, so that success in one population improves the survival rate in the other.

The concept of co-evolution in EAs was first proposed by Reed *et al.* in 1967 [111]. Reed *et al.* used different populations to evolve alternative strategies for a simple game of poker. Smith and Price [198, 199, 200] used game theory and competitive co-evolution simulations in 1973 to show that a "limited war" or arms race benefits individual animals within a species as well as animals within other species. In 1990, Hillis [132] showed how competitive co-evolution can be applied to a practical optimisation problem, and more specifically, how the addition of co-evolving populations can improve performance by preventing the system from becoming stuck in local optima. Hillis experimented with GAs that evolve sorting networks. Hillis introduced the notion of host and parasite populations. The host tries to find the optimal sorting network, while the parasites are scored on how well they make the sorting network fail the test data.

In 1994, Jan Paredis [79] successfully applied co-evolution to a constraint satisfaction problem. In 1995, De Jong and Potter [79] used co-operative co-evolution to evolve islands of populations, where each island solves part of a solution. The island model is an example of co-operative co-evolution. A book by Axelrod, titled "The Evolution of Co-operation" [51] explores co-operative evolution versus competitive evolution. Axelrod solicited strategies from other game theorists to compete in a tournament. Each strategy was paired with another strategy for 200 iterations of the prisoner's dilemma game

and scored on the total points accumulated throughout the tournament. Axelrod showed that strategies which focused on co-operation outperformed those strategies that focused on competition with regards to the prisoner's dilemma game.

In the standard GA a fitness function measures how close a solution is to the perfect solution that solves the *objective*. The perfect solution represents the global optimum. Because the fitness measure is viewed independently of any other solution within the population, it is referred to as the *absolute* fitness [47]. *Objectivity* and *absoluteness* come only at the expense of significant knowledge about the task being solved [47]. This is significantly true for difficult problems with no analytical description, for example, a game or stock market trading. Consider the abstract strategy board game Go, in which the aim is to surround more territory than the opponent.

An absolute objective fitness function must consider every Go strategy both known and unknown to measure how close a solution is to the optimal solution, which is not possible. However, the rules of the game Go are well defined and if two solutions are made to play the game Go against one another the outcome results in a winner. This kind of fitness evaluation is known as a competitive fitness function. Rather than calculating how close a solution is to playing the perfect game of Go, a competitive fitness function determines which solution is better at playing the game. The competitive fitness function determines the success of a solution relative to another solution. Angeline and Pollack [47] defined a relative fitness function as any fitness calculation that is dependent on other individuals to calculate the fitness measure [47].

To measure the performance of a solution against all the solutions within the population, Angeline and Pollack used a tournament relative fitness [47, 185]. Individuals are randomly paired to compete using a competitive fitness function. The loser is removed from the tournament. Winners move to the next round. The process is repeated until an overall winner is determined. Individuals are ranked by the number of rounds they lasted. Hillis [79, 132, 185] used a bipartite relative fitness, where each individual is tested against all the other competing individuals. If the individual performs better than the competing individual, then that individual receives a point. The individuals are ranked by their accumulative points.

Evaluating co-operatively co-evolved individuals is more complex than competitively co-evolved individuals. Each co-operatively co-evolved individual is part of a solution and must share its fitness value with an individual from another population. Holland [144, 193] devised the "bucket brigade" credit scoring system. The "bucket brigade" was modelled after the water passing chains of fire fighters. Each type of bucket is passed through a set of fire fighter chains. The best bucket, is the one that lost the least amount of water as it was passed along the various chains. The best fire fighter chain, is the

one that lost the least amount of water across all the buckets the fire fighters passed along. A "bucket brigade" credit system can be used to evaluate co-operatively evolved individuals. The fitness value of each individual is the sum of all the evaluations the individual took part in.

Various "bucket brigade" sampling approaches exist. The most computational approach is to pair all co-operatively co-evolved individuals with every other individual. De Jong and Potter [79, 193], and Axelrod [51, 193] paired each individual from one island to each other individual from another island. Cliff *et al.* [83, 193] paired each individual from one population to the best individual from the other populations. Price [179, 193], and Phelps *et al.* [174, 193] paired each individual from one population with a randomly selected set of individuals from another population. Co-operative co-evolution evolves populations together, as one population increases its performance, so do the other populations.

Rosin and Belew [79, 185] showed that competitive co-evolution can lead to an arms race, in which the two populations reciprocally drive one another to increasing levels of performance and complexity. Two populations were used, one to act as a host and the other as a parasite. Rosin and Below used the games of Nim and 3-D Tic-Tac-Toe as test problems to explore three new techniques in competitive co-evolution, namely competitive fitness sharing, shared sampling, and hall-of-fame.

Competitive fitness sharing is a relative fitness function, where parasites are ranked by the number of hosts that can not defeat it. The host fitness is calculated as a function of parasite difficulty. The effect is to reward hosts that can defeat parasites that other hosts can not defeat.

Shared sampling provides a method for selecting a strong, diverse set of parasites. Testing all the hosts against all the parasites is computationally expensive. A sub-set of parasites could be selected at random. However, the best parasite could be missed. Rosin and Belew proposed a competitive fitness sharing amongst the parasites. Each parasite is tested using competitive fitness sharing against the best performing parasites of the previous generation. The best performing parasites are those parasites that outperformed the most hosts. The parasites that outperform the previous generation of parasites are used to test the competitive fitness of the host population [185].

The hall-of-fame encourages an arms race by saving the best individuals from prior generations. Rosin and Belew introduced an infinite population model: The best $n$ parasites of a generation are maintained indefinitely. Shared sampling is then performed on the best individuals from all the previous generations, and not just the previous generation [185].

Different co-evolutionary approaches have been implemented by researchers. Grefenstette and Daley [129] wanted to understand which approach worked best, and if co-evolution outperforms a traditional GA. Their experiments used a GA to optimise what they termed "the most optimum symbolic reactive rule (SAMUEL)". The evolved rule has applications in sentry robots, autonomous delivery

vehicles, undersea surveillance vehicles, and automated warehouse robots. Grefenstette's and Daley's tests included one competitive task and one co-operative task. The tests require two agents, either acting as a co-operator or a competitor depending on the test. The following co-evolutionary approaches were compared:

- A single GA where the first agent evolves, and the second agent uses a hand crafted set of rules.
- A single GA that evolves a test of rules used by both agents (traditional GA).
- A separate GA for each agent, with each agent being evaluated against a random member of the opposite agent.
- A separate GA for each agent, with each agent being evaluated against the best individual of the previous generation of the opposite agent.
- A separate GA for each agent, with each agent being evaluated against a random selection of the previous generations of opposite agents (hall-of-fame).
- A separate GA for each agent, with each agent being evaluated on its ability to defeat opponents that few others in the current generation can (competitive fitness sharing).

Grefenstette and Daley showed that a competitive fitness sharing strategy produced generations with greater diversity and resulted in a solution that performed better than the other co-evolutionary approaches [129]. While Grefenstette and Daley considered evolutionary GAs, their approaches could be applied to GPs.

## 3.5 Summary

This chapter began with the history of evolutionary theory, which researchers simulated using a computer. The chapter covers the history of evolutionary computation and how it is used to evolve solutions to problems. The chapter introduced GP as an extension of GAs. The next chapter discusses research into stock market forecasting. More specifically, using evolutionary computation to evolve trading rules that are used to trade shares on various stock markets.

# Chapter 4

# Evolutionary Algorithms in Trading

*Progress comes from the intelligent use of experience.*

*-Elbert Hubbard (Writer and Philosopher)*

Technical analysis functions, as described in Chapter 2, are used to determine the market trend and potential changes in the trend. Traders use combinations of these functions to create trading rules. This chapter focuses on how researchers have used EAs described in Chapter 3 to determine the optimal trading rules for share trading. This chapter concludes with a summary in Section 4.2.

## 4.1 Evolved Trading Rules

A trading rule is simply a set of rules that decide to buy, sell or hold stock. One of the first researchers to use GAs to evolve trading rules was Richard Bauer [63]. Bauer published a number of papers on GAs and computerised trading strategies between 1992 and 1994 [64, 66, 67]. Bauer was one of the first researches to publish a book covering GAs and trading rules, titled "Genetic Algorithms and Investment Strategies" in 1994 [63]. His book was followed by another book in 1999, titled "Technical Market Indicators: Analysis & Performance" [65]. Bauer used the GA popularised by Holland [53, 111, 138] (defined in Chapter 3) to select the best combination of trading functions from a set of trading functions to form a trading rule. Bauer linked each bit position in the GA bit string to a technical analysis function. The bit value determined if the function was included in the rule or not. Each function returned a buy, sell or hold signal, encoded as 1, -1 or 0 respectively. The sum of all the enabled functions determined the action. If the sum was greater than 0 a buy action was taken, if the sum was less than 0 a sell action was taken, else nothing was done.

In 2001, Lam [154] implemented the same GA structure as Bauer [65]. However, each technical function was a form of fuzzy logic. Fuzzy logic is a general system for performing approximate reasoning [154]. Lam defined fuzzy logic as the mapping from an input space to an output space by making use of a set of *if-then* conditional statements or rules. An example of a fuzzy logic trading rule defined by Lam is "*if RSI is low then decision is short*". An alternative rule is "*if RSI is high then decision is long*". Low and high are arbitrary values determined by the creators of the technical analysis function. In this case high is anything above 70 and low is anything below 30. Hold, short, and long are defined as 0, -1 and 1 respectively, and correspond with hold, sell and buy. Lam defined 36 such fuzzy logic rules [154]. Like Bauer, Lam linked each of the functions to a bit within the chromosome string of a GA. The technical analysis function corresponding to the bit is enabled if the bit is 1, otherwise the function is disabled. The GA was tested on historic Hong Kong market data. Each individual within the population was tested against a share by running each enabled function against the share's daily price and volume data to determine a hold, sell, or buy action. The sum of the results determine the action. If the sum is greater than 0, then a buy action is performed. However, if the sum is less than

0, a sell action is performed. If the sum is 0, then nothing is done. Lam implemented two different training approaches, namely a static and a dynamic training approach. The static approach considered the standard GA used by Holland [53, 97, 135, 136, 137]. Data is divided into two distinct sets of data; training data and out of sample testing data. Lam implemented a dynamic data training approach that evolves a GA on market data of $m$ preceding days and then uses the best performing GA to trade $n$ days. Lam used four different values for $m$ and $n$:

- $m = 90, n = \infty$
- $m = 90, n = 60$
- $m = 60, n = 30$
- $m = 30, n = 15$

Lam showed that a GA could return profit on historic data, and that an incremental approach to training indicated more reliability, because the incremental approach generated more profitable buy and sell signals [154]. Lam used a return on investment (ROI) as quantifier for trader performance. ROI measures the gain or loss generated by the trading rule relative to the money invested.

$$ROI = \frac{\left[\sum_{t=1}^{T}(Price_t - sc) \times I_s(t)\right] - \left[\sum_{t=1}^{T}(Price_t + bc) \times I_b(t)\right]}{\left[\sum_{t=1}^{T}(Price_t + bc) \times I_b(t)\right]}$$

where $I_b(t)$ and $I_s(t)$ are equal to one if a rule signals a buy and sell, respectively, and zero otherwise; $sc$ represents the selling cost and $bc$ the buying cost. ROI is the difference between final bank balance and starting bank balance after trading.

In 2004, Schoreels *et al.* [190] showed that a simplistic GA could evolve an optimised trading rule that produced similar returns to investment funds run by human professionals. Schoreels *et al.* used an integer-based string of varying cardinality to represent a chromosome. Each gene in the chromosome corresponds to a technical analysis function. The gene value represents a parameter value used by the corresponding technical analysis function. Schoreels *et al.* [189] defined asset value as the sales price of a share on any day less the purchase price including costs. The average asset value was used as a mechanism to quantify the performance of a trading rule. The accumulated average asset value (AAV) is defined as:

$$AAV = \frac{\sum_{i=1}^{N}[(Price_s - sc) - (Price_b + bc)]}{N}$$

where $i$ is a buy and sell trading event, $N$ is the number of buy and sell events, $s$ the day the sale took place, and $b$ is the day the purchase took place.

Schoreels *et al.* expanded their GA to include 28 trading rules [188, 189] and compared the traditional static training strategy to the dynamic trading strategy of Lam [154]. Schoreels *et al.* showed that the dynamic approach outperformed the static approach. The dynamic approach of Lam [154], and Schoreels *et al.* [188, 189], which evolves trading rules on data closest to the testing data, validates the claim by Gartley presented in Section 2.2 that the price performance on a specific day could indicate the probable changes that could occur the next day [147].

Nicholls *et al.* [171] implemented a GA, with a similar structure to that of Lam [154], to explore the effect of the fitness function on trader performance. They compared the return on investment generated by trading rules evolved using the ROI fitness function implemented by Lam [154] to the return on investment generated by trading rules evolved using the AAV fitness function implemented by Schoreels *et al.* [188, 189]. Nicholls *et al.* also explored the effect of Ockham's razor on the trader performance.

Ockham's razor is a principle from philosophy, that supposes that if there exist two explanations for an occurrence, the simpler one is usually better. Nicholls *et al.* [171] introduced a penalty factor to penalise individuals with more enabled trading rules. It was shown that no difference between the results of trading rules generated by either fitness functions were observed when the volatility of the share was low. However, when volatility increased, the AAV produced better trading rules. No significant difference between GAs penalised for complex rules or those that were not penalised could be found.

In 2009, Ghandar *et al.* [125] also implemented a GA to generate trading rules using fuzzy logic. Like Lam [154], Ghandar *et al.* defined trading rules, where each rule is linked to a bit in a chromosome. While Lam hard coded the 36 fuzzy logic rules [154], Ghandar *et al.* allowed the GA to determine the fuzzy logic rule from a set of nine simple trading functions [125].

Ghandar *et al.* defined change in price, SMA, OBV, double SMA, and portfolio value as linguistic variables, and used these variables to form nine simple trading functions, called *linguistic trading functions* [125]. A linguistic trading function returns a trading signal. The signal is either a 1 for buy, 0 for nothing, or -1 for sell. Each simple linguistic trading function requires three inputs: enabled, fuzzy logic set and weight. Enabled is a boolean value, either 0 or 1. The fuzzy logic set is one of seven categories namely: extremely low (EL), very low (VL), low (L), medium (M), high (H), very high (VH), and extremely high (EH). The fuzzy logic set is calculated by running the linguistic variable against all the historic data to determine a range. The range is divided into seven equal parts and assigned to the different categories. The categories are updated as new trading data is observed. A weight variable determines the strength of the returned trading function signal. The weight variable can be any of the

predefined set of nine possible values starting at 0.1 and incrementing by 0.1 (0.1, . . . ,1.0). The returned trading signal is multiplied by the weight.

An example of a simple trading rule is "*if (enabled and SMA is [Fuzzy Logic]) then return 1 × [weight] else return 0;*". A total of 30 possible fuzzy logic rules were defined. Ghandar *et al.* did not use a binary string encoding as used by Holland, but rather a matrix of variables [53, 111, 138]. The chromosome structure is illustrated in Figure 4.1. The matrix consists of 30 rows, one for each fuzzy logic rule. Figure 4.1 illustrates rows 1 to 30. Each fuzzy logic rule comprises of a boolean variable (B), an integer variable (I) and nine linguistic functions. The boolean variable determines if the fuzzy rule is enabled or not. The integer contains a natural number between 1 and 9, corresponding to the weight values 0.1 to 1.0. Each linguistic function consists of 3 variables; a boolean variable (B) to enable or disable the function, a fuzzy logic set category (FL) and a weight (I) assigned to the function. Each enabled linguistic function returns a value between -1 and 1. If the rule is disabled a value of 0 is returned. If the fuzzy rule is enabled, the sum of the linguistic functions multiplied by the weight is returned as the value for the corresponding row, otherwise zero is returned.

The trading action is determined by the sum of all the trading rules. If the sum is greater than 0, a buy action is taken. If the sum is less than 0, a sell action is taken, otherwise no action is taken.

It was shown that the fuzzy logic approach of Ghandar *et al.* outperformed fixed trading strategies such as buy-and-hold, random-walk, and market indices such as MSCI Europe listed stocks for European stocks spanning 1990 to 2005 [125]. The MSCI represents the largest European shares based on market capitalisation. The buy-and-hold strategy buys shares on the first day of trade and sells the shares on the last day of trade. A random-walk strategy randomly buys shares, sells shares, or does nothing over the trading period. Ghandar *et al.* implemented the same static and dynamic training approach as Lam, and showed that a dynamic training outperforms static training.

Nicholls *et al.* [171] used a GA to compare the weighted strategy of Ghandar *et al.* [125] to the standard enabled or disabled strategy of Lam *et al.* [154]. The purpose was to explore the effect of weighted technical analysis rules on trader performance. It was shown that weighted strategies perform better than non-weighted strategies on bearish shares, and concluded that the weighted approach of Ghandar *et al.* is risk adverse.

Bauer [65], Lam [154], Ghandar *et al.* [125], and Schoreels *et al.* [188, 189] used GAs to evolve trading rules. A GA uses a vector structure to represent a chromosome. A tree structure provides three advantages over a vector, namely not restricted to a fixed length, a natural structure to represent hierarchical decisions rules, and a tree can store both binary, logical and numeric functions within its nodes. A tree structure is therefore a natural representation for a trading rule. One of the first researchers

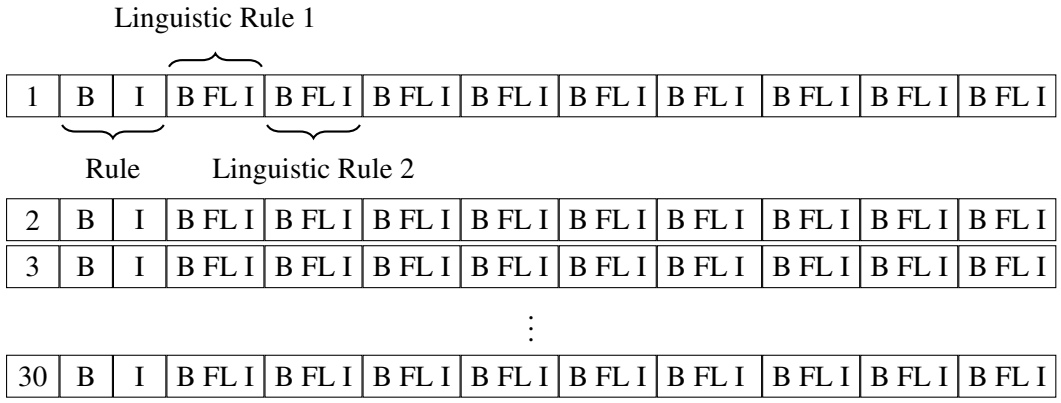



**Figure 4.1:** Chromosome structure implemented by Ghandar *et al.* [125] where B represents a boolean value enabled or disabled, FL is one of seven fuzzy logic categories, and I is a natural number between 1 and 9 representing a weight. Each row represents a fuzzy logic rule. A rule consists of 9 linguistic rules. Linguistic rules are defined by their position and are hard coded.

to evolve trading rules using tree structures was Allen and Karjalainen [43, 44].

Allen and Karjalainen [43, 44] encoded a trading rule as a decision tree. A grammar and tree traversal were defined to extract and execute the trading rule. An in-order traversal was used to extract the node values from the tree. The grammar defines non-terminal node values and terminal node values.

Non-terminal nodes have two possible function types: a real number function and a boolean function. A real number function returns a real number while a boolean function returns a boolean value.

The real value functions include "average", "minimum", "maximum", "arithmetic operators", "lag" and "norm". "Minimum" returns the minimum of two real value parameters. "Maximum" returns the maximum value of two real-valued parameters. The "arithmetic operators" perform their functions $(+, -, \div, \times)$ on two real-valued parameters. "Lag" returns the price of the share $n$ days prior, where $n$ is a natural number. "Average" returns the SMA for the $n$ days prior. "Norm" returns the absolute value between the difference of two real-valued parameters.

The boolean functions include "if-then-else","and","or","not", "$<$" and "$>$". "If-then-else" requires three boolean parameters, "and" requires two boolean parameters, "or" requires two boolean parameters, and the logical comparison operators ("$<$" and "$>$") require two real-valued parameters. Parameters are defined as the children of the node.

Terminal nodes are either the boolean constants `true` or `false`, "price", a real number constant, or a natural number constant. Real number and natural number constants are drawn from a uniform distribution of numbers between a minimum and maximum configured value during initialisation of the

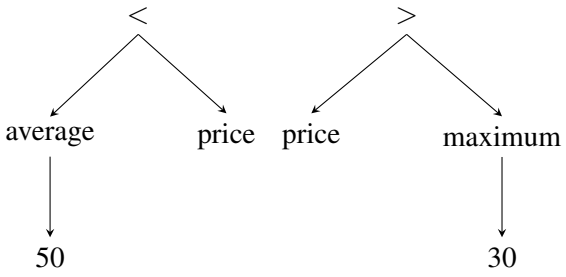



**Figure 4.2:** Allen and Karjalainen [43, 44] decision trees. The left tree corresponds to a 50-day moving average, returning a buy action if the closing price is greater than the 50-day SMA otherwise a sell action is returned. The tree on the right depicts a break out rule, similar to BBs. If the day's trading price is greater than the maximum price over the last 30 days then a buy action is returned, otherwise a sell actions is returned.

GP. "Price" is the price on that day's trade.

The trading rule returns a boolean value of `true` or `false`, which is transcoded to buy, sell or hold depending on whether the trader has shares or not. The rule is described in Table 4.1. For the rule to be grammatically correct, the root node must return a boolean value.

Examples of the Allen and Karjalainen [43, 44] decision trees are shown in Figure 4.2.

**Table 4.1:** Trading actions based on the result of a trading rule and shares on hand as defined by Allen and Karjalainen [43, 44].

| | **has shares** | **no shares** |
|---|---|---|
| **true** | hold | buy |
| **false** | sell | hold |

Allen and Karjalainen [43, 44] used a fitness measure based on the compounded excess returns over the buy-and-hold strategy. The excess return is given by:

$$\Delta r = r - r_{bh}$$

where the continuously compounded return of the trading rule is computed as

$$r = \sum_{t=1}^{T} r_i I_b(t) + \sum_{t=1}^{T} r_f I_s(t) + n \log\left(\frac{1-c}{1+c'}\right)$$

and the return for the buy-and-hold strategy is calculated as

$$r_{bh} = \sum_{t=1}^{T} r_t + n \log\left(\frac{1-c}{1+c'}\right)$$

In the above,

$$r_i = \log P_t - \log P_{t-1}$$

and $P$ is the daily close price for a given day $t$, $c$ denotes the one-way transaction cost; $r_f$ is the risk free cost when the trader is not trading, $I_b(t)$ and $I_s(t)$ are equal to one if a rule signals buy and sell, respectively, and zero otherwise; $n$ denotes the number of trades and $r_b h$ represents the returns of a buy-and-hold, while $r$ represents the returns of the trader.

A fixed trading cost of $c = 0.25\%$ of the transaction [43, 44] was defined. The continuously compounded return function rewards an individual when the share value is dropping and the individual is out of the market. The continuously compounded return function penalises the individual when the market is rising and the individual is out of the market.

The same crossover process defined in the Section 3.3 was used by Allen and Karjalainen. Mutations are introduced by randomly generating a new tree in place of the second parent used in crossover [43, 44]. This technique is known as headless-chicken [145].

Using data from the S&P 500 index and the NYSE Allen and Karjalainen [43, 44] showed that evolved trading rules do not earn excess returns over a simple buy-and-hold strategy after transaction costs. The trading rules enter the market when returns are positive and daily volatility is low and stay out of the market when the returns are negative and volatility is high. The trading rules are sensitive to trading costs, but showed robustness to the impact of the 1987 stock market crash [43, 44]. Lowering the trading costs, increases the returns of the GP [43, 44]. The GP of Allen and Karjalainen was reimplemented by Neely *et al.* [170], Telbany [102], Mahfoud and Mani [160, 161], Li and Tsang [204, 205], and Potvina *et al.* [177] with varying degrees of success.

Neely *et al.* [170] implemented a similar GP as Allen and Karjalainen, and showed that consistent, significant returns could be earned in currency markets against the USD over the period 1981-1995.

Telbany [102] experimented with a GP similar to Neely *et al.* and Allen and Karjalainen, to trade on the Egyptian Stock Market. The results of the GP were compared with the results of an ANN. The results re-enforced the findings of Mahfoud and Mani [160, 161], and showed that a GP outperforms an ANN when predicting stock market time series data. Iba and Sasaki [141] also compared a GP to an ANN using a simulated stock market and came to the same conclusions. Telbany [102], Iba and Sasaki [141], and Mahfoud and Mani [160, 161] showed mixed results in returns generated by the GP. They showed that the GP performed marginally better than a buy-and-hold strategy [102, 141, 160, 161]. They also showed that some shares are easier to trade than others. Their results are consistent with Neely *et al.* [170], and Allen and Karjalainen [43, 44].

Butler [205] implemented a GP to classify horses as winners or losers. Butler named his GP the

evolutionary dynamic data investment evaluator (EDDIE). EDDIE was shown to be significantly better at predicting the winners and losers of a race than two other horse classifying systems, SEAGUL developed by de la Maza [92] and HOBBES developed McNatton [166]. Butler showed that SEAGUL, HOBBES and EDDIE are significantly better at predicting the outcome of a race than human experts [205]. Li and Tsang [204, 205] later enhanced EDDIE and applied it to financial forecasting. Li and Tsang changed the decision tree of EDDIE to use the grammar defined by Allen and Karjalainen [204, 205]. Tsang showed mixed results using EDDIE to forecast the market [204, 205]: EDDIE is successful at making high volume low margin trades. The downside is that high volume incur high transaction costs. If forced to make high margin trades, EDDIE fails to capitalise on changes in the market. The findings of Tsang were that transaction costs determine the success of a GP.

Potvina *et al.* [177] evolved trading rules using the GP structure defined by Allen and Karjalainen, only to show that the algorithm over-fitted the training data. Potvina *et al.* showed positive returns on sample data, but not on the out-of-sample data.

Chen and Yeh [82, 188] also developed a GP for stock market forecasting, using an approach fundamentally different from the approaches described above. Until now, a GA or GP was used to find a trading rule that outperformed the market. The fitness of an individual was determined by the return on investment obtained at the end of trading. Chen and Yeh defined a GP to predict the next day's share price. The fitness of an individual is based on how well the individual predicts the future price, using the mean absolute percentage error (MAPE). The MAPE is calculated as follows:

$$MAPE = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

where $A_t$ is the actual share price for day $t$ and $F_t$ is the forecast share price for day $t$. The closer MAPE is to zero, the better the individual performed.

An in-order traversal was used to extract the future price calculation. Terminal nodes are defined as a natural number between 1 and a maximum, or real numbers between -1 and 1. A natural number terminal node represents a day's closing price. Therefore, a value of 10 represents the closing price 10 days ago. A real number is a node used as a parameter to a non-terminal node. The non-terminal nodes use the mathematical operators $+, -, \div, \times$, log, sin, and cos. The algorithm was evaluated on the S&P index and TAIEX indexes. If the evolved solution predicted an increase in the future price, the shares were bought. If the evolved solution predicted a decrease in the price the shares were sold. The main findings were that the GP successfully evolved a function to map the sample data, but over-fitted and therefore failed to produce positive returns on out-of-sample data [82].

Kaboudan [146], also developed a GP to predict the future share price using a GP and a grammar

similar to that of Chen and Yeh. Non-terminal nodes comprised of mathematical operators $(+, -, \div, \times)$, sin, cos, exponent, square root, and the natural logarithm. Terminal nodes could be a random real number constant, or a price $n$ days in the past. The GP was evaluated on different share data (i.e. Citigroup, Compaq, General Electric, Pepsi, Sears and Microsoft) and randomly generated share data. The GP was used to predict the share's high price, and to predict the share's low price. The regression sum of squared errors (ESS) was used to quantify fitness, calculated as follows:

$$ESS = \frac{100}{n} \sum_{t=1}^{n} (A_t - F_t)^2$$

Results showed that the GP could forecast the next day's minimum and maximum share prices. However, when presented with the randomly generated share data, the GP was unable to predict the next day's low and high share price. When presented with different shares it was evident that the GP could forecast the next day's minimum and maximum share prices with varying degrees of accuracy [146].

Most EAs used the daily close, open, high, and low price to evolve the trading rules. Wilson and Banzhaf [210] focused on forecasting the individual trades that occur throughout the day using the same GP as Kaboudan [210]. Each daily transaction is known as a *tick*. Intra-day trading focuses on ticks rather than the daily close, open, high or low price. Intra-day trading generates a large amount of data. Filters were developed to reduce the dataset: a low frequency filter, which removes trades that do not change the share price, and a high frequency filter, which calculates the moving average of the trade and blocks trades that do not change the moving average.

It was shown that adding filters to high frequency trading results in more conservative investing. The filtered system results in greater return on investment in some price trends, but overall does not perform as well as filter-less systems. The filter-less system takes advantage of anomalies in the data that the filtered system misses, resulting in greater return on investment. The low frequency filter had no effect on trading performance. Wilson and Banzhaf concluded that agents should use all data and not filter out data that may be deemed outlier data [210], because a GP may use the outlier data to detect trend changes [210].

By 1999, fewer than fifty papers had been published on the topic of EAs in financial markets [146]. Researchers showed that EAs could capitalise on market bias and outperform a buy-and-hold strategy. However, transaction costs determined the profitability of the trading rules.

Myszkowski *et al.* [169] compared three different GP approaches: A standard GP with one tree structure that determines both the buy and sell action, an extended GP where the GP evolves two trees (one for a buy action and the other for a sell action), and a co-operative co-evolved GP where one

population evolves a buy action and the other the sell action. An in-order traversal was used to extract the trading rule [169]. Terminal nodes contain a technical analysis function, closing price, volume, or a real number. The technical analysis functions include SMA, MACD, ROC, and RSI. Non-terminal nodes contained the functions $>$, $<$, AND, and OR. The fitness function is calculated as follows:

$$fitness = \frac{CASH_{stop}}{CASH_{start}}$$

where $CASH_{start}$ is the opening bank balance and $CASH_{stop}$ is the bank balance at the end of trading.

Tournament selection was used to select the individuals for the next generation. The co-evolved GP used an average of the fitness function to find the best individuals from each population. An individual is selected from the buy population and tested against all the individuals in the sell population. The average fitness across all the tests is assigned to the individual from the buy population. The process is repeated using the sell population.

It was shown that a GP can generate a trading rule that outperforms the standard buy-and-hold strategy. However, not enough data was available to show which GP technique was better than the another [169]. The co-evolved traders produced greater gains on upward markets while the standard GP produced greater gains on downward markets.

Myszkowski and Rachwalski [169] noted that data used to evolve the GP is important as those individuals evolved on downward trending data produced higher returns in all market conditions compared to individuals evolved on upward trending data [169].

Seshadri [193] compared the general algorithm on Allen and Karjalainen [43, 44] to two variations of the same algorithm. The first variation extended an individual's chromosome to two distinct branches joined by a combination node. The combination node decided which of the two branches to execute. The left branch determined when to buy shares and the right branch determined when to sell shares. The second variation was a co-operative co-evolved GP. Seshadri [193] evolved two different populations. The first population evolved a buy rule, and the second population a sell rule. Seshadri [193] showed that a co-operative co-evolved GP performed better than a single population GP and could outperform a buy-and-hold strategy when trading the S&P500.

Dreżewski *et al.* [98] successfully implemented co-evolved GPs to generate buy and sell rules. Dreżewski *et al.* compared a simple GP to a co-operative co-evolved GP and found the co-evolved GP approach to be more profitable.

## 4.2 Summary

This chapter provides a review of previous research on EAs used to generate trading rules. The next chapter builds on the work discussed in this chapter, and presents a GP and a competitively co-evolved GP that are used in this study to evolve trading rules.

# Chapter 5

# Genetic Program For Trading Rules

*Evolution continually innovates, but at each level it conserves the elements that are recombined to yield the innovations.*

*- John Henry Holland (Scientist, Professor of Psychology, Electrical Engineering and Computer Science.)*

The purpose of this chapter is to describe the GP implementation used in this study. This chapter begins with Section 5.1 describing the GP implementation used in the empirical study with regards to grammar, recombination, mutation, selection, and the fitness function. Section 5.2 introduces the datasets used in this thesis. Section 5.3 empirically determines the most suited selection operator, by presenting various selection operators, describing the empirical process and discussing the results. Section 5.4 discusses the effect that three different fitness functions had on evolving trading rules. The results of parameter sensitivity analysis are presented in Section 5.5. This chapter concludes with a summary in Section 5.6.

## 5.1 Genetic Program for Stock Market Trading

A GP is a specialised GA where the individual's chromosome is encoded as a decision tree. A tree structure is able to encode logical formulae, arithmetic formulae, or computer programs. Each tree represents a symbolic expression defined by a tree grammar.

A trading rule results in a buy, sell or hold action. A trading rule is represented as either a logical formulae, arithmetic formulae, or a computer program. Therefore, a tree is a perfect encoding structure for a trading rule.

A tree requires a grammar to correctly encode and read a logical formulae, arithmetic formulae, or "a computer program". The grammar used in this thesis is based on the tree structure implemented by Allen and Karjalainen [43], Neely *et al.* [170], Telbany [102], and Li and Tsang [204, 205].

The basic tree structure defined by Allen and Karjalainen was introduced in Chapter 4. This thesis extends the grammar of Allen and Karjalainen with the introduction of additional functional operators and terminal constants. The additions make it possible for the GP to evolve trading rules similar to technical analysis functions: CCI, MFI, OBV, and ADI.

CCI, MFI, OBV, and ADI require daily volume, high, and low price values. To make it possible for evolution to evolve these technical analysis functions, the terminal set was expanded to include: "high" and "low", respectively returning the highest, and the lowest prices of the day. The functional set operators "minimum" and "maximum" were replaced by "minimum price", "maximum price", "minimum volume", and "maximum volume" over a period of $n$ days, where $n$ is a parameter to the operator. The complete list of function set operators is presented in Table 5.1.

Each function within the function set had a set arity or number of parameters. A function with an arity $n$ had $n$ branches associated to it. Each parameter within the function had a required type. Each associated branch returned a value of the required parameter type.

Three parameter types were defined to ensure that a function was passed the correct parameter type. These types were: natural number, numeric, and boolean. A numeric could either be a natural number or a real number. The boolean constants were either `true` or `false`, while the real number constants were drawn from a uniform distribution between configured minimum and maximum values, and natural numbers were drawn from a uniform distribution between configured minimum and maximum natural numbers. Allen and Karjalainen [43] did not specify the configured minimum and maximum natural numbers, except that they are parameters to the lag, and average functions. For this study, natural numbers were drawn from a uniform distribution between 1 and 500. The reason for choosing 1 and 500 was that the dataset samples used did not exceed 500 days.

Allen and Karjalainen [43] specified the limits of the real numbers to be between 0.0 and 2.0, without offering any explanation why. A positive real number limit between 0.0 and 2.0 in conjunction with positive natural number limits does not allow the evolution of negative numbers. Some technical analysis functions such as the oscillating indexes defined in Chapter 2 return values within limits such as -100.0 and 100.0 or -1.0 and 1.0. Changing the real number limits from 0.0 and 2.0 to -1.0 and 1.0 allowed the GP to evolve a sub-tree that returned negative numbers such as -100.0 ($-1.0 \times 100$).

The mathematical operator "$\div$" can invalidate a trading rule, as division by zero is undefined. The second parameter of the "$\div$" operator was therefore restricted to a non-zero numeric constant.

To simplify the tree structure, the tree generation algorithm automatically simplified nodes. For example, if initialisation or mutation grew a sub-tree, ($-1.0 \times 100$), the node $\times$ was automatically simplified to a numeric constant -100. This ensured that the tree sizes remained small and decreased the computation time when evaluating the decision trees. Only sub-trees that did not include a variable such as opening price were simplified.

Each tree structure was subject to the evolutionary process defined by Holland [135, 137, 136] and Koza [150]. The evolutionary process has four operations, namely competition, reproduction, mutation, and selection [111].

Many variations of the evolutionary process exist, some of which have been described previously. This thesis implements the same basic GP algorithm implemented by Allen and Karjalainen [43]. The pseudo code listing presented in Algorithm 3 outlines Allen and Karjalainen's evolutionary process. The process is described later in more detail.

A GP is a specialised GA requiring evolutionary operators adapted to the tree encoded chromosome structure. A GP has a set of meta-parameters that must be configured. The evolutionary operators were selected through empirical analysis presented in Section 5.2.1, and the meta-parameters selected through parameter sensitivity analysis described in Section 5.5.

**Table 5.1:** GP Grammar - A list of possible nodes, their function, parameters and return types.

| Name | Placement | Description | Parameters | Return type |
|---|---|---|---|---|
| Boolean constant | Terminal Node | Returns either `true` or `false` | | Boolean |
| Real Number constant | Terminal Node | Returns a real number between -1.0 and 1.0 inclusive | | Numeric |
| Natural Number constant | Terminal Node | Returns a natural number between 1 and 500 inclusive | | Numeric |
| High | Terminal Node | Returns the highest price of the day | | Numeric |
| Low | Terminal Node | Returns the lowest price of the day | | Numeric |
| Price | Terminal Node | Returns the closing price of the day | | Numeric |
| Average | Non-terminal Node | Returns the SMA for the $n$ days prior | Natural number: $n$ | Numeric |
| Maximum Price | Non-terminal Node | Returns the maximum price for the $n$ days prior | Natural number: $n$ | Numeric |
| Minimum Price | Non-terminal Node | Returns the minimum price for the $n$ days prior | Natural number: $n$ | Numeric |
| Maximum Volume | Non-terminal Node | Returns the maximum volume for the $n$ days prior | Natural number: $n$ | Numeric |
| Minimum Volume | Non-terminal Node | Returns the minimum volume for the $n$ days prior | Natural number: $n$ | Numeric |
| Lag | Non-terminal Node | Returns the closing price $n$ days prior | Natural number: $n$ | Numeric |
| Norm | Non-terminal Node | Returns the absolute value of $x$ | Numeric: $x$ | Numeric |
| $+$ | Non-terminal Node | Returns the result of adding $a$ and $b$ | Numeric: $a$, Numeric : $b$ | Numeric |
| $-$ | Non-terminal Node | Returns the result of subtracting $b$ from $a$ | Numeric: $a$, Numeric: $b$ | Numeric |
| $\div$ | Non-terminal Node | Returns the result of dividing $b$ into $a$ | Numeric: $a$, non-zero numeric constant $b$ | Numeric |
| $\times$ | Non-terminal Node | Returns the result of multiplying $a$ and $b$ | Numeric: $a$, Numeric: $b$ | Numeric |
| Boolean if-then-else | Non-terminal Node | if $a$ is `true` return $b$ else return $c$ | Boolean: $a$, Boolean: $b$, Boolean: $c$ | Boolean |
| Numeric if-then-else | Non-terminal Node | if $a$ is `true` return $b$ else return $c$ | Boolean: $a$, Numeric: $b$, Numeric: $c$ | Numeric |
| And | Non-terminal Node | Returns the result of a logical AND between $a$ and $b$ | Boolean: $a$, Boolean: $b$ | Boolean |
| Or | Non-terminal Node | Returns the result of a logical OR between $a$ and $b$ | Boolean: $a$, Boolean: $b$ | Boolean |
| Not | Non-terminal Node | Returns the result of a logical negation of $a$ | Boolean: $a$ | Boolean |
| $<$ | Non-terminal Node | Returns the result of a logical less than comparison between $a$ and $b$ | Numeric: $a$, Numeric: $b$ | Boolean |
| $>$ | Non-terminal Node | Returns the result of a logical greater than comparison between $a$ and $b$ | Numeric: $a$, Numeric: $b$ | Boolean |
| Root | Root Node | Any function or constant that returns a boolean value | | Boolean |

**Algorithm 3:** Implemented GP algorithm as presented by Allen and Karjalainen [43].

$g = 0$; ▷ Generation counter
Set the maximum number of generations $G$;
Set the total number of individuals $\mu$;
Initialise the initial population $C_g$;
Draw three continuous samples from the dataset ▷ The dataset is the share data
Initialise the in sample dataset $Sample_{in}$ ▷ Defined in Section 5.2
Initialise the validation sample dataset $Sample_{sel}$
Evaluate fitness $f_{C_{g,n},Sample_{in}}$ of each individual $C_{g,n}$ in population $C_g$ using $Sample_{in}$;
Set $B = C_{g,n}$ where $C_{g,n}$ has the largest $Sample_{in}$ fitness;
Evaluate fitness $f_{B,Sample_{sel}}$ of $B$ using $Sample_{sel}$;
**while** $g < G$ **do**
  $g = g + 1$;
  Set number of offspring $\lambda = 0$;
  **while** $\lambda < \mu$ **do**
    Set random number $r \sim U(0,1)$;
    **if** $r < P_m$ **then**
      ▷ Sampling is performed using rank selection
      Sample parent $P_1$ from $C_{g-1}$ with bias to worst fitness;
      Mutate $P_1$ to form $O_1$;
      Evaluate the $f$ of $O_1$ using $Sample_{in}$ ;
      Add $O_1$ to $C_g$;
      Sample $D_1$ from $C_g$ with bias to worst $f_{C_g,Sample_{in}}$ ;
      Remove $D_1$ from $C_g$;
      $\lambda = \lambda + 1$;
      } Mutation
    **else**
      ▷ Sampling is performed using rank selection
      Sample parents $P_1$ and $P_2$ from $C_{g-1}$ with bias to best $f_{C_g,Sample_{in}}$;
      Offspring $O_1$ = Re-combination of $P_1$ and $P_2$;
      Offspring $O_2$ = Re-combination of $P_2$ and $P_1$;
      Evaluate the $f$ of $O_1$ and $O_2$ using $Sample_{in}$;
      Add $O_1$ and $O_2$ to $C_g$;
      Sample $D_1$ and $D_2$ from $C_g$ with bias to worst $f_{C_g,Sample_{in}}$;
      Remove $D_1$ and $D_2$ from $C_g$;
      $\lambda = \lambda + 2$;
      } Crossover
    **end if**
  **end while**
  Set $B_g = C_{g,n}$ where $C_{g,n}$ has the largest $f_{C_g,Sample_{in}}$
  **if** $f_{B_g,Sample_{in}} > f_{B,Sample_{in}}$ **then**
    Evaluate the $f$ of $B_g$ using $Sample_{sel}$;
    **if** $f_{B_g,Sample_{sel}} > f_{B,Sample_{sel}}$ **then**
      $B = B_g$;
    **end if**
  **end if**
**end while**
**return** $B$ as the solution;

The initial population of the GP was created stochastically. The population consisted of $\mu$ individuals. An individual represented a trading rule encoded as a parse tree. The trading rule was initialised stochastically to a maximum configured initialisation tree depth of three.

Three continuous samples were drawn from a dataset. The first sample represents the evolution sample referred to as $Sample_{in}$. The fitness values of the individuals within the population were quantified by a fitness function over $Sample_{in}$. The quantified fitness value was used by a selection operator to determine which individuals were selected for crossover, mutation, and removal for the current generation. An empirical analysis of three different fitness functions showed that the Allen and Karjalainen fitness function (AK) was the preferred fitness function for this study.

The second sample represents a selection validation sample and is referred to as $Sample_{sel}$. As discussed in Chapter 3, GAs are prone to over-fitting the data. To minimise over-fitting, the best individual within a generation ($B_g$) was compared to the global best individual ($B$). If the $Sample_{in}$ fitness of $B_g$ was greater than the $Sample_{in}$ fitness of $B$ and the $Sample_{sel}$ fitness of $B_g$ was greater than the $Sample_{sel}$ of $B$, then $B_g$ became the new global best $B$.

The third sample represents an out-of-sample dataset and is referred to as $Sample_{out}$. $Sample_{out}$ was not used during the evolutionary process, but $Sample_{out}$ was used to test the generalisation of the evolved solution on unseen data.

A steady state population model was implemented allowing for stronger individuals to survive many generations. During a generation, a probability factor determined if mutation was performed. Parameter sensitivity analysis determined that a static mutation probability factor of 35% was preferred for this thesis. Allen and Karjalainen [43] used a static mutation probability and a low number of generations. Further work is required to determine if a dynamic mutation probability that decreases over time results in better trading rules than a static mutation probability.

Mutation used a rank selection biased towards the worst individuals within the population. An empirical analysis of various selection operators showed that rank selection was the preferred selection operator.

Allen and Karjalainen [43] used headless-chicken mutation [43, 145] to mutate an individual. Headless chicken, while very simple in concept, is computationally more expensive than updating a single node. For example, suppose a terminal node is mutated, headless-chicken requires that an entire individual is initialised and then crossed over with the selected parent. In this case only one node is altered. It is computationally more efficient to alter the terminal node than growing an entire new tree to swap one node. For this reason two types of mutation were implemented, namely grow mutation and prune mutation.

Each mutation type was executed with equal probability. Prune mutation replaced a randomly selected node with a syntactically correct terminal node. Grow mutation replaced a randomly selected node with a new stochastically generated syntactically correct sub-tree.

Once mutation was completed, the new offspring was evaluated using a fitness function across the $Sample_{in}$ dataset. The new mutated individual was added to the population. An individual was then removed from the population using rank selection biased towards the worst individual. The offspring count $\lambda$ was incremented by 1.

If mutation did not occur, two individuals were selected as parents and underwent crossover to form two new offspring. Rank selection biased towards the best individuals within the population selected two parents from the population to undergo crossover. Crossover randomly selected a node within the first parent, and randomly selected a node from the second parent where the selected node was grammatically valid. If no grammatically correct node was found, a node within the second parent was randomly selected and matched to a grammatically correct node in the first parent. If no grammatically valid nodes were found, the parents were discarded and two new parents were sampled from the population.

If a node from the first parent was a grammatically valid replacement for a node within the second parent, the nodes were swapped to form two new offspring. The new offspring were quantified using a fitness function over the $Sample_{in}$ dataset. The new offspring were added to the population. Rank selection sampled two individuals from the population with a bias towards the worst individuals. The two selected individuals were then removed from the population. The offspring count $\lambda$ was incremented by 2.

The processes of crossover and mutation were repeated until the total number of generated offspring $\lambda$ was equal to or greater than the configured population size $\mu$. Because individuals were removed during crossover and mutation, the size of the population remained constant.

Once crossover and mutation have completed, the global best individual ($B$) was compared to the current generation's best ($B_g$) to determine a new global best individual. The fitness values obtained by the trading rules over both the $Sample_{in}$ and $Sample_{sel}$ datasets were compared. The current generation's best individual would only become the global best if its fitness was greater than the global best over both the samples. The generation counter was incremented and the process was repeated until the maximum number of generations had elapsed.

At the end of evolution the global best individual was returned as the evolved trading rule.

## 5.2 Datasets

The datasets used in this thesis were sourced from Sharenet. Sharenet is a market statistics company based in Cape Town, South Africa. The data contained eleven shares from a variety of market sectors; including mining, resources, consumer goods and services, industrials, banking, and insurance. The stocks are listed in Table 5.2 and cover the period April 2003 to June 2008. The start date is referred to as day 0. Trading days exclude weekends and public holidays. Simulations reference the share code, which is the code used by the JSE.

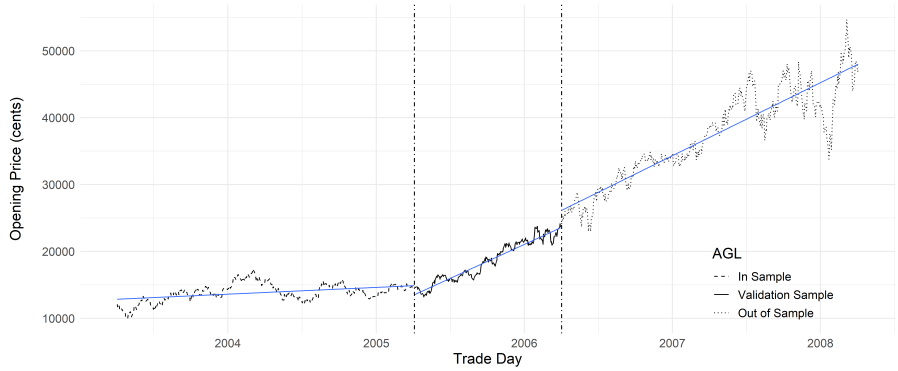
The stock market dataset spans 1$\dot{3}$50 days. In addition to company shares, the data includes the All Share Index 40 (ALSI40) market index. The ALSI40 index constitutes the 40 largest companies by market capitalisation across all sectors listed on the JSE. In addition to the datasets listed in Table 5.2, fictitious datasets known as *inverted shares* were created by reversing the share price of Nedbank, Remgro and Standard Bank. To reverse the dataset, a new list of inverted prices were generated where the inverted price on day 0 corresponded to the original price data on day 1$\dot{3}$50. The inverted price on day 1 corresponded to the original price on day 1$\dot{2}$49. The inverted price on day 2 equalled the original price on day 1$\dot{2}$48, and so on.

Each share dataset was divided into three continuous samples. The first sample, $Sample_{in}$, was used to drive the rule generation process. $Sample_{in}$ spans 2 years of trading days from day 0 to day 500. The selection validation sample $Sample_{sel}$, was used compare the performance of the best individual within a generation to the global best individual to determine the new global best. $Sample_{sel}$ spans a year of trading days from day 500 to 750. The out-of-sample, $Sample_{out}$, dataset is used to determine the performance of the trading rule against unseen data. $Sample_{out}$ spans 2 years of trading days from day 750 to 1250. Figure 5.1 plots the daily closing price of each share presented in Table 5.2 as well as the market trend across the three samples.

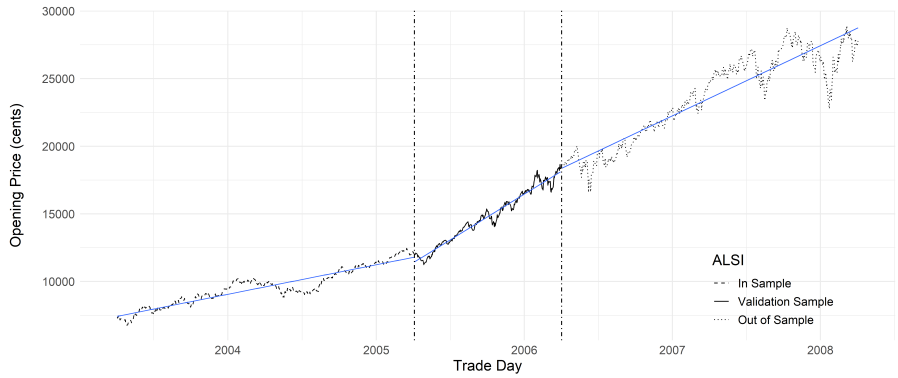
Appendix A provides a market overview covering each of the shares presented in Table 5.2.

**Table 5.2:** The share data used in simulations.

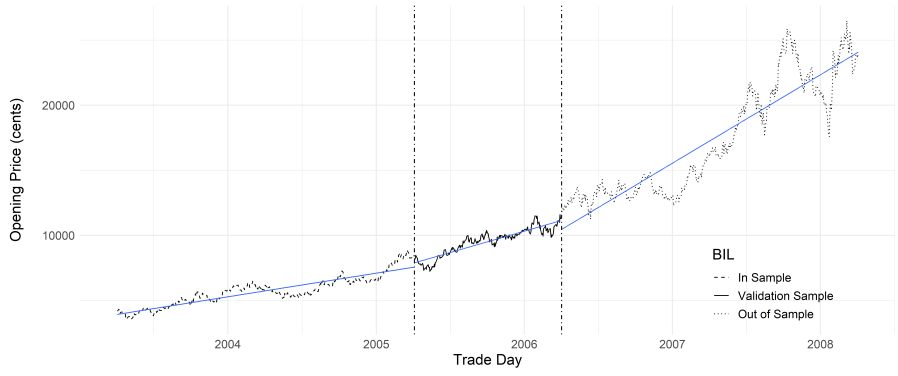
| Code | Short Name | Long Name | JSE Sector (Sector Code) | Start | End |
|---|---|---|---|---|---|
| AGL | Anglo | Anglo American Plc | Metals & Minerals (1775) | 2003-04-03 | 2008-06-18 |
| ALSI | All Share Index | JSE All Share Index (Top 40) | Exchange Traded Fund | 2003-04-04 | 2008-06-19 |
| BIL | Billiton | Bhp Billiton Plc | Metals & Minerals (1775) | 2003-04-05 | 2008-06-20 |
| GFI | Gfields | Gold Fields Limited | Gold Mining (1777) | 2003-04-06 | 2008-06-21 |
| IMP | Implats | Impala Platinum Holdings Limited | Platinum (1779) | 2003-04-07 | 2008-06-22 |
| LON | Lonmin | Lonmin Plc | Platinum (1779) | 2003-04-08 | 2008-06-23 |
| NED | Nedcor | Nedbank Group Ltd | Banks (8355) | 2003-04-09 | 2008-06-24 |
| RCH | Richemont | Richemont Securities AG | Clothing & Footware (3763) | 2003-04-10 | 2008-06-25 |
| REM | Remgro | Remgro Limited | Diversified Industrials (2727) | 2003-04-11 | 2008-06-26 |
| SAB | Sabmiller | Sabmiller Plc | Beverages - Brewers (3533) | 2003-04-12 | 2008-06-27 |
| SBK | Stanbank | Standard Bank Group Limited | Banks (8355) | 2003-04-13 | 2008-06-28 |
| SOL | Sasol | Sasol Limited | Oil - Integrated (0537) | 2003-04-14 | 2008-06-29 |



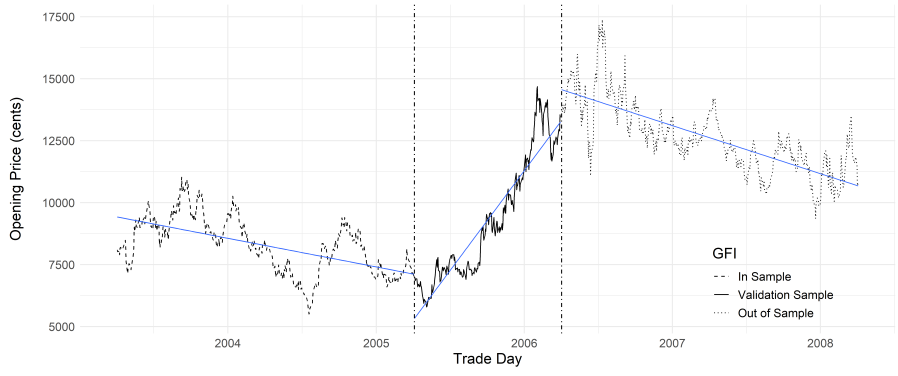

**(a)** AGL




**(b)** ALSI

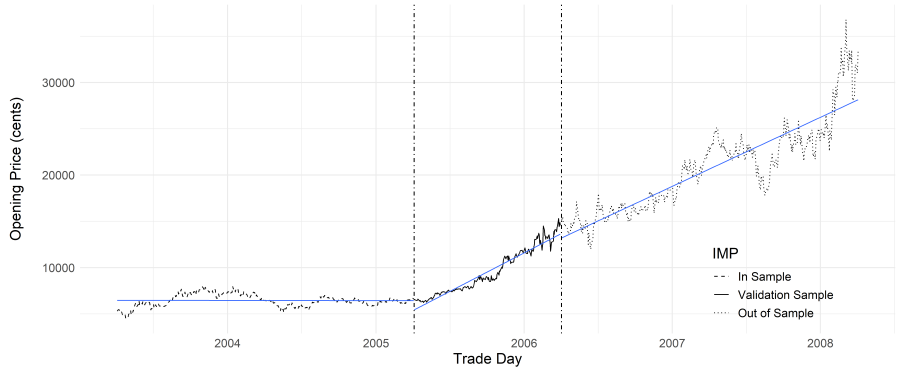



**(c)** BIL

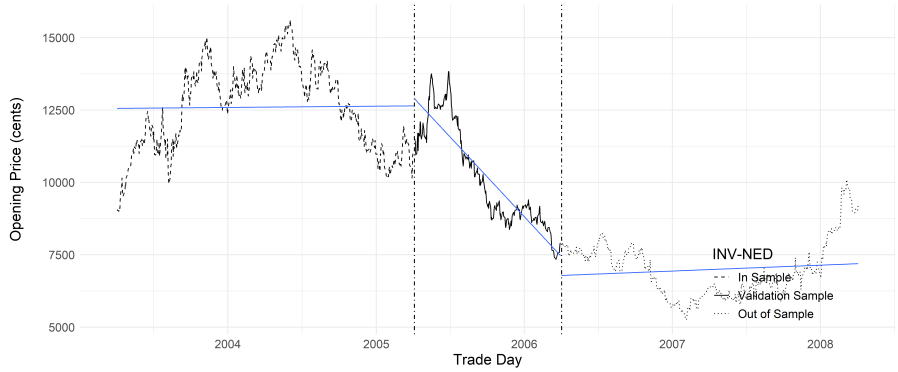



**(d)** GFI

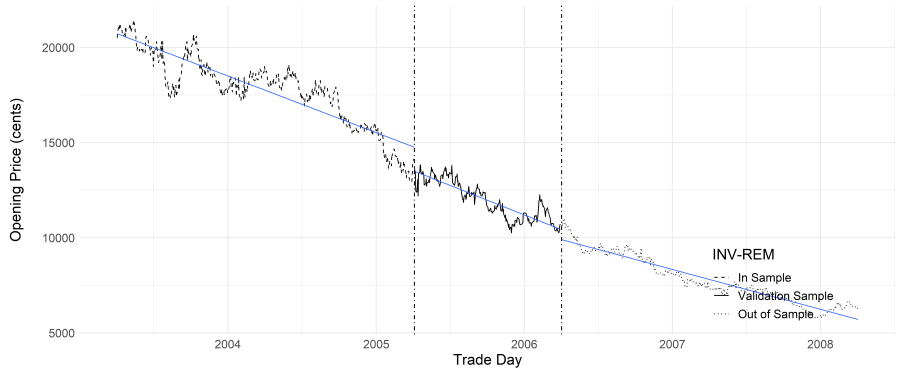
**Figure 5.1:** Share data used for the trading rule evaluation. Each graph shows the opening share price in cents, the average trend line per data sample, and the sample composition.



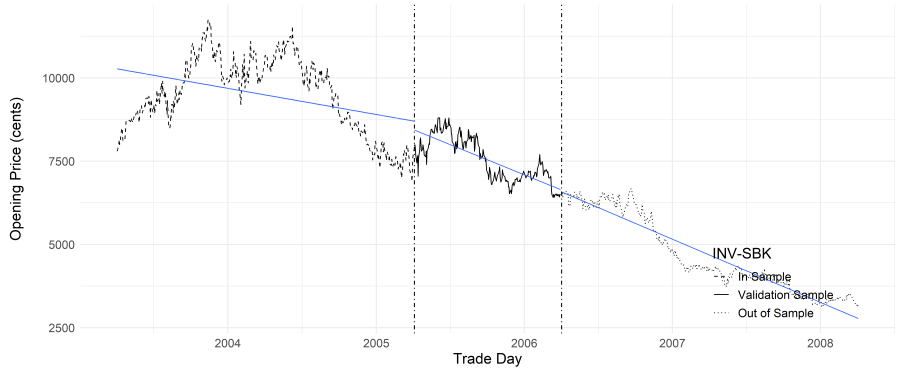

**(e)** IMP




**(f)** INV-NED

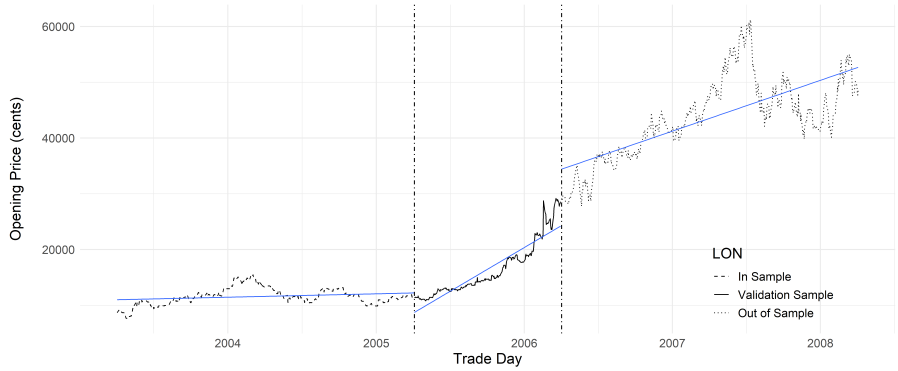



**(g)** INV-REM

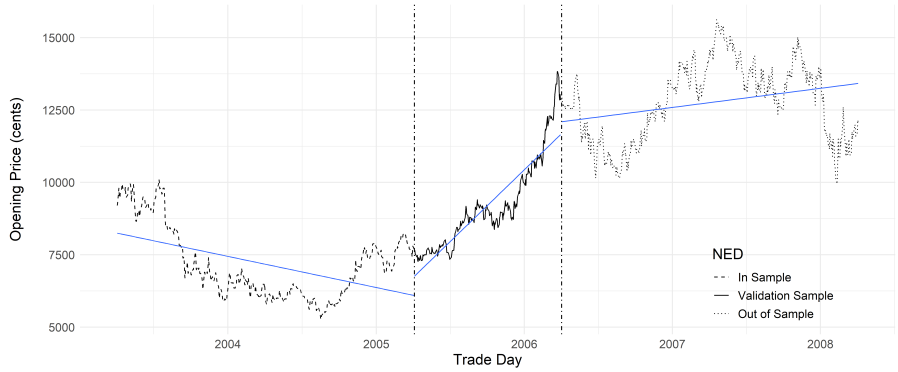



**(h)** INV-SBK

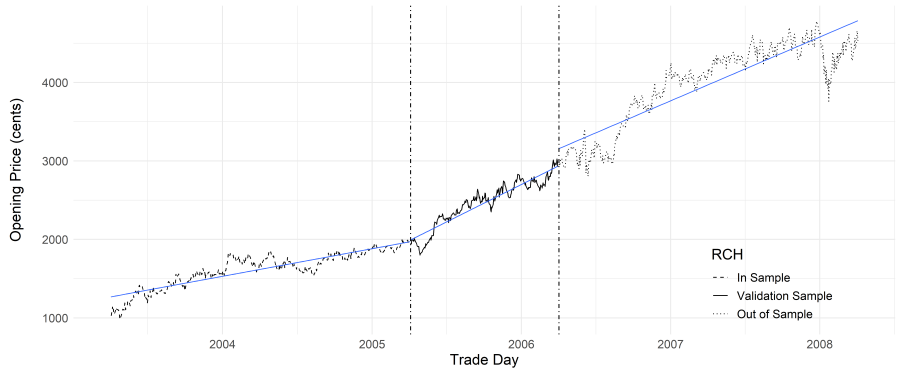
**Figure 5.1:** Share data used for the trading rule evaluation. Each graph shows the opening share price in cents, the average trend line per data sample, and the sample composition (cont.).



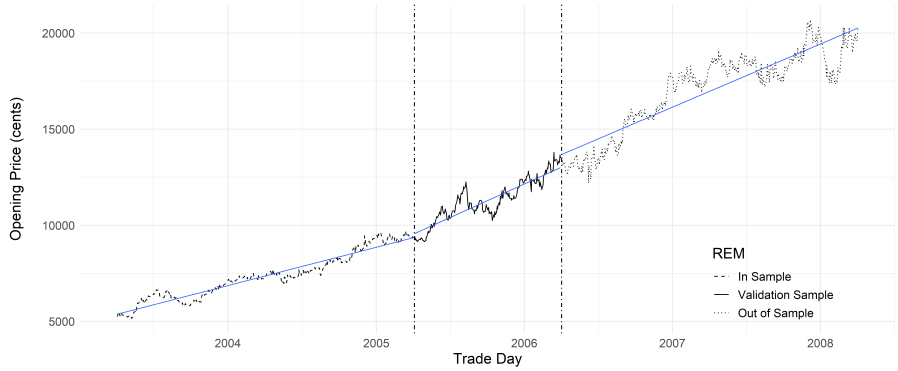

**(i)** LON




**(j)** NED

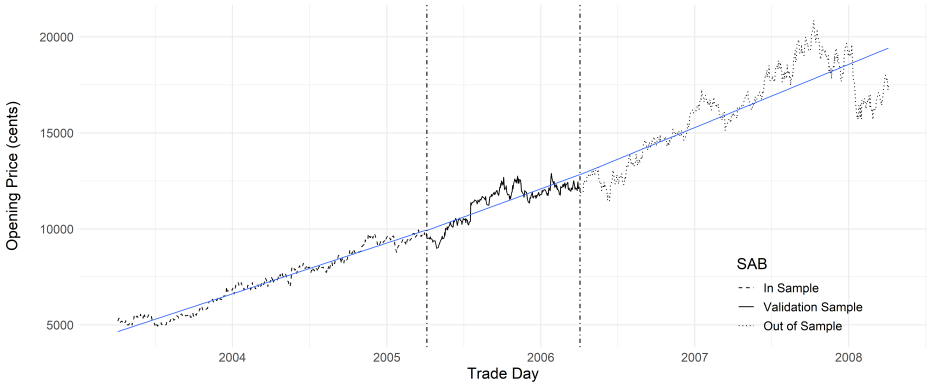



**(k)** RCH

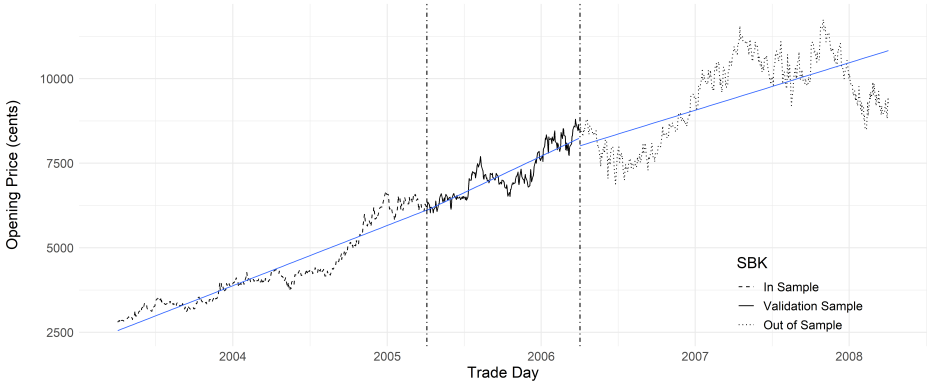



**(l)** REM

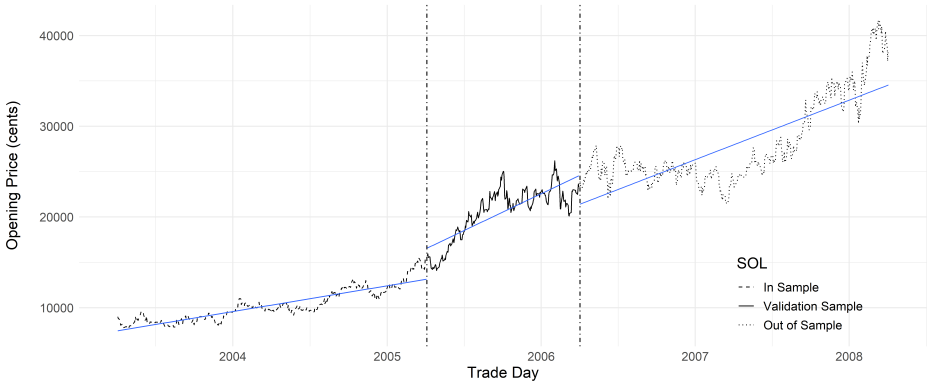
**Figure 5.1:** Share data used for the trading rule evaluation. Each graph shows the opening share price in cents, the average trend line per data sample, and the sample composition (cont.).




**(m)** SAB




**(n)** SBK




**(o)** SOL

**Figure 5.1:** Share data used for the trading rule evaluation. Each graph shows the opening share price in cents, the average trend line per data sample, and the sample composition (cont.).

### 5.2.1 Empirical Process

A GP is stochastic, meaning that each sequential run of a GP yields a random result from a probable distribution of results. To compare the performance of one GP configuration to another, an empirical process is required. An empirical process is a stochastic process based on the *central limit theorem* [70]. The central limit theorem states that, as the sample size increases without limit, the shape of the distribution of the sample means approaches a normal distribution. This is generally true regardless of whether the source results are normally distributed or skewed, provided that the sample is sufficiently large [40].

The evolutionary process drives a GP towards an optimal solution. Therefore, GP results are not normally distributed within the solution space, but rather skewed towards a set of solutions within the solution space. To ensure that the central limit theorem holds, a sufficiently large sample, usually greater or equal to 30, is required [70]. A large data sample ensures that there is enough data to distinguish the normal distribution from the outliers, providing a reasonable degree of confidence that the data are consistent with the distribution.

A sample was the result of a simulation run. A simulation was a fixed configuration of a GP. A simulation was repeated 30 times, from different initial conditions, recording the results as individual samples. The simulation was repeated for all share datasets independently.

Each configuration of a simulation was intended to test the effect the configuration had on the outcome of the simulation. However, each run of the simulation requires an initial set of randomly generated individuals. Naturally, if two GPs are initialised with two different populations of individuals, the outcome could be different. Furthermore, each decision such as whether to implement mutation or crossover depends on a random probability that affects the outcome of the simulation. To minimise the effect randomness had in comparing various simulations a fixed set of random seeds were used.

The random seed ensures that the GP returns the same result no matter how may times it is run. The random seed ensures that the GP is initialised with the same initial population. This means that if two GPs are run with the same random seed but a different configuration, then the result of the run is due to the configuration change and not the initial population. Thirty random seeds were generated, one for each simulation run and used across all simulations.

To compare the results of two or more different simulations the *null-hypothesis* was assumed [70]. The null-hypothesis assumes that the results of various simulations are the same, and that no further investigation is required unless the results are significantly different. A statistical test was performed on the results to determine if two of more simulation results rejected the null-hypothesis.

The most common statistical tests used to test the validity of the null-hypothesis are t-tests and

Wilcoxon signed rank tests [70, 96, 122]. These tests are not adequate when comparing more than two different result sets due to the multiplicity effect [128, 186]. Experiments in EAs have shown that the results do not fulfil the requirements for parametric testing such as t-tests [128, 195]. Therefore, a common non-parametric statistical test known as a Krushkal-Wallis test is used to test the validity of the null-hypothesis across more than two result sets.

The Krushkal-Wallis test is a non-parametric method based on a one-way analysis of variance (ANOVA) between the sample ranks. The Krushkal-Wallis test indicates if at least one sample is stochastically different from the other samples. The Krushkal-Wallis test does not show which samples are different [128]. If the Krushkal-Wallis test rejects the null-hypothesis, a post-hoc test is required to determine which result set fails the null-hypothesis [96, 128].

Post-hoc analysis performs a pairwise comparison of sample results to determine which sample results are different from one another. Traditionally, Wilcoxon signed rank tests are used for post-hoc analysis [70, 96, 122, 128].

The statistical tests return a value known as a *p-value*. The p-value is a number between 0 and 1 denoting the probability that the null-hypothesis is valid. A p-value of less than 0.05 indicates a strong possibility that the null-hypothesis is incorrect and that the samples are not the same.

Demšar [96] noted that ANOVA is based on assumptions which are most probably violated when testing the results of EAs. ANOVA assumes the samples are drawn from normal distributions. Demšar pointed out that there is no guarantee for normality of accuracy across a set of algorithms or various configurations of algorithms. The second more important assumption of ANOVA is that the random variables such as parameters and the initialised population have equal variance, which, as Demšar noted, can not be taken for granted [96].

Demšar [96] proposed that the Friedman test be used for pairwise comparison, and that the Iman and Davenport extension [142] to the Friedman statistical test be used for multiple results comparison in place of a one-way ANOVA.

If the null-hypothesis is rejected, Demšar proposed the Nemenyi test as a post-hoc pairwise test [96]. The Nemenyi test produces a critical difference value which is easily displayed graphically as a critical difference plot. An example of a critical difference plot is presented in Figure 5.2. The top line in the figure represents the average ranks. The lowest or best ranks are on the right. The critical difference is displayed above the graph. Algorithms similar to the control are within the critical difference marking. Similar algorithms are plotted to the left and the right of the control. Groups that are not significantly different are connected by a line. If there is no control sample, the sample with the highest mean is used as a control.

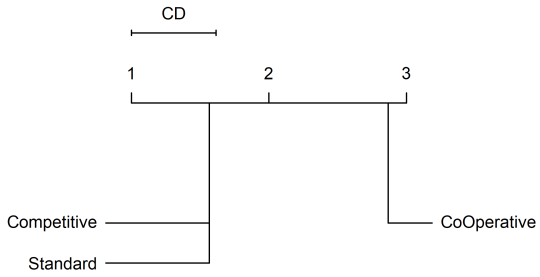



**Figure 5.2:** Critical difference plot illustration.

Figure 5.2 shows no significant difference was found between **Competitive** and **Standard** and that **CoOperative** is significantly different from both **Competitive** and **Standard**.

García *et al.* [122, 123, 124] discussed the advances in non-parametric testing and present corrections and extensions that improve the results obtained by the Friedman test when comparing multiple results. García *et al.* proposed that the Iman and Davenport extension is used instead of the standard Friedman test [122, 123, 124] for multiple comparisons.

The probability of rejecting the null-hypothesis increases as more samples are compared. The p-value is adjusted to compensate for this probability. A number of correction procedures exist including Holm [140], Holland [134], Hochberg [133], Rom [183], and Bonferroni-Dunn [99] to name a few [122, 123, 124]. García *et al.* recommended that Finner's [108] correction is performed on the result of Iman and Davenport test [142].

The empirical study conducted in this thesis used the Iman and Davenport test [142] with Finner's correction [108] to determine if the null-hypothesis was rejected. The null-hypothesis assumed that all the sample results are the same. A confidence level of 95% was used to reject the null-hypothesis. If the null-hypothesis was rejected post-hoc analysis was performed to find which samples failed the null-hypothesis.

Two kinds of post-hoc analysis were performed. The first was a Friedman test as suggested by Demšar [96] and García *et al.* [122, 123, 124], and the second was the Nemenyi test to generate critical difference plots. The Friedman test returns a p-value indicating the probability of the null-hypothesis being valid, while the critical difference plot provides a visual representation of result groups.

The empirical analysis was performed using the statistical comparison of multiple algorithms in multiple problems [78] library for R (a statistical programming language).

Experiments ran different simulations that were configured to use different fitness functions. To ensure that experiments were compared fairly, the profit returned at the end of trading was used to

quantify the performance of a simulation. Profit was calculated as:

$$Profit = \left[\sum_{t=1}^{T}(Price_t - sc) \times I_s(t)\right] - \left[\sum_{t=1}^{T}(Price_t + bc) \times I_b(t)\right]$$

where $I_b(t)$ and $I_s(t)$ are equal to one if a trading rule signals a buy and sell, respectively, and zero otherwise; $sc$ represents the selling cost and $bc$ the buying cost.

Chatper 4 discussed the effects that transactional costs have on trading. The greater the costs, the less profitable the trader's are. Because costs are generally proportional to the volume traded, trading costs were ignored for optimisation.

The final set of simulations were performed with cost and the trade of $1\dot{0}00$ shares. The mean profit was calculated as the mean of the profit returned by the best individual from each run of the simulation. The mean standard deviation of profit ($\sigma$) was calculated as:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

where $x_1, x_2, \ldots, x_N$ are the observed values of the sample items (i.e. profit with or without cost), $\bar{x}$ is the mean value of these observations, and $N$ is the number of observations in the sample. The lower the $\sigma$ the similar the trading rules.

## 5.3 Selection Strategy Sensitivity Analysis

Various selection strategies have been presented. Allen and Karjalainen [43] used rank selection to sample individuals for mutation, parental selection, and survivor selection. The empirical results discussed in Subsection 5.3.2 confirmed that rank selection was the preferred selection strategy.

Various selection strategies were compared using the empirical process defined in Section 5.2.1 to determine the most adequate selection strategy. The selection strategies compared were tournament selection, roulette wheel selection, elitist selection, and SUS [103].

Tournament selection requires a sample size. The smaller the sample the size, the more random the sampling, while the greater the sample size, the more biased the sampling is towards the best performing individuals within the population. A set of simulations were run comparing the results returned by the trading rules evolved using various tournament selection sample sizes, to determine the preferred sample size of 20%. The results are discussed in Subsection 5.3.1.

The results of trading rules evolved using tournament selection with a sample size of 20% was then compared to the other the results of trading rules evolved using other selection strategies to confirm that rank selection was the preferred selection operator. To compare the various selection strategies a

base GP configuration was required. The GP parameters of Allen and Karjalainen [43] were used as presented in Table 5.3. No transaction fees were used, and the number of shares bought and sold were limited to one.

**Table 5.3:** GP parameter configuration for selection strategy evaluation, based on the parameters used by Allen and Karjalainen [43].

| Parameter | Value |
|---|---|
| **Number of individuals** | 500 |
| **Number of generations** | 50 |
| **Maximum tree depth** | 10 |
| **Maximum number of tree nodes** | 100 |
| **Mutation probability** | 30% |
| **Fitness Function** | Allen and Karjalainen |

### 5.3.1 Tournament Selection

Tournament selection requires a sample size as a parameter. The sample size can be a fixed value or a percentage of the population size. To determine the preferred sample size, a percentage of the population was used. Ten different GP simulations were run each with a different sample size. All the simulations were configured using the same configuration presented in Table 5.3.

An initial sample size of 10% was used, incrementing it by 10 to 100% for a total of ten simulations. The simulations are named Tour10, Tour20, Tour30, ..., Tour100, where the number refers to the percentage of the population used for tournament sampling.

Each simulation recorded the best individual's $Sample_{out}$ profit. The mean of the 30 runs for each simulation across each share dataset is presented in Table 5.4. The bold values represent the highest mean profit, while the italic values represent the lowest mean profit. Profit after trading is presented in cents.

At first glance, it appears that a tournament sample size of 50% or 90% is preferred as they resulted in the best $Sample_{out}$ mean profit.

Table 5.5 compares the $Sample_{out}$ mean profit deviation. The deviation ($\sigma$) represents the level of convergence between the individuals. The lower $\sigma$, the more the individuals have converged on a solution that returns a similar profit. The $\sigma$ for Tour50 and Tour90 was too large, therefore not robust. Because Tour20 had the next best mean profit and a low $\sigma$, a tournament selection sample size of 20% was preferred.

To determine if Tour20 results were significantly different from the other tournament selection results, the null-hypothesis was assumed, declaring that all the results were not significantly different.

To test if the null-hypothesis was valid, the Iman and Davenport extension of the Friedman test with Finner's [108] correction was run [142] across the sample results. The p-values of the statistical test are presented in Table 5.6

The statistical results show that, after correction, none of the $Sample_{out}$ results were found to be significantly different. A significant difference was found in the $Sample_{sel}$ results when trading Richmond Securities AG (RCH) and SABMiller Plc (SAB).

Based on the $Sample_{out}$ results, it is not feasible to continue with a post-hoc analysis. However, because the p-values calculated from the Nedcor Ltd (NED) simulations show a significant difference before correction on the $Sample_{out}$ dataset, and the p-values of the RCH and SAB simulations show a significant difference before correction on the $Sample_{sel}$ dataset, post-hoc analysis was performed.

The Friedman [96] pairwise test was preformed on the significantly different results and is presented in Figure 5.3. The p-values for the RCH results indicate a significant difference between a Tour100 and the rest of the sample sizes. A significant difference was found between Tour80 SAB $Sample_{sel}$ results and Tour40 SAB $Sample_{sel}$ results. In most cases, the null-hypothesis was valid, and any tournament sample size would have been sufficient. However, a sample size still needed to be selected.

A win-loss ranking (win-loss) was performed to determine the overall performance of the individuals within a population. The profit from each best individual across each simulation run was compared to the profit of all the best individuals from all the other simulations. A simulation was awarded a point for each individual that returned a greater profit than another individual from a different simulation with the same share dataset. A simulation was deducted a point each time an individual had a profit less than an individual from another simulation with the same share dataset. For each share that the total wins were greater than the losses for a specific share, the rank was incremented.

The win-loss table shows the number of times that the best individuals outperformed the best individuals from another simulation. The win-loss table also shows in brackets the number of times the win count was greater than the loss count for a specific share and simulation run.

A win-loss ranking presented in Table 5.7 shows that tournament sample sizes of 10% and 20% resulted in trading rules that out-performed all the other trading rules across all three *Samples*, and that a sample size of 20% evolved better performing trading rules than trading rules evolved using a tournament selection sample size of 10%. It is for this reason that a tournament sample size of 20% was used during the selection strategy comparison that follows.

**Table 5.4:** Mean $Sample_{out}$ profit per share analysis.

| Share Code | Tour10 | Tour20 | Tour30 | Tour40 | Tour50 | Tour60 | Tour70 | Tour80 | Tour90 | Tour100 |
|---|---|---|---|---|---|---|---|---|---|---|
| **agl** | 25438.23 | 25768.77 | **26249.50** | 25991.00 | *24864.17* | 25991.00 | 25745.90 | 25246.70 | 25219.47 | 25855.07 |
| **alsi** | 9301.07 | 9272.70 | 9361.70 | 9357.37 | *9232.77* | 9403.10 | 9238.70 | 9278.37 | **9452.93** | 9287.17 |
| **bil** | 13728.30 | 14491.07 | *12507.50* | 13329.13 | **15280.87** | 13929.90 | 14987.33 | 13481.30 | 14583.57 | 14500.23 |
| **gfi** | -253.13 | 144.50 | -1074.10 | 216.10 | *-1776.40* | -749.57 | -313.93 | -543.57 | **223.97** | -708.97 |
| **imp** | **16612.00** | **16612.00** | **16612.00** | **16612.00** | **16612.00** | **16612.00** | **16612.00** | *16601.23* | **16612.00** | 16609.67 |
| **inv-ned** | 12233.53 | 11996.37 | 11938.90 | *11589.93* | **12249.00** | 12025.03 | 11817.83 | 11962.83 | 11869.37 | 11890.20 |
| **inv-rem** | 3461.60 | 4247.60 | 3465.47 | **4521.40** | 3318.33 | 3706.43 | 3768.40 | *3141.20* | 3413.10 | 3465.77 |
| **inv-sbk** | 5093.07 | *4767.13* | 5268.83 | 5018.37 | 5085.00 | 5203.30 | 5041.00 | 4833.00 | **5594.97** | 5418.10 |
| **lon** | 20239.87 | 19221.90 | *16209.37* | 20225.73 | 20227.17 | 19067.87 | 16698.50 | 16506.40 | **22319.00** | 19320.93 |
| **ned** | -1892.90 | -1104.80 | -961.03 | *-2313.00* | **-661.27** | -2015.07 | -2281.00 | -1880.97 | -2285.73 | -1790.40 |
| **rch** | 2190.60 | 2161.60 | 2208.57 | 2192.00 | 2183.57 | **2236.03** | 2156.80 | 2164.80 | 2176.17 | *2024.87* |
| **rem** | 9819.00 | 9832.40 | 9835.10 | 9832.20 | **9846.27** | 9843.57 | 9774.50 | 9802.77 | 9798.63 | *9595.00* |
| **sab** | 5365.33 | 5439.07 | 5314.60 | *5088.80* | 5422.43 | 5241.53 | **5698.17** | 5112.23 | 5607.97 | 5461.37 |
| **sbk** | 1623.17 | 1606.17 | 1653.27 | 1548.20 | *1544.60* | **1663.37** | 1549.03 | 1556.97 | 1618.97 | 1597.13 |
| **sol** | **14849.53** | 12926.00 | 13500.70 | 14614.67 | 14004.13 | 12133.67 | 12459.73 | 12970.87 | 13849.60 | *12082.00* |

Note: **bold** represents the highest value, and *italic* the lowest value.

**Table 5.5:** Mean $Sample_{out}$ profit $\sigma$ analysis.

| Share Code | Tour10 | Tour20 | Tour30 | Tour40 | Tour50 | Tour60 | Tour70 | Tour80 | Tour90 | Tour100 |
|---|---|---|---|---|---|---|---|---|---|---|
| **agl** | 1042.91 | 429.65 | 499.77 | **0.00** | *2103.42* | **0.00** | 865.32 | 1438.98 | 1491.63 | 262.80 |
| **alsi** | *677.78* | 492.23 | 325.77 | 381.79 | 355.95 | 316.47 | 552.60 | 450.75 | **282.84** | 354.84 |
| **bil** | 3444.07 | 2250.78 | *3775.63* | 3361.45 | **1549.90** | 2970.73 | 2286.38 | 3080.06 | 2585.24 | 2213.34 |
| **gfi** | 2194.79 | 2203.90 | 2525.96 | 2299.78 | 2083.65 | 1855.00 | 2395.67 | **1664.90** | *2755.69* | 2171.03 |
| **imp** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | *20.82* | **0.00** | 4.51 |
| **inv-ned** | **695.01** | 1082.50 | 1059.09 | *1348.23* | 894.13 | 925.82 | 1187.79 | 1282.58 | 1113.47 | 1075.25 |
| **inv-rem** | 2427.25 | 2060.71 | 2453.10 | **1074.36** | 2266.36 | 2164.28 | 2074.47 | 2384.79 | 2117.34 | *2458.88* |
| **inv-sbk** | 1145.44 | *1371.16* | **600.78** | 1212.38 | 1187.73 | 1006.47 | 1085.00 | 1266.07 | 615.98 | 610.13 |
| **lon** | **4251.12** | 5621.37 | 7294.49 | 5784.84 | 4900.71 | 6189.79 | *7733.17* | 7420.85 | 5083.07 | 5746.61 |
| **ned** | 2260.85 | **1662.28** | 2290.90 | 2279.00 | 1982.10 | 2177.60 | 1767.67 | 1735.63 | *2298.33* | 1922.40 |
| **rch** | 202.67 | 214.31 | **162.66** | 182.60 | 182.10 | 192.48 | 196.72 | 190.15 | 200.38 | *259.01* |
| **rem** | **0.00** | 25.91 | 31.13 | 25.92 | 55.36 | 47.50 | 87.70 | 147.44 | 214.28 | *396.40* |
| **sab** | 758.11 | 754.10 | 578.35 | *832.25* | 429.37 | 530.32 | **313.13** | 729.56 | 772.22 | 424.55 |
| **sbk** | 227.90 | 178.84 | 224.82 | 225.57 | 211.68 | 175.51 | 238.05 | 176.88 | **151.82** | *277.85* |
| **sol** | 3354.24 | 5961.40 | 5454.83 | **3013.07** | 4403.26 | 6244.76 | *6849.46* | 5399.15 | 3498.16 | 6086.60 |

Note: **bold** represents the lowest value or more preferred $\sigma$, and *italic* the highest or least preferred $\sigma$.

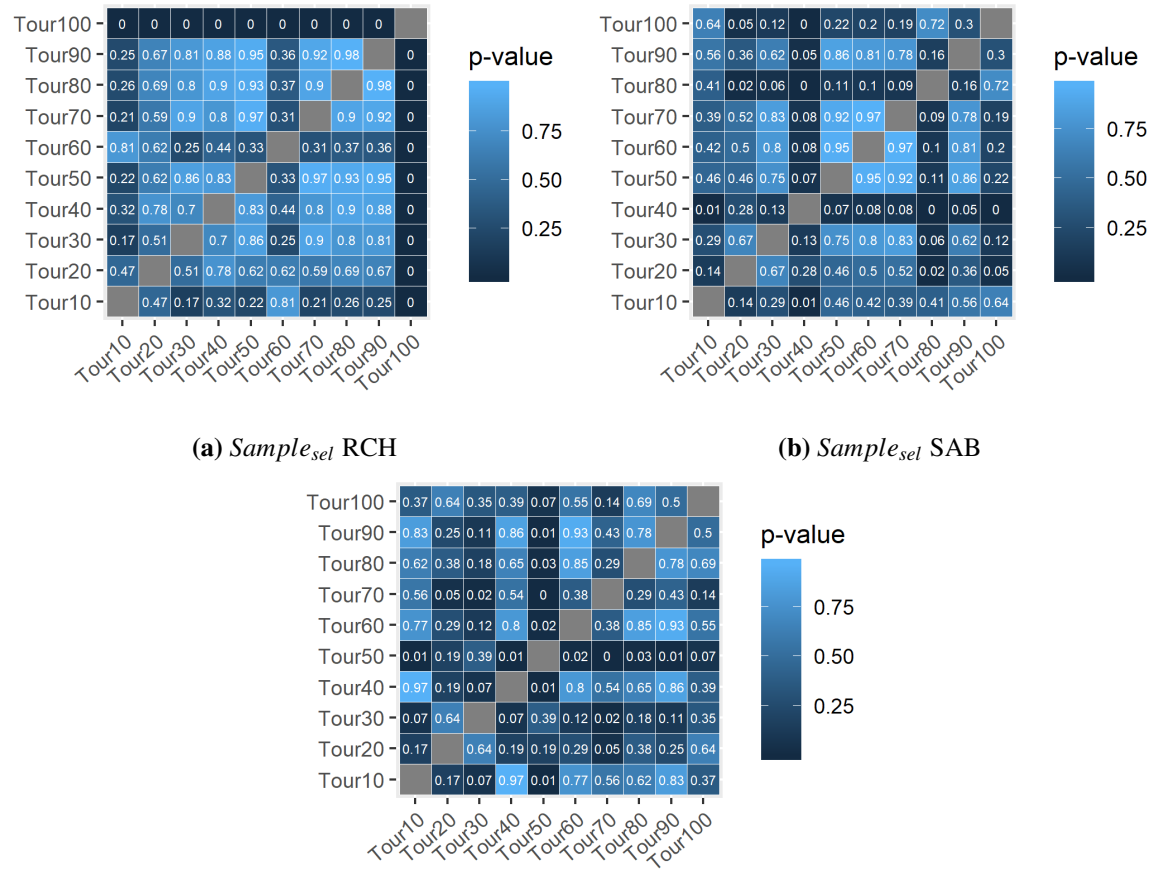
**Table 5.6:** p-value results using Iman and Davenport test with Finner's correction.

| | $Sample_{in}$ | | $Sample_{sel}$ | | $Sample_{out}$ | |
|---|---|---|---|---|---|---|
| **Share Code** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** |
| **agl** | 0.995212 | 0.997181 | 0.977789 | 0.987635 | 0.999948 | 0.999974 |
| **alsi** | 0.958453 | 0.995016 | 0.165154 | 0.594462 | 0.323347 | 0.858150 |
| **bil** | 0.552850 | 0.993638 | 0.466016 | 0.847741 | 0.503300 | 0.858150 |
| **gfi** | 0.904743 | 0.993638 | 0.208242 | 0.594462 | 0.365972 | 0.858150 |
| **imp** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| **inv-ned** | 0.631749 | 0.993638 | 0.871336 | 0.956562 | 0.801395 | 0.946880 |
| **inv-rem** | 0.722276 | 0.993638 | 0.755475 | 0.928697 | 0.791003 | 0.946880 |
| **inv-sbk** | 0.973805 | 0.995703 | 0.847693 | 0.956562 | 0.398486 | 0.858150 |
| **lon** | 0.851801 | 0.993638 | 0.506021 | 0.847741 | 0.075282 | 0.457946 |
| **ned** | 0.693556 | 0.993638 | 0.670151 | 0.907142 | **0.040004** | 0.457946 |
| **rch** | 0.490500 | 0.993638 | **0.000001** | **0.000012** | 0.840278 | 0.946880 |
| **rem** | 0.985376 | 0.995703 | 0.968617 | 0.986791 | 0.991544 | 0.995942 |
| **sab** | 0.281567 | 0.992989 | **0.044250** | 0.287833 | 0.707602 | 0.928277 |
| **sbk** | 0.973569 | 0.995703 | 0.998751 | 0.999225 | 0.985131 | 0.994808 |
| **sol** | 0.993833 | 0.997181 | 0.863695 | 0.956562 | 0.924223 | 0.970345 |

Note: **bold** values represent a significant difference.

**Table 5.7:** Win-loss ranking.

| **Simulation** | $Sample_{in}$ | $Sample_{sel}$ | $Sample_{out}$ | **Average** |
|---|---|---|---|---|
| **Tour10** | 4701(66) | 4417(67) | 2048(69) | 3722(67.333) |
| **Tour20** | 1084(71) | 1501(79) | 4851(66) | 2478.667(72) |
| **Tour30** | 3066(64) | -2285(91) | -1072(74) | -97(76.333) |
| **Tour70** | 1280(69) | -1117(79) | -1180(83) | -339(77) |
| **Tour90** | -987(86) | 1601(78) | 5001(67) | 1871.667(77) |
| **Tour50** | -999(82) | -755(81) | 2829(69) | 358.333(77.333) |
| **Tour60** | -1637(95) | -535(78) | 1207(63) | -321.667(78.667) |
| **Tour40** | -5208(111) | 7072(65) | 53(80) | 639(85.333) |
| **Tour100** | -2655(80) | -4681(75) | -4985(101) | -4107(85.333) |
| **Tour80** | 1355(71) | -5218(102) | -8752(124) | -4205(99) |




**(a)** $Sample_{sel}$ RCH

**(b)** $Sample_{sel}$ SAB

**(c)** $Sample_{out}$ NED

**Figure 5.3:** Friedman pairwise comparison of p-values.

### 5.3.2 Selection Strategy Comparison

A number of selection strategies exist. Four of the most popular selection strategies were compared to determine if the rank selection used by Allen and Karjalainen [43] was preferred. The selection strategies compared were tournament selection with a 20% sample size, roulette wheel selection, elitist selection, and SUS [103].

A simulation was setup for each of the selection strategies. Each simulation was run independently for each share presented in Table 5.2. The average profit obtained by the best trading rules from each simulation was recorded and is presented in Table 5.8. The mean profit results show that Tour20 obtained the highest mean profit in 6 of the 15 shares, followed by roulette wheel selection and SUS obtaining the highest mean profits across 4 of the shares traded. The worst performing simulations

returning the lowest profit were configured with elitism selection.

The results of the simulations were compared using the empirical process described in Section 5.2.1 to determine which of the selection strategies was best suited for this study. The Iman and Davenport [142] statistical test was applied to the simulation results to determine if the results were significantly different. Table 5.10 shows the p-values returned by the Iman and Davenport. The p-values show a high confidence that a significant difference exists between the $Sample_{out}$ results returned by the trading rules evolved using the BHP Billiton Plc (BIL), Inverted Remgro Ltd (INVREM), and NED datasets.

Post-hoc analysis was done comparing the significantly different results. The Friedman pairwise results presented in Figure 5.4 show a significant difference between the results returned by the trading rules evolved using elitism selection and the trading rules of the rest of the selection operators. The critical difference plots presented in Figure 5.5 show that the results returned by the trading rules evolved using elitism selection were critically different from the results returned by the other selection strategies. Based on the poor performance of the trading rules evolved with elitism selection as presented in Table 5.8 and the post-hoc results, elitism selection was excluded from any further analysis.

The Friedman p-values failed to find a significant difference between the results returned by traders evolved with tournament selection, rank selection, roulette wheel selection and SUS. The critical difference plot in Figure 5.5g shows that the trading rules evolved using Tour20 for the BIL dataset produced critically different $Sample_{out}$ results compared to the trading rules evolved using SUS and roulette wheel selection. Table 5.8 shows that the trading rules evolved using Tour20 resulted in a greater average $Sample_{out}$ profit than those trading rules using SUS and roulette wheel selection. Because no significant difference and no critical difference was found between the results returned by the trading rules evolved using Tour20, and the results returned by the trading rules evolved using rank selection, both rank selection and Tour20 are the preferred selection strategies when evolving trading rules for the BIL dataset.

Figure 5.5i shows that a critical difference exists between the results returned by that trading rules evolved using SUS and the results of the trading rules for the rest of the selection operators for the NED dataset. Because trading rules evolved with SUS returned the lowest average $Sample_{out}$ NED profit as presented in Table 5.8, SUS is the least preferred selection strategy when evolving trading rules for the NED dataset.

The results presented show that Tour20 and rank selection are the most preferred selection strategies, because no significant difference or critical difference was found between the results returned by the trading rules evolved using Tour20 and rank selection. Tour20 and rank selection simulations together resulted in the highest mean $Sample_{out}$ profit as shown in Table 5.8.

The win-loss results presented in Table 5.9 show that rank selection evolved trading rules that out-performed other trading rules consistently across all the *Samples*. The win-loss results also show that rank selection resulted in trading rules that on average out-performed trading rules evolved using Tour20. Therefore, based on these observations, and that Allen and Karjalainen [43] used rank selection, rank selection remains the preferred selection strategy for the purposes of this study.

**Table 5.8:** Mean $Sample_{out}$ profit per share analysis.

| Share Code | Tour20 | Elitism | Rank | Roulette | SUS |
|---|---|---|---|---|---|
| **agl** | 25768.77 | **25991.00** | 25258.77 | *25246.70* | 25901.27 |
| **alsi** | 9272.70 | 9376.33 | **9400.57** | 9263.63 | *9097.17* |
| **bil** | **14491.07** | 13703.50 | 14327.53 | *10601.17* | 11655.03 |
| **gfi** | **144.50** | -1246.73 | *-1260.00* | 79.67 | -432.90 |
| **imp** | **16612.00** | **16612.00** | **16612.00** | **16612.00** | **16612.00** |
| **inv-ned** | 11996.37 | *11646.50* | **12891.57** | 12414.73 | 12015.13 |
| **inv-rem** | 4247.60 | *3252.00* | 5315.40 | 5588.80 | **5817.63** |
| **inv-sbk** | *4767.13* | 5097.23 | 5465.23 | **5590.57** | 5454.63 |
| **lon** | **19221.90** | 18639.13 | *18202.93* | 19158.03 | 18539.13 |
| **ned** | **-1104.80** | -1889.40 | -2399.90 | -1802.83 | *-3155.03* |
| **rch** | 2161.60 | *2089.63* | 2188.17 | 2225.33 | **2261.57** |
| **rem** | 9832.40 | *9819.00* | *9819.00* | **9848.50** | 9838.50 |
| **sab** | 5439.07 | 5308.10 | *5160.70* | 5456.87 | **5516.80** |
| **sbk** | **1606.17** | 1533.23 | 1555.27 | *1382.17* | 1439.73 |
| **sol** | 12926.00 | *10300.30* | 14196.20 | **14772.27** | 14290.93 |

Note: **bold** represents the highest value, and *italic* the lowest value.

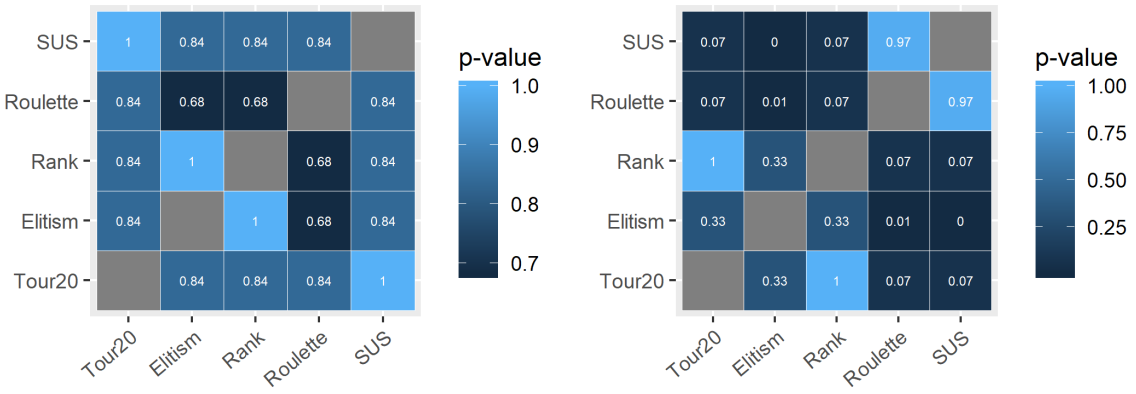
**Table 5.9:** Win-loss ranking.

| Simulation | $Sample_{in}$ | $Sample_{sel}$ | $Sample_{out}$ | Average |
|---|---|---|---|---|
| **Roulette** | -825(46) | 6007(28) | 516(38) | 1899.333(37.333) |
| **SUS** | -633(44) | 4082(34) | 384(37) | 1277.667(38.333) |
| **Rank** | 1416(36) | 2548(35) | 907(50) | 1623.667(40.333) |
| **Tour20** | 2811(35) | -2965(55) | 1914(36) | 586.667(42) |
| **Elitism** | -2769(52) | -9672(61) | -3721(53) | -5387.333(55.333) |

**Table 5.10:** P-value results using Iman and Davenport test with Finner's correction.

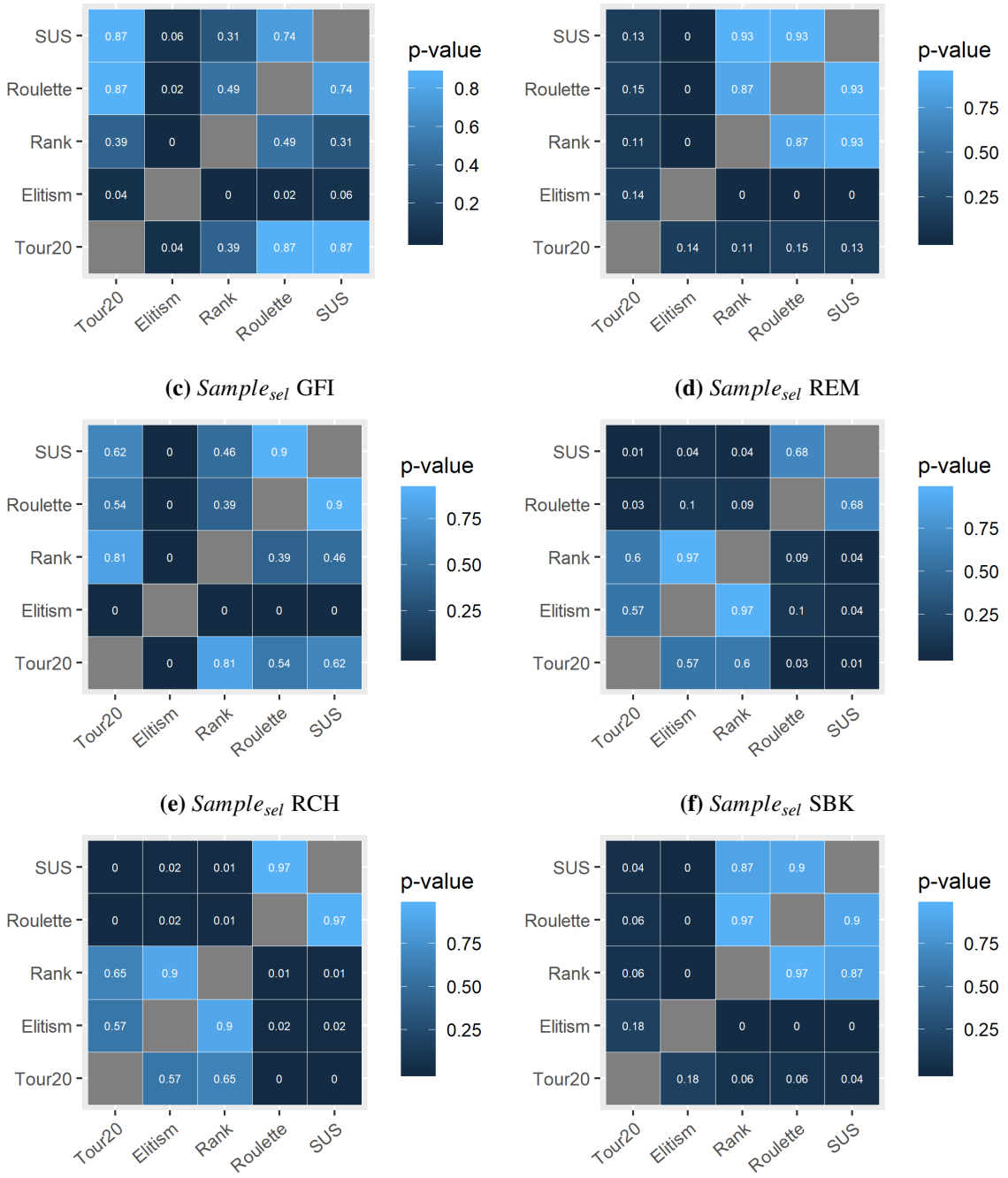
| | $Sample_{in}$ | | $Sample_{sel}$ | | $Sample_{out}$ | |
|---|---|---|---|---|---|---|
| **Share Code** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** |
| **agl** | 0.996869 | 0.997926 | 0.968058 | 0.986496 | 0.996869 | 0.997926 |
| **alsi** | 0.273701 | 0.558096 | **0.013071** | 0.063669 | 0.816455 | 0.900914 |
| **bil** | 0.260592 | 0.558096 | 0.137509 | 0.271664 | **0.001424** | **0.010631** |
| **gfi** | 0.061243 | 0.331357 | **0.043859** | 0.125890 | 0.261235 | 0.433174 |
| **imp** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| **inv-ned** | 0.180946 | 0.526934 | 0.062665 | 0.149377 | 0.114719 | 0.306185 |
| **inv-rem** | **0.000613** | **0.009157** | **0.005069** | **0.037397** | **0.000664** | **0.009917** |
| **inv-sbk** | 0.838586 | 0.916839 | 0.533236 | 0.681106 | 0.298664 | 0.446383 |
| **lon** | 0.386357 | 0.599740 | 0.826999 | 0.908594 | 0.986727 | 0.995495 |
| **ned** | 0.577753 | 0.762347 | 0.297231 | 0.444495 | **0.009361** | **0.045937** |
| **rch** | 0.052253 | 0.331357 | **0.000000** | **0.000000** | 0.136042 | 0.306204 |
| **rem** | 0.993990 | 0.997264 | 0.993990 | 0.995829 | 0.993655 | 0.997087 |
| **sab** | 0.972063 | 0.988579 | 0.991079 | 0.995684 | 0.479425 | 0.624401 |
| **sbk** | 0.597220 | 0.762347 | **0.029020** | 0.104557 | 0.179075 | 0.344813 |
| **sol** | 0.238314 | 0.558096 | 0.241309 | 0.404171 | 0.057819 | 0.200159 |

Note: **bold** values represent a significant difference.




**(a)** $Sample_{in}$ REM **(b)** $Sample_{sel}$ ALSI

**Figure 5.4:** Friedman pairwise comparison of p-values for results.




**(c)** $Sample_{sel}$ GFI

**(d)** $Sample_{sel}$ REM

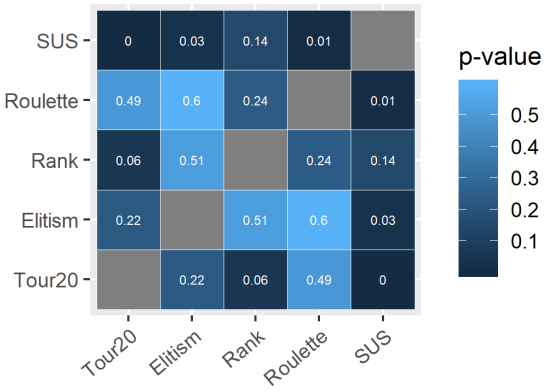
**(e)** $Sample_{sel}$ RCH

**(f)** $Sample_{sel}$ SBK

**(g)** $Sample_{out}$ BIL

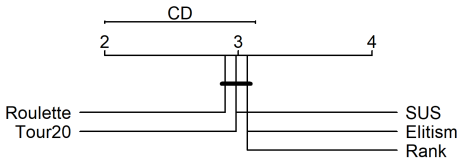
**(h)** $Sample_{out}$ INV-REM

**Figure 5.4:** Friedman pairwise comparison of p-values for results (Cont.).

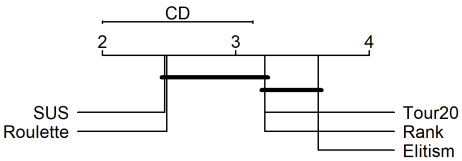



**(i)** $Sample_{out}$ NED

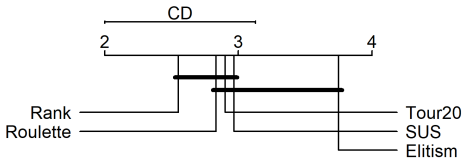
**Figure 5.4:** Friedman pairwise comparison of p-values for results (Cont.).



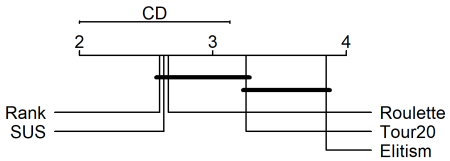

**(a)** $Sample_{in}$ REM



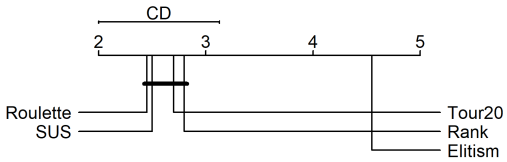

**(b)** $Sample_{sel}$ ALSI



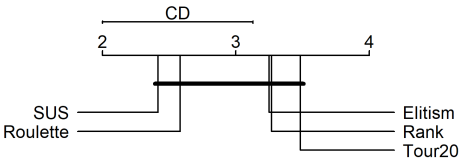

**(c)** $Sample_{sel}$ GFI




**(d)** $Sample_{sel}$ REM

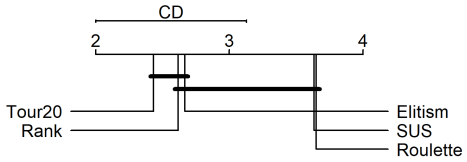



**(e)** $Sample_{sel}$ RCH

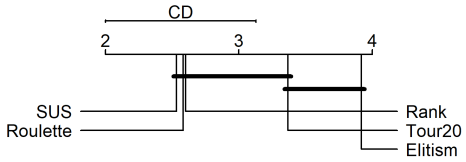



**(f)** $Sample_{sel}$ SBK

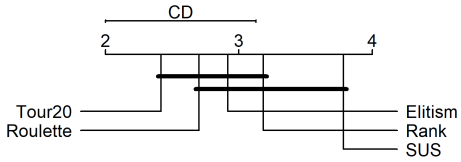
**Figure 5.5:** Critical difference plots for results.




**(g)** $Sample_{out}$ BIL




**(h)** $Sample_{out}$ INV-REM




**(i)** $Sample_{out}$ NED

**Figure 5.5:** Critical difference plots for results (Cont.).

**Table 5.11:** Standard deviation of $Sample_{out}$ profit per share analysis.

| Share Code | Tour20 | Elitism | Rank | Roulette | SUS |
|---|---|---|---|---|---|
| **agl** | 429.65 | **0.00** | 1366.84 | *1438.98* | 173.48 |
| **alsi** | 492.23 | **243.64** | 587.77 | 352.31 | *709.92* |
| **bil** | **2250.78** | 3313.30 | 2434.89 | *4563.56* | 3132.42 |
| **gfi** | 2203.90 | 2178.27 | *2440.20* | 2381.93 | **1479.91** |
| **imp** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| **inv-ned** | 1082.50 | *1520.60* | **614.99** | 717.24 | 1137.67 |
| **inv-rem** | 2060.71 | *2352.13* | 2083.13 | **1420.25** | 2125.70 |
| **inv-sbk** | *1371.16* | 1071.26 | 978.17 | **823.36** | 855.74 |
| **lon** | 5621.37 | **4301.88** | 5970.22 | *6287.76* | 5761.31 |
| **ned** | **1662.28** | 2220.73 | 2235.23 | *2398.69* | 1933.17 |
| **rch** | *214.31* | 190.31 | 163.68 | 170.82 | **138.16** |
| **rem** | 25.91 | **0.00** | **0.00** | *55.07* | 37.70 |
| **sab** | *754.10* | 558.17 | 635.53 | **339.99** | 447.05 |
| **sbk** | **178.84** | 233.41 | 205.99 | 355.38 | *489.47* |
| **sol** | 5961.40 | *7137.87* | 3791.01 | 4456.72 | **3689.58** |

Note: **bold** represents the lowest value or more preferred $\sigma$, and *italic* the highest or least preferred $\sigma$.

## 5.4 Fitness Function Analysis

A fitness function quantifies the performance of a solution. Each individual within the population represents a solution, in this case a trading rule.

A trading rule's primary objective is to determine when to buy, sell or hold a position to maximise

profit. This simple objective can be quantified by the total gain or loss when trading a given share. Return on investment (ROI) is a measure of return. ROI was used by Lam [154] as a fitness function and presented here as the *profit* fitness function. The ROI rewards a trading rule when it is profitable at the end of trading.

The second objective of a trading rule is profit consistency. A trading rule should at any point within the trading period be profitable. The ROI measures the difference between the initial investment and the final asset value at the end of the trading period. ROI does not measure the average investment return obtained during trading. Schoreels *et al.* [190] introduced accumulated (average) asset value (AAV) as the average daily ROI. The AAV is the second fitness function presented here as *avgProfit*. The AAV rewards a trading rule when it is consistently profitable over the course of trading.

The third objective of a trading rule is to be in the market when the market is trending up, and out of the market when the market is trending down. Furthermore, buying shares early on and returning a profit result in compounded capital gains, while buying shares and returning a loss result in compounded capital losses. Allen and Karjalainen [43] implemented a fitness function (AK) that measures the compounded excess return over the buy-and-hold strategy presented here as *ak*. AK rewards a trading rule when the rule has stock and the shares are moving up, or if the trading rule does not have stock and the shares are moving down.

Three simulations were run to determine which of the three fitness functions was preferred. The empirical process described in Section 5.2.1 was followed. The configuration presented in Table 5.3 was used with the selection strategy set to rank selection, and the fitness function defined as one of the three fitness functions: *ak*, *avgProfit*, or *profit*. The simulations were run independently across the share data defined in Table 5.2

Table 5.12 shows that the *profit* fitness function evolved trading rules that resulted in the highest mean profit in 8 of the 15 shares. The *ak* fitness function evolved trading rules that returned the highest mean profit for 6 of the 15 shares. The mean profit results clearly show that *avgProfit* evolved the worst performing trading rules.

A statistical comparison of the results presented in Table 5.14 found a significant difference between the $Sample_{out}$ results obtained for the BIL, INVREM, and NED datasets.

A post-hoc analysis was performed for those *Samples* that showed a significant difference. The p-value results are presented in Figure 5.6. A significant difference between *ak* and *profit* results was found when trading Gold Fields Ltd (GFI). A significant difference between *avgProfit* and *profit* results was found when trading BIL and Lonmin (LON).

Figure 5.7 confirms the significant difference between the results obtained by the trading rules

evolved using the *ak* and *profit* fitness functions when trading GFI. The critical difference plots also show that the results from the *avgProfit* simulation was significantly different from the results returned in the *profit* and *ak* simulations when trading Inverted Nedcor Ltd (INVNED) and LON.

The win-loss results presented in Table 5.13 show that the trading rules evolved using the *ak* fitness function performed consistently better than the trading rules evolved using *avgProfit* and *profit* when trading shares using the $Sample_{in}$ and $Sample_{out}$ datasets. Although *ak* resulted in the least $Sample_{sel}$ win-loss score, its overall average was still greater than *avgProfit* and *profit*. It is for these reasons that *ak* was used for the purposes of this study.

**Table 5.12:** Mean $Sample_{out}$ profit per share without fees.

| Share Code | AK | AvgProfit | Profit |
|---|---|---|---|
| **agl** | 25258.77 | *23849.50* | **25294.30** |
| **alsi** | **9400.57** | *9120.37* | 9301.53 |
| **bil** | **14327.53** | 14028.07 | *12058.30* |
| **gfi** | **-1260.00** | -1905.53 | *-2696.00* |
| **imp** | **16612.00** | *16168.43* | **16612.00** |
| **inv-ned** | **12891.57** | *11694.53* | 12705.50 |
| **inv-rem** | **5315.40** | *4206.37* | 5027.93 |
| **inv-sbk** | 5465.23 | *4810.83* | **5514.00** |
| **lon** | 18202.93 | *10723.07* | **21960.50** |
| **ned** | *-2399.90* | -1588.63 | **-1584.57** |
| **rch** | 2188.17 | *2159.83* | **2227.77** |
| **rem** | 9819.00 | *9618.50* | **9820.17** |
| **sab** | *5160.70* | **5571.83** | 5412.37 |
| **sbk** | 1555.27 | *1355.80* | **1613.17** |
| **sol** | 14196.20 | **16937.10** | *13690.60* |

Note: **bold** represents the highest value, and *italic* the lowest value.

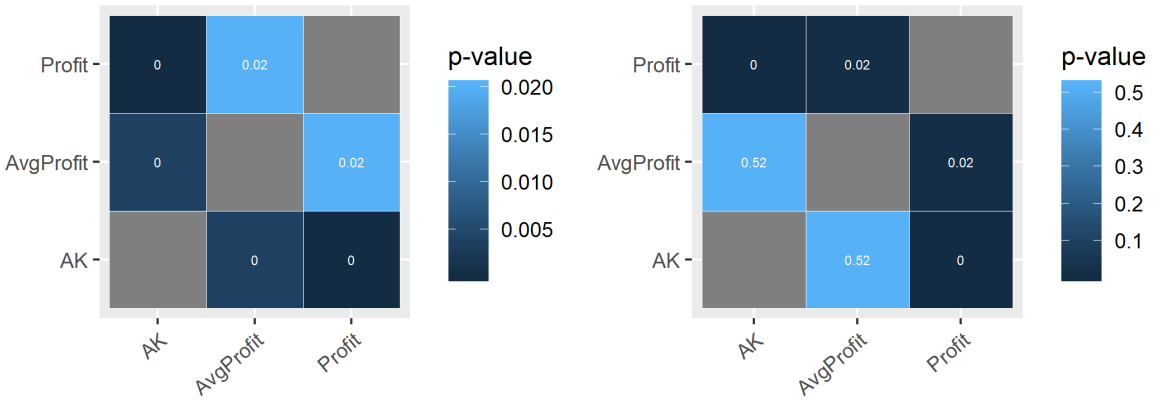
**Table 5.13:** Win-loss ranking.

| Simulation | $Sample_{in}$ | $Sample_{sel}$ | $Sample_{out}$ | Average |
|---|---|---|---|---|
| **Profit** | 437(27) | 957(27) | 74(28) | 489.333(27.333) |
| **AK** | 1288(31) | -1330(31) | 2083(24) | 680.333(28.667) |
| **AvgProfit** | -1725(31) | 373(31) | -2157(36) | -1169.667(32.667) |

**Table 5.14:** p-value results using Iman and Davenport test with Finner's correction.

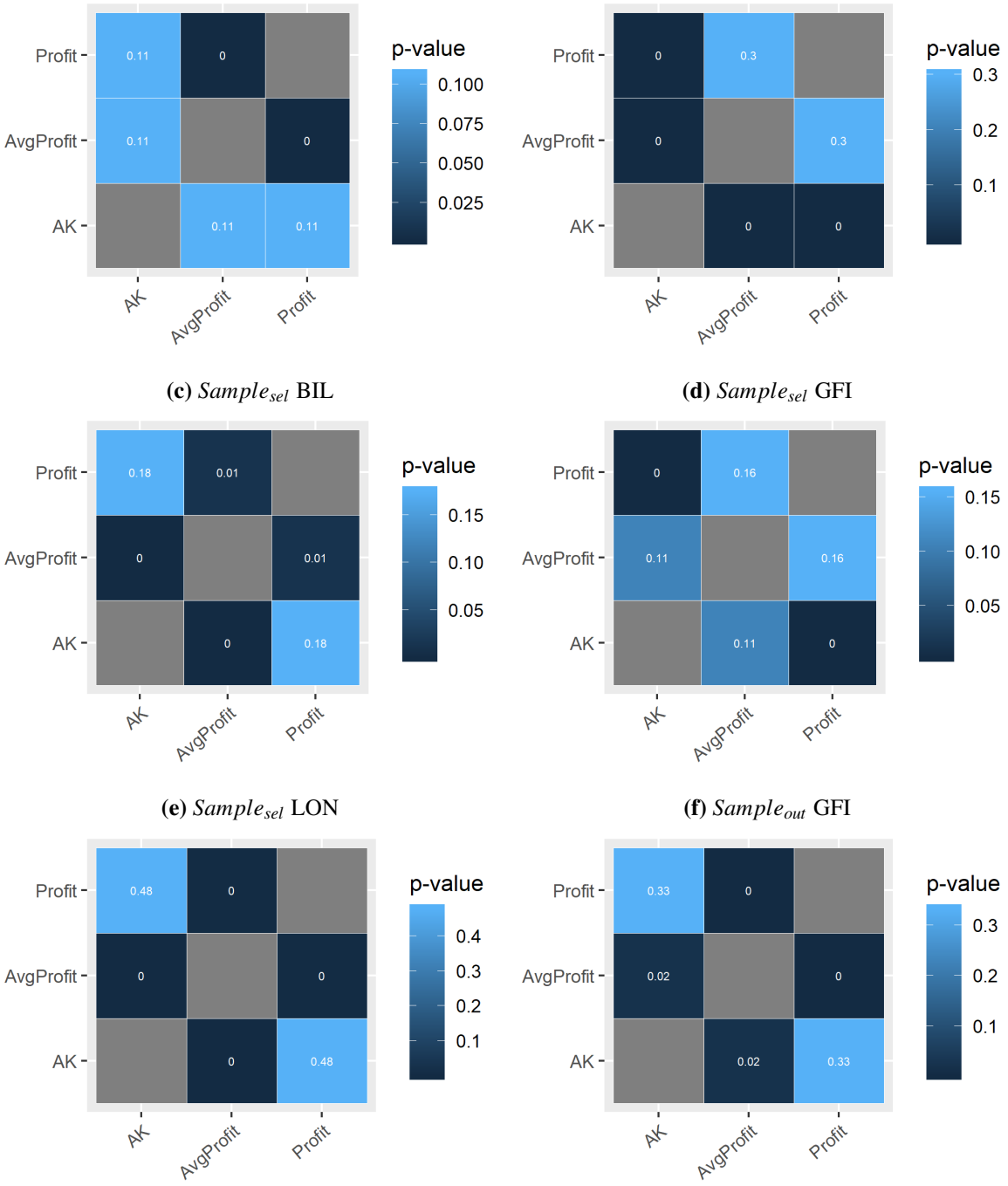
| | $Sample_{in}$ | | $Sample_{sel}$ | | $Sample_{out}$ | |
|---|---|---|---|---|---|---|
| **Share Code** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** |
| **agl** | 0.976113 | 0.976113 | 0.803868 | 0.847346 | 0.976113 | 0.976113 |
| **alsi** | **0.024323** | 0.088205 | **0.023022** | 0.083635 | 0.655771 | 0.798037 |
| **bil** | 0.594406 | 0.707867 | **0.003966** | **0.019673** | 0.103354 | 0.266339 |
| **gfi** | **0.000000** | **0.000000** | **0.000001** | **0.000011** | **0.007935** | **0.039049** |
| **imp** | 0.413409 | 0.573048 | 0.677553 | 0.757019 | 0.543053 | 0.769724 |
| **inv-ned** | **0.017460** | 0.084303 | 0.103354 | 0.238706 | **0.000140** | **0.002098** |
| **inv-rem** | 0.361860 | 0.569258 | 0.594406 | 0.707867 | 0.552059 | 0.769724 |
| **inv-sbk** | 0.907712 | 0.922156 | 0.475932 | 0.621544 | 0.098086 | 0.266339 |
| **lon** | 0.057826 | 0.138356 | **0.000178** | **0.001334** | **0.002153** | **0.016037** |
| **ned** | 0.399894 | 0.573048 | **0.023022** | 0.083635 | 0.878819 | 0.928502 |
| **rch** | 0.741116 | 0.804603 | 0.929981 | 0.929981 | 0.778175 | 0.871709 |
| **rem** | 0.729146 | 0.804603 | 0.843950 | 0.863341 | 0.929981 | 0.942093 |
| **sab** | **0.046682** | 0.133609 | 0.138774 | 0.273951 | **0.016515** | 0.060538 |
| **sbk** | **0.004457** | **0.032948** | 0.209314 | 0.356189 | 0.900404 | 0.930157 |
| **sol** | 0.276732 | 0.500542 | 0.441775 | 0.621544 | 0.106089 | 0.266339 |

Note: **bold** values represent a significant difference.




**(a)** $Sample_{in}$ GFI **(b)** $Sample_{in}$ SBK

**Figure 5.6:** Friedman pairwise comparison of p-values for results.




**(c)** $Sample_{sel}$ BIL




**(d)** $Sample_{sel}$ GFI




**(e)** $Sample_{sel}$ LON




**(f)** $Sample_{out}$ GFI




**(g)** $Sample_{out}$ INV-NED




**(h)** $Sample_{out}$ LON

**Figure 5.6:** Friedman pairwise comparison of p-values for results (Cont.).

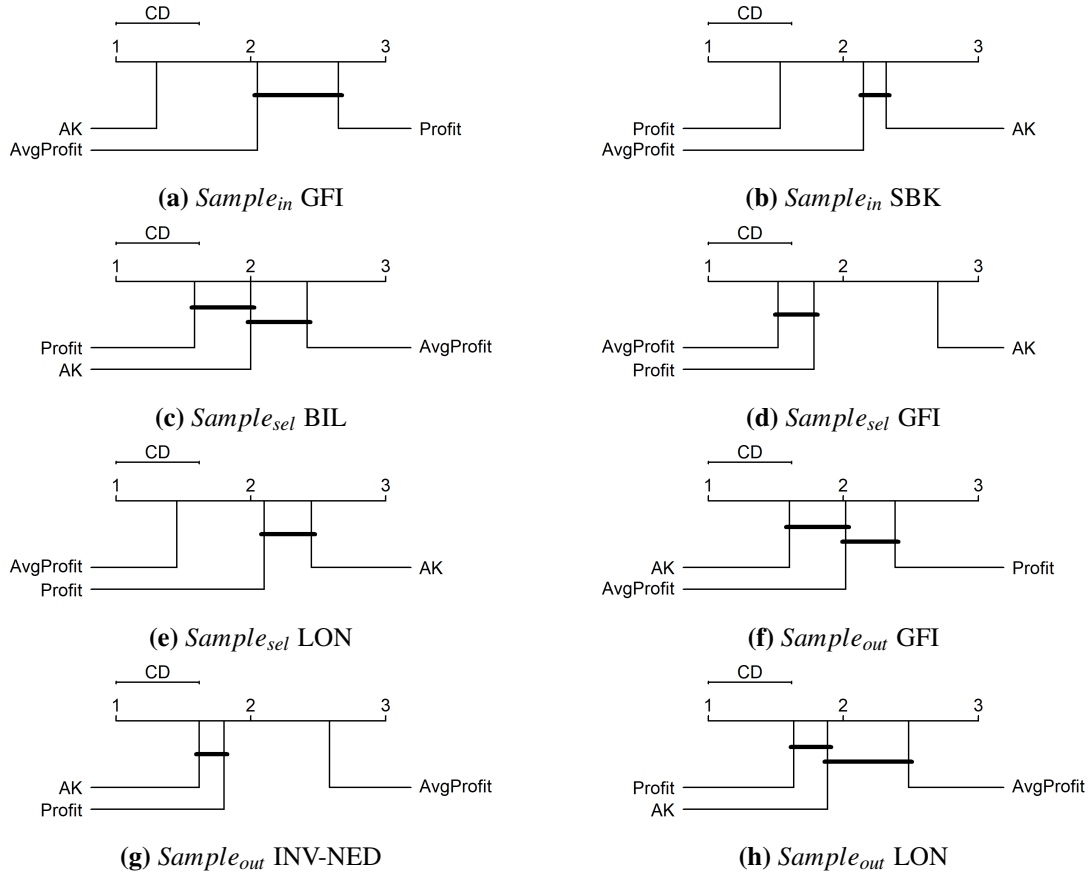



**Figure 5.7:** Critical difference plots for results.

## 5.5 Control Parameter Sensitivity Analysis

EAs and most machine learning algorithms require the setting of control parameter values. For GAs and GPs, these values include the probability of crossover, probability of mutation, the number of individuals within a population, and the number of generations.

There are two main approaches to parameter setting: parameter tuning and parameter control. The main difference is that parameter control is done as part of the EA's execution and parameter tuning is done before the EA is executed. An example of parameter control is a dynamic mutation probability control parameter. Initially, the probability of mutation can be high to promote exploration. Each subsequent generation decreases the mutation probability, reducing exploration and increasing optimisation. Parameter tuning relies on an appropriate parameter value set before execution.

Parameter tuning is a hard problem because there are a great number of possible parameter com-

binations [93]. Most EAs use parameters based on basic rules, for example, mutation should be low and crossover high. Allen and Karjalainen [43] admitted in their paper that they did not optimise the parameters chosen [211]. This thesis uses parameter tuning.

To select the most appropriate parameters, the response surface methodology (RSM) was used [93]. RSM is a collection of statistical techniques useful for modelling and analysis of problems in which responses are affected by several variables [194]. The relationship between parameters of a GP are not definite and must be estimated. RSM uses a sequence of experiments to obtain the relationship between parameters and their optimal values.

Box and Wilson [73] suggested using a second-degree polynomial model for the RSM. A second-degree polynomial model is complex, and should be performed on those parameters that have the largest impact on the results. To determine which parameters impact the results the most, a first-degree polynomial model is used.

The simplest first-degree polynomial is a fractional factorial design [175]. A factorial design is used to determine the relationship between the parameters and the effect that the parameters have on the results, i.e. if the factor is increased or decreased does the result of the simulation change, and is that change significant. A factorial design obtains the most appropriate information about the factors with the least amount of experimentation. With reference to GP, a factor refers to a control parameter. The control parameter tuning methodology proposed by Lima *et al.* [93] was used. Their methodology is based on $2^k$ factorial design to estimate the influence that a GP control parameter has on the results returned by evolved trading rules. Once the parameters with the largest impact were identified, they were fine-tuned using the empirical process described in Section 5.2.1.

Consider that an evolved trading rule is required to return a profit. The profit is the response variable. Factors are variables that affect the outcome of the response variable. Each factor can have several alternative values which Lima *et al.* refer to as *levels* [93]. The combination of factors and levels is large. Computing each alternative variation of factors and levels might not be the best possible use of computing resources. Therefore, it is important to reduce the number of factors and to consider only those that have a significant impact on the GP's performance.

$2^k$ factorial design focuses on the minimum and maximum factor levels for each factor $k$. This is based on the assumption that the effect of the factor is unidirectional [93]. The intent is to determine if the difference in response variables of two levels of a factor warrants further investigation. Furthermore, $2^k$ factorial design also analyses the interaction of the factors to determine if the change in one factor affects another factor. $2^k$ factorial design requires one simulation run per factor. However, GPs are stochastic and one run is not sufficient to determine the effect of a factor. Lima *et al.* altered $2^k$ factorial

design to include $r$, where $r$ denotes the number of runs.

The empirical process presented in Section 5.2.1 assumed the normal distribution and showed that the results of EAs are generally skewed. Therefore, a comparison of the results of EAs requires non-parametric statistical tests as opposed to standard parametric statistical tests. To ensure that results generated by the GPs were caused by a factor and not the naturally skewed results a multiplicative factorial model was used as proposed by Lima *et al.* [93].

$2^k r$ factorial design is performed in four steps: compute the effect each factor has on the outcome of the simulation, allocate the response variable variation explained by each factor, determine the confidence interval of the effects to determine which are significant, and verify if the technique's assumptions are not violated. Each step is discussed in more detail in the paragraphs that follow.

Computation of the effects of each factor or factor interaction has on the change in the response variables requires a list of factors. For this thesis four factors were identified, namely the number of individuals within a population (factor A), the maximum number of generations (factor B), the maximum tree size quantified as two values, i.e. tree depth and number of nodes (factor C), and the probability of mutation (factor D). Each factor is required to have an upper and lower bound. The factor boundaries are defined as a factor configuration set. One upper and lower value per factor.

Deciding on the upper and lower bounds is one of the most difficult tasks when building a $2^k r$ factorial design [93]. For example, should the mutation probability boundaries be set at [0%, 100%], or [10%, 50%]? No mutation takes place when the probability of mutation is set at 0%, while no crossover takes place when the probability of mutation is set at 100%. Therefore, one set of simulations could test the effect that two different degrees of mutation probability have on the response variable, while another simulation could test if no probability of mutation or a low mutation probability effects the change in the response variable. For this reason, two different factor configuration sets were defined as presented in Table 5.15.

**Table 5.15:** Factor boundary values for $2^k r$ factorial design.

| Parameter | Config. 1 | | Config. 2 | |
|---|---|---|---|---|
| | Lower bound | Upper bound | Lower bound | Upper bound |
| **Population Size (A)** | 20 | 1000 | 100 | 500 |
| **Number of Generations (B)** | 5 | 500 | 20 | 50 |
| **Tree Size [Depth, Nodes] (C)** | 3 , 30 | 20, 300 | 3, 20 | 20, 200 |
| **Mutation Probability (D)** | 0.1 | 0.4 | 0 | 0.5 |

To compute the effect that each factor has on the response variable, each factor must be evaluated $r$ times. A simulation was run 30 times ($r$ is 30) for each share dataset and each control parameter combination. Each control parameter combination is a distinct combination of factors and levels defined by the factor configuration set. The combinations of factors and response variable results are recorded in a $2^4 30$ sign matrix as defined by Lima *et al.* [93]. To illustrate how the effects are computed consider the $2^2 3$ signed matrix is presented in Table5.16.

The $2^2 3$ (2 factors of 2 levels each with 3 repetitions) signed matrix defines two factors namely, **A** and **B**. The first column **I** of the matrix always consists of 1's. The second and third columns **A** and **B** contain every possible combination of -1 and 1. Column **AB** is the product of the entries in columns **A** and **B**. Each line of the matrix represents a simulation. Each simulation is configured with control parameters represented by the factors **A** and **B**. If the column contains a -1, then the lower factor bound is set as the control parameter for that factor. The simulation is repeated three times. The response variable is the result of a simulation run. The response variables of twelve simulation runs (2 factors $\times$ 2 levels $\times$ 3 repetitions) are presented in column $y$. The average of the results is recorded in column $\bar{y}$.

Next, the sum of the products of the entries in column **I** and the mean results in column $\bar{y}$ is recorded under column **I** (i.e., $1 \times 15 + 1 \times 24 + 1 \times 48 + 1 \times 77 = 164$). The process is repeated for columns **A**, **B**, and **AB**. The sums are then divided by $2^2$ to give the effect each factor has on the result recorded in column $y$. Once the effects have been calculated, the factorial model is used to estimate the response variable returned by a given combination of factors. The estimated response variable is compared to the actual response variable recorded in column $y$ to estimate the variance of the error. The sum of the squared errors is referred to as the experimental error. The experimental error is a result of experimental factors such as the value of the random seed used to generate random numbers and not the control parameters. The total factor effect less the experimental error is referred to as the explained effect. A factor's importance is measured by the proportion of the total variation in the response variable it explains. A factor or combination of factors with the higher response variation is deemed more influential in the outcome of the response variable.

**Table 5.16:** Example of a sign matrix for a $2^2 3$ factorial design.

| I | A | B | AB | $y$ | Mean $\bar{y}$ |
|---|---|---|---|---|---|
| 1 | -1 | -1 | 1 | (15, 18, 12) | 15 |
| 1 | -1 | 1 | -1 | (25, 28, 19) | 24 |
| 1 | 1 | -1 | -1 | (45, 48, 51) | 48 |
| 1 | 1 | 1 | 1 | (75, 75, 81) | 77 |
| 164 | 86 | 38 | 20 | | *Total* |
| 41 | 21.5 | 9.5 | 5 | | $Total/2^2$ |

Table 5.17 and Table 5.18 report the importance of each factor and factor interaction as a contribution percentage, calculated as its coefficient in the $2^4 30$ sign matrix model. The results in Table 5.17 and Table 5.18 were calculated using simulations run with the factor configuration set 1 and configuration set 2, respectively. Both results show that the simulations run using the INVNED, Inverted Standard Bank Group Ltd (INVSBK), and RCH datasets have the largest explained variation percentages.

The results presented in Table 5.17 show that the factors with the largest influence on the GP's performance when run using the INVNED, INVSBK, and RCH datasets were A, B, D, AB, and AD. The individual factors A, B, and D had a larger influence on the response variable than the combinations of AB and AD. The overall influencers were therefore, population size, number of generations and mutation probability.

The results presented in Table 5.18 show that factor A, D, and AD had the greatest influence on the response variable returned by the simulations using the INVNED, INVSBK, and RCH datasets. Factors A and D represent the population size and the mutation probability respectively. The population size factor was configured with a lower bound population size of 100 individuals and an upper bound population size of 500 individuals. The mutation probability was configured with a lower bound value of 0% and an upper bound value of 50%. These configured factors meant one quarter of the simulations ran with a population size of 100 individuals and no mutation.

Mutation drives exploration, without mutation a population will rapidly converge. Simulations that use mutation have a greater probability of evolving better solutions than those simulations that do not use mutation. Without mutation constantly introducing new genetic material, an alternative source of genetic diversity is required. Increasing the number of individuals within the population increases the genetic diversity. Therefore, without mutation, larger populations have a greater probability of converging on a better solution than smaller populations. The upper and lower bounds set for factor con-

figuration set 2 resulted in at least one quarter of the simulations converging much faster than the other simulations, resulting in larger influence percentages of factor A, D and AD presented in Table 5.18 than the results presented in Table 5.17.

The $2^4 30$ factorial design results showed that the individual factors population size, mutation probability and the number of generations influenced the response variable the most. The empirical process described in Section 5.2.1 was used to determine the most appropriate control parameter values for the three identified factors.

The results of the empirical process is presented in the subsections that follow. Subsection 5.5.1 explores the impact that the mutation probability had on the result. Subsection 5.5.2 explores the impact that the population size had on the result. Subsection 5.5.3 investigates the effect the maximum number of generations had on the result.

### 5.5.1 Mutation Probability Sensitivity Analysis

Allen and Karjalainen [43] used a mutation probability of 30%. The simulations performed found that a mutation probability of between 30% and 40% was adequate.

Ten simulations were run, each with a different probability of mutation. Each simulation implemented the same configuration presented in Table 5.3. Rank selection was used as the selection strategy. The first simulation used a mutation probability of 0% implying that no mutation took place. The second simulation used a mutation probability of 10%, the next 20%, with each subsequent simulation increasing the mutation probability by 10 until 90%.

The best profit was recorded for each of the simulation runs, and the mean profits are summarised in Table 5.19. The mean profit results did not show a clear preferred simulation. Table 5.20 presents the p-values from the Iman Davenport test [142]. The statistical test applied to the results reject the null-hypothesis for a majority of the share datasets. The only $Sample_{out}$ shares where no significant difference was found between the mean profit results were Anglo American Plc (AGL), GFI, Impala Platinum (IMP), Remgro Ltd (REM), SAB, SBK, and Sasol Ltd (SOL).

The results of a post-hoc analysis using the Friedman aligned ranks test are presented in Figure 5.8. The pairwise results found a significant difference between a mutation probability of less than 10% and a mutation probability greater than 20%.

A comparison of the mean deviation ($\sigma$) values presented in Table 5.21 confirms the significant differences between a mutation probability of less than 10% and a mutation probability greater than 20%. The $\sigma$ values for a mutation probability of 0% is zero. No exploration occurs when mutation is zero, and all the individuals tend to converge on the same $Sample_{out}$ solution. Each significantly

**Table 5.17:** Parameter sensitivity configuration 1 results of the $2^4 30$ factorial design for symbolic regression. The significant effects and corresponding percentage of explained variation are presented.

| Parameter | AGL | ALSI | BIL | GFI | IMP | INV-NED | INV-REM | INV-SBK | LON | NED | RCH | REM | SAB | SBK | SOL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Population Size (A)** | 1.2032 | 6.7100 | 0.6465 | 0.8002 | 37.3705 | 46.8630 | 8.6431 | 44.7952 | 1.5829 | 3.1593 | 52.9886 | 32.0028 | 1.0685 | 1.9616 | 0.5451 |
| **Number of Generations (B)** | 0.0391 | 1.2623 | 0.6658 | 0.0427 | 3.8152 | 13.0360 | 6.3266 | 7.4509 | 0.0876 | 0.7782 | 3.6169 | 3.2111 | 2.0019 | 0.0247 | 0.5758 |
| **Tree Size (C)** | 0.0355 | 0.1596 | 0.0175 | 0.0000 | 0.0109 | 0.0079 | 0.0114 | 0.0050 | 0.0058 | 0.0006 | 0.0011 | 0.0995 | 0.1769 | 0.2764 | 0.4010 |
| **Mutation Probability (D)** | 0.4882 | 0.0547 | 0.3906 | 0.4682 | 1.0585 | 3.1551 | 0.8600 | 3.4595 | 0.9915 | 1.2534 | 1.8550 | 0.7551 | 0.4496 | 0.0601 | 0.0022 |
| **AB** | 0.0662 | 0.3041 | 0.6732 | 0.1650 | 3.3741 | 0.4534 | 1.0808 | 4.0688 | 0.0857 | 0.1166 | 0.0007 | 5.7375 | 1.8435 | 1.2350 | 0.5661 |
| **AC** | 0.0355 | 0.0764 | 0.0163 | 0.0000 | 0.0001 | 0.0023 | 0.0893 | 0.0005 | 0.0372 | 0.0024 | 0.0020 | 0.0995 | 0.1565 | 0.3046 | 0.4104 |
| **AD** | 0.5751 | 0.1233 | 0.3962 | 0.3726 | 0.8326 | 3.3332 | 0.0316 | 3.3140 | 1.0035 | 0.8173 | 0.1816 | 1.9203 | 0.3762 | 0.8558 | 0.0011 |
| **BC** | 0.0355 | 0.0641 | 0.3392 | 0.0181 | 0.0109 | 0.0070 | 0.2913 | 0.0037 | 0.0091 | 0.0073 | 0.0001 | 0.0995 | 0.1254 | 0.2577 | 0.0050 |
| **BD** | 0.0289 | 0.0714 | 0.0002 | 0.1696 | 0.2248 | 2.8551 | 0.1058 | 2.7793 | 0.0444 | 0.0759 | 0.1221 | 0.7551 | 0.8135 | 0.0526 | 0.2738 |
| **CD** | 0.0355 | 0.0059 | 0.0568 | 0.0095 | 0.0109 | 0.0035 | 0.0015 | 0.0031 | 0.0040 | 0.0156 | 0.0173 | 0.0741 | 0.6505 | 0.2647 | 0.0352 |
| **ABC** | 0.0355 | 0.0169 | 0.3339 | 0.0111 | 0.0001 | 0.0019 | 0.0180 | 0.0010 | 0.0450 | 0.0002 | 0.0074 | 0.0995 | 0.1083 | 0.2849 | 0.0050 |
| **ABD** | 0.0121 | 0.0120 | 0.0004 | 0.0705 | 0.1280 | 1.7329 | 0.1853 | 2.7781 | 0.0292 | 0.1286 | 1.5014 | 1.9203 | 0.7138 | 0.8267 | 0.2806 |
| **ACD** | 0.0355 | 0.0399 | 0.0590 | 0.0155 | 0.0001 | 0.0004 | 0.0499 | 0.0013 | 0.0322 | 0.0029 | 0.0030 | 0.0741 | 0.6108 | 0.2922 | 0.0380 |
| **BCD** | 0.0355 | 0.0498 | 0.0449 | 0.0098 | 0.0109 | 0.0030 | 0.1116 | 0.0022 | 0.0068 | 0.0168 | 0.0082 | 0.0741 | 0.5476 | 0.2493 | 0.1408 |
| **ABCD** | 0.0355 | 0.1198 | 0.0430 | 0.0013 | 0.0001 | 0.0002 | 0.0223 | 0.0021 | 0.0396 | 0.0034 | 0.0002 | 0.0741 | 0.5113 | 0.2761 | 0.1408 |
| **Explained** | 2.6965 | 9.0704 | 3.6835 | 2.1540 | 46.8479 | 71.4547 | 17.8284 | 68.6645 | 4.0045 | 6.3785 | 60.3054 | 46.9964 | 10.1544 | 7.2224 | 3.4208 |
| **Experimental Error** | 97.3035 | 90.9296 | 96.3165 | 97.8460 | 53.1521 | 28.5453 | 82.1716 | 31.3355 | 95.9955 | 93.6215 | 39.6946 | 53.0036 | 89.8456 | 92.7776 | 96.5792 |

**Table 5.18:** Parameter sensitivity configuration 2 results of the $2^4 30$ factorial design for symbolic regression. The significant effects and corresponding percentage of explained variation are presented.

| Parameter | agl | alsi | bil | gfi | imp | inv-ned | inv-rem | inv-sbk | lon | ned | rch | rem | sab | sbk | sol |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Population Size (A)** | 2.1153 | 0.8657 | 0.2608 | 0.6671 | 1.4123 | 13.3120 | 3.5219 | 10.0390 | 0.0042 | 0.1122 | 2.0783 | 10.6248 | 1.2676 | 3.8100 | 0.3998 |
| **Number of Generations (B)** | 0.4931 | 0.2305 | 0.0631 | 0.0012 | 0.0032 | 0.2112 | 0.3194 | 0.0416 | 0.3370 | 0.0604 | 0.2682 | 0.0007 | 1.4487 | 0.0006 | 0.0371 |
| **Tree Size (C)** | 0.0326 | 0.1930 | 0.0726 | 0.0000 | 0.0032 | 0.0098 | 0.2028 | 0.0000 | 0.0194 | 0.7840 | 0.0006 | 0.0206 | 0.0124 | 0.0528 | 0.5386 |
| **Mutation Probability (D)** | 0.1616 | 0.2686 | 3.8714 | 2.3759 | 1.1558 | 26.5728 | 6.8873 | 13.7922 | 0.5360 | 10.2680 | 37.9090 | 6.6135 | 1.1023 | 0.3598 | 1.6823 |
| **AB** | 0.0326 | 0.0153 | 0.0015 | 0.0001 | 0.0032 | 0.0183 | 0.0262 | 0.0008 | 0.1235 | 0.0328 | 0.3136 | 0.0003 | 0.3396 | 0.0087 | 0.0344 |
| **AC** | 0.4931 | 0.8558 | 0.0096 | 0.0007 | 0.0032 | 0.0162 | 0.2026 | 0.0000 | 0.8154 | 0.0161 | 0.2482 | 0.0185 | 0.1092 | 0.0080 | 0.5282 |
| **AD** | 0.2312 | 0.5045 | 1.2610 | 0.6671 | 5.9405 | 1.9041 | 0.0028 | 7.1848 | 0.0003 | 0.1122 | 3.5436 | 6.3997 | 1.2676 | 2.7463 | 0.7347 |
| **BC** | 0.0326 | 0.0213 | 0.0003 | 0.0008 | 0.0032 | 0.0007 | 0.2474 | 0.0000 | 0.1680 | 0.0020 | 0.0253 | 0.0000 | 0.0066 | 0.0167 | 0.0560 |
| **BD** | 0.4931 | 0.2305 | 0.0631 | 0.0012 | 0.0032 | 0.2112 | 0.3194 | 0.0416 | 0.3370 | 0.0604 | 0.2682 | 0.0007 | 1.4487 | 0.0006 | 0.0371 |
| **CD** | 0.0326 | 0.1930 | 0.0726 | 0.0000 | 0.0032 | 0.0098 | 0.2028 | 0.0000 | 0.0194 | 0.7840 | 0.0006 | 0.0206 | 0.0124 | 0.0528 | 0.5386 |
| **ABC** | 0.4931 | 0.1120 | 0.0017 | 0.0000 | 0.0032 | 0.0004 | 0.1811 | 0.0004 | 0.2456 | 0.0001 | 0.0405 | 0.0001 | 0.1750 | 0.0087 | 0.0526 |
| **ABD** | 0.0326 | 0.0153 | 0.0015 | 0.0001 | 0.0032 | 0.0183 | 0.0262 | 0.0008 | 0.1235 | 0.0328 | 0.3136 | 0.0003 | 0.3396 | 0.0087 | 0.0344 |
| **ACD** | 0.4931 | 0.8558 | 0.0096 | 0.0007 | 0.0032 | 0.0162 | 0.2026 | 0.0000 | 0.8154 | 0.0161 | 0.2482 | 0.0185 | 0.1092 | 0.0080 | 0.5282 |
| **BCD** | 0.0326 | 0.0213 | 0.0003 | 0.0008 | 0.0032 | 0.0007 | 0.2474 | 0.0000 | 0.1680 | 0.0020 | 0.0253 | 0.0000 | 0.0066 | 0.0167 | 0.0560 |
| **ABCD** | 0.4931 | 0.1120 | 0.0017 | 0.0000 | 0.0032 | 0.0004 | 0.1811 | 0.0004 | 0.2456 | 0.0001 | 0.0405 | 0.0001 | 0.1750 | 0.0087 | 0.0526 |
| **Explained** | 5.6625 | 4.4947 | 5.6908 | 3.7156 | 8.5471 | 42.3022 | 12.7710 | 31.1017 | 3.9584 | 12.2832 | 45.3238 | 23.7181 | 7.8205 | 7.1073 | 5.3104 |
| **Experimental Error** | 94.3375 | 95.5053 | 94.3092 | 96.2844 | 91.4529 | 57.6978 | 87.2290 | 68.8983 | 96.0416 | 87.7168 | 54.6762 | 76.2819 | 92.1795 | 92.8927 | 94.6896 |

different result set was analysed in detail to find the most adequate mutation probability for this thesis.

**Table 5.19:** Mean $Sample_{out}$ profit per share.

| Share Code | 0% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% |
|---|---|---|---|---|---|---|---|---|---|---|
| **agl** | 25991.00 | 25991.00 | 25981.73 | 25258.77 | **26369.63** | 25991.00 | 25430.80 | 26007.33 | *24533.67* | 25383.70 |
| **alsi** | 9234.00 | 9265.23 | 9390.07 | 9400.57 | 9413.50 | 9261.17 | **9449.50** | 9097.87 | *8936.23* | 9059.63 |
| **bil** | **16559.00** | 16194.20 | 13301.83 | 14327.53 | 12275.53 | 12254.70 | 12997.63 | *11134.63* | 12831.83 | 13125.20 |
| **gfi** | 0.00 | -170.80 | -369.00 | *-1260.00* | -784.33 | -760.17 | -519.90 | -292.17 | 60.43 | **983.73** |
| **imp** | 16612.00 | 16612.00 | **16711.40** | 16612.00 | 16612.00 | **16711.40** | 16612.00 | 16612.00 | 16215.50 | *16210.90* |
| **inv-ned** | *7878.00* | 10156.00 | 12391.57 | 12891.57 | 12642.10 | 12615.23 | 12843.53 | **13365.70** | 13281.00 | 12839.27 |
| **inv-rem** | *2037.00* | 3623.37 | 4928.70 | 5315.40 | 4451.57 | 4774.47 | 5941.67 | 5442.23 | **6484.17** | 5711.57 |
| **inv-sbk** | *1703.00* | 4662.43 | 5153.70 | 5465.23 | 5326.30 | 5895.87 | **6016.97** | 5876.40 | 5947.90 | 5908.63 |
| **lon** | 17517.00 | 19475.97 | **19543.53** | 18202.93 | 18408.33 | 19099.03 | 18917.67 | 17854.13 | 12252.50 | *11824.10* |
| **ned** | **0.00** | -1649.77 | -2141.17 | -2399.90 | -1372.43 | -3137.40 | -2350.50 | *-3555.03* | -2784.33 | -2237.77 |
| **rch** | *1977.00* | 2193.23 | 2213.03 | 2188.17 | 2161.23 | 2213.40 | 2197.40 | 2187.53 | **2274.23** | 2205.80 |
| **rem** | 9819.00 | 9819.00 | 9854.60 | 9819.00 | 9819.00 | 9821.77 | 9838.50 | **9861.83** | *9601.53* | 9836.53 |
| **sab** | 5500.00 | 5481.57 | **5536.77** | 5160.70 | 5337.63 | 5107.20 | *4758.10* | 4941.17 | 5221.93 | 5318.30 |
| **sbk** | 1538.00 | 1657.90 | 1562.47 | 1555.27 | 1644.30 | 1502.67 | 1673.83 | *1477.23* | **1677.50** | 1641.13 |
| **sol** | **16459.00** | 16363.27 | 16453.27 | 14196.20 | 15614.47 | 12682.87 | 13190.80 | 14258.40 | 13477.30 | *10050.43* |

Note: **bold** represents the highest value, and *italic* the lowest value.

**Table 5.20:** p-value results using Iman and Davenport test with Finner's correction.

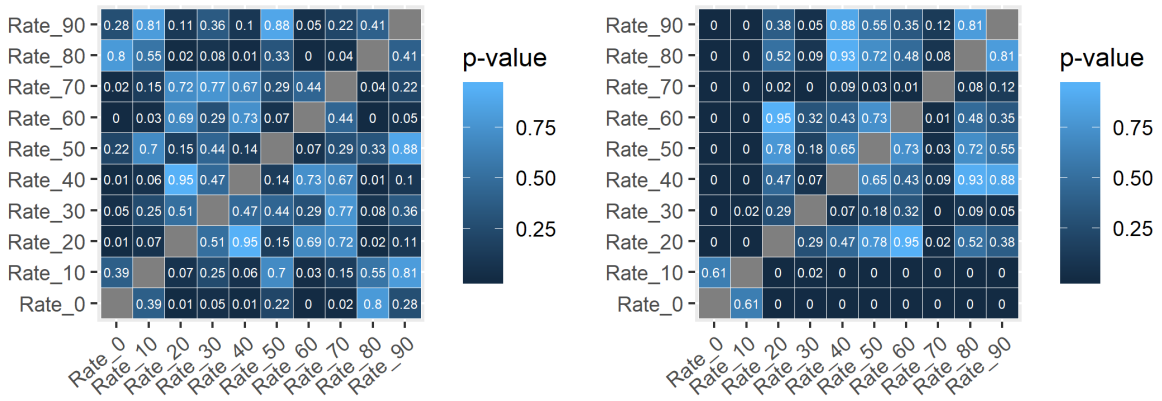
| | $Sample_{in}$ | | $Sample_{sel}$ | | $Sample_{out}$ | |
|---|---|---|---|---|---|---|
| **Share Code** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** |
| **agl** | 0.999064 | 0.999431 | 0.999669 | 0.999813 | 0.999539 | 0.999851 |
| **alsi** | **0.004895** | **0.006669** | **0.000000** | **0.000000** | **0.010590** | **0.022556** |
| **bil** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **gfi** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | 0.522826 | 0.635387 |
| **imp** | 0.999988 | 0.999988 | 0.999956 | 0.999956 | 0.999993 | 0.999993 |
| **inv-ned** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **inv-rem** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **inv-sbk** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **lon** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.012128** | **0.022619** |
| **ned** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **rch** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.001752** | **0.004374** |
| **rem** | 0.692617 | 0.743633 | 0.947281 | 0.966477 | 0.999517 | 0.999851 |
| **sab** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | 0.146689 | 0.232322 |
| **sbk** | 0.054678 | 0.067874 | 0.514034 | 0.594251 | 0.834941 | 0.894792 |
| **sol** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | 0.429543 | 0.569142 |

Note: **bold** values represent a significant difference.

**Table 5.21:** Standard deviation of $Sample_{out}$ profit per share analysis.

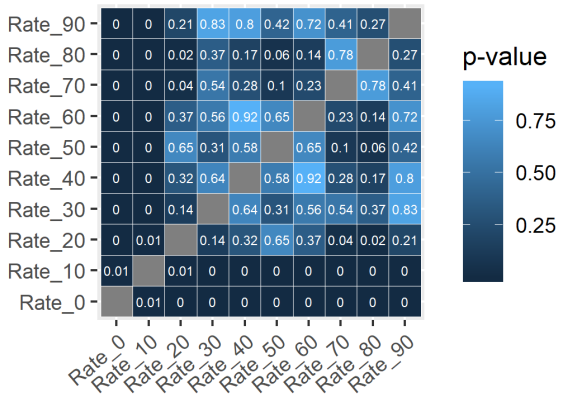
| Share Code | 0% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% |
|---|---|---|---|---|---|---|---|---|---|---|
| **agl** | **0.00** | **0.00** | 110.78 | 1366.84 | 706.78 | **0.00** | 1078.37 | 31.58 | *2753.02* | 1174.11 |
| **alsi** | **0.00** | 238.42 | 272.67 | *587.77* | 425.90 | 171.87 | 344.50 | 526.40 | 551.96 | 549.98 |
| **bil** | **0.00** | 608.00 | 3126.13 | 2434.89 | 3223.83 | *3791.31* | 3237.02 | 3108.51 | 2683.76 | 1950.33 |
| **gfi** | **0.00** | 1806.91 | 2130.13 | 2440.20 | *2866.33* | 2790.36 | 2781.47 | 2015.67 | 2317.50 | 2241.45 |
| **imp** | **0.00** | **0.00** | 192.17 | **0.00** | **0.00** | 192.17 | **0.00** | **0.00** | 766.57 | *775.46* |
| **inv-ned** | **0.00** | *2381.33* | 1115.73 | 614.99 | 646.78 | 661.89 | 617.48 | 1015.71 | 1088.93 | 741.37 |
| **inv-rem** | **0.00** | 1951.02 | 1699.41 | 2083.13 | 2206.62 | 2212.24 | 2236.51 | 1996.56 | 1966.99 | *2243.00* |
| **inv-sbk** | **0.00** | 1495.13 | *1530.09* | 978.17 | 825.03 | 486.81 | 894.35 | 350.68 | 519.86 | 584.34 |
| **lon** | **0.00** | 3918.96 | 4418.50 | 5970.22 | 5030.53 | 7017.43 | 7841.58 | 8660.73 | 8668.00 | *8899.77* |
| **ned** | **0.00** | 2189.47 | 2204.63 | 2235.23 | 1864.80 | 1544.16 | 1902.43 | 1873.44 | *2688.11* | 2215.90 |
| **rch** | **0.00** | 160.64 | 173.35 | 163.68 | *207.42* | 158.29 | 166.05 | 191.88 | 147.84 | 159.44 |
| **rem** | **0.00** | **0.00** | 66.45 | **0.00** | **0.00** | 10.68 | 37.70 | 115.30 | *501.24* | 120.12 |
| **sab** | **0.00** | 166.61 | 416.45 | 635.53 | 653.17 | 812.44 | *1263.59* | 856.37 | 861.96 | 668.01 |
| **sbk** | **0.00** | 228.53 | 229.99 | 205.99 | 234.44 | 198.31 | 310.56 | 316.04 | *558.77* | 318.48 |
| **sol** | **0.00** | 395.02 | 366.17 | 3791.01 | 3661.59 | 6572.80 | 6796.61 | 6097.17 | 6834.80 | *8376.04* |

Note: **bold** represents the lowest value or more preferred $\sigma$, and *italic* the highest or least preferred $\sigma$.

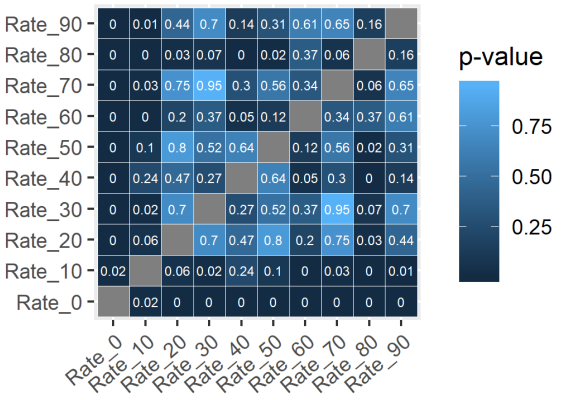



**(a)** $Sample_{out}$ ALSI **(b)** $Sample_{out}$ BIL

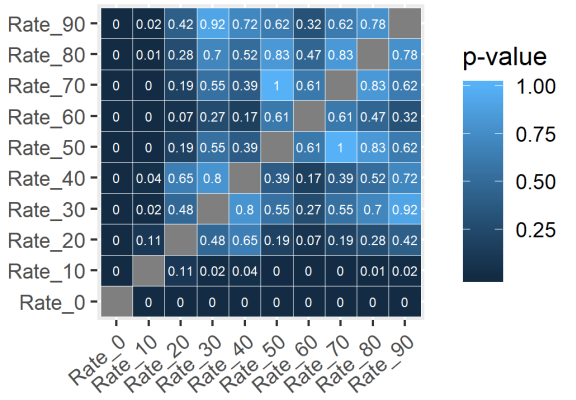
**Figure 5.8:** Friedman pairwise comparison of p-values for results.



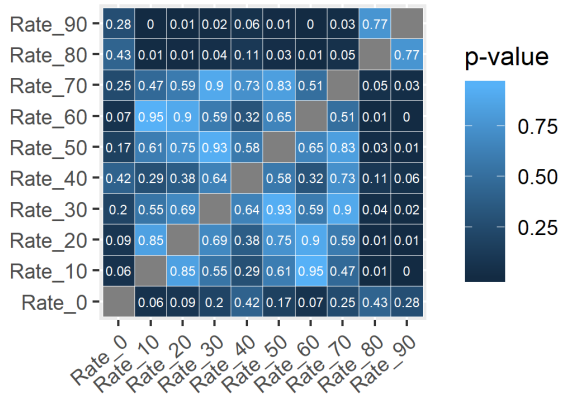

**(c)** $Sample_{out}$ INV-NED



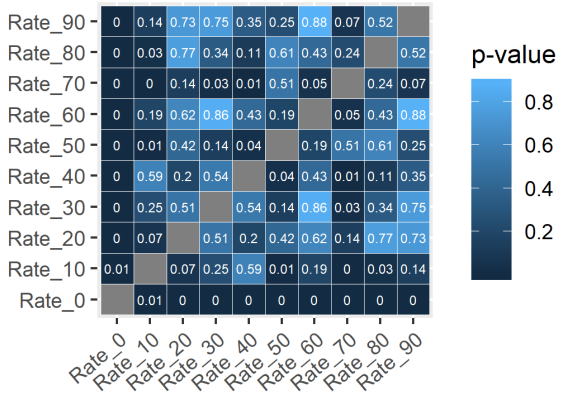

**(d)** $Sample_{out}$ INV-REM



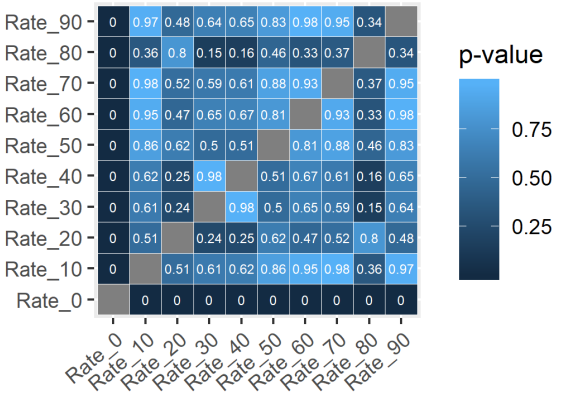

**(e)** $Sample_{out}$ INV-SBK




**(f)** $Sample_{out}$ LON




**(g)** $Sample_{out}$ NED




**(h)** $Sample_{out}$ RCH

**Figure 5.8:** Friedman pairwise comparison of p-values for results (Cont.).

A comparison of the $Sample_{out}$ results returned by the trading rules evolved using the ALSI40 dataset is presented as a critical difference graph in Figure 5.9. The critical difference graph shows no significant difference between the $Sample_{out}$ results for the various simulations. This is consistent with the p-value results that show only one significant difference between the $Sample_{out}$ results obtained by the trading rules evolved using a mutation probability of 60% and the results of the rules evolved with a mutation probability of 0%.

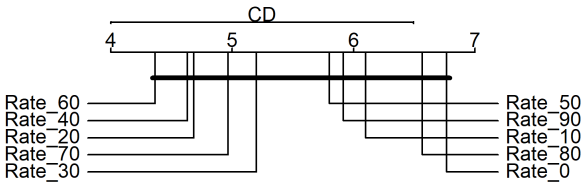



**Figure 5.9:** Critical difference plot of ALSI $Sample_{out}$ results using various mutation probabilities.

A mutation probability is required, and while the $Sample_{out}$ results are similar, the graph presented in Figure 5.10 shows that mutation probability does impact population diversity. The graph plots the mean deviation ($\sigma$) of the $Sample_{in}$ profit returned by all the individuals within the population over each generation per configured mutation probability. The graph shows that $\sigma$ is large for all simulations at generation zero, and the lower the mutation probability the faster the $\sigma$ drops to zero as the populations evolve.

The average $Sample_{in}$ profit returned by the trading rules evolved using different mutation probabilities is presented in Figure 5.11. The figure shows that simulations configured with a mutation probability larger than 20% evolved trading rules that returned larger profits than trading rules evolved with lower mutation probabilities.

The scatter plot in Figure 5.14 shows that the impact of the mutation probability on evolving profitable solutions by plotting the average generation profit per simulation as a dot and the average best $Sample_{in}$, $Sample_{sel}$, and $Sample_{out}$ profit as lines.

The scatter plots in Figure 5.14 shows that the trading rules evolved with a mutation probability between 10% and 40% reached a $Sample_{in}$ trading profit of around 5000 at generation 15. Figure 5.14 shows that the trading rules evolved using a mutation probability greater than 40% reached a $Sample_{in}$ profit of 5000 much later at around 25 to 30 generations. The only simulations to reach a $Sample_{in}$ profit greater than 6000 within 50 generations were those trading rules that were evolved with a mutation probability of 10% and 50%. A mutation probability of 0% resulted in profit stagnation, where none of the generations shows any increase in profit. A similar result was seen when mutation probability was 90%.

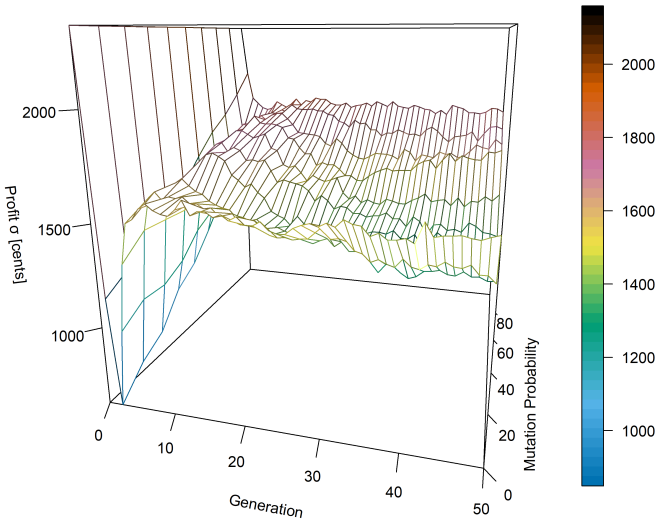



**Figure 5.10:** 3D Perspective plot of the mean deviation of ALSI $Sample_{in}$ profit results using various mutation probabilities.

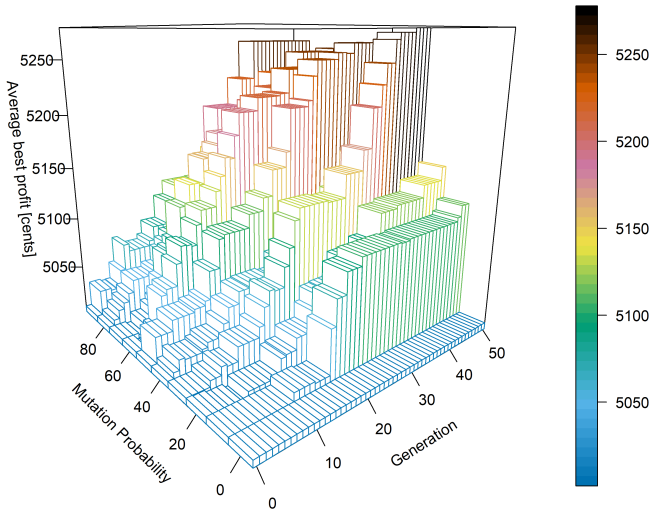



**Figure 5.11:** 3D Perspective plot of the mean ALSI $Sample_{in}$ profit results using various mutation probabilities.

The observed profit stagnation is expected, because, when the mutation probability is set at 0%, no new individuals are introduced. Selection pressure results in premature convergence, due to no exploration, resulting in profit stagnation. Premature convergence also results in fewer nodes per tree as observed in Figure 5.13. When the mutation probability is 90%, each generation undergoes mutation with little to no crossover. Each generation is compared to the global best solution, and because more mutation becomes more of a random search, it becomes less likely that a better global best solution will be found, resulting in the observed global best profit stagnation. The continuous reintroduction of new individuals through exploration reduces the effect that crossover had on evolution and manifests as fewer nodes within a tree as observed in Figure 5.13. Except for the results returned by the trading rules evolved from the LON dataset, a similar pattern was found across all the share code samples examined in this section.

Figure 5.12 compares the average best $Sample_{out}$ profit returned by each simulation per generation. The figure shows that a mutation probability of 40% resulted in the highest $Sample_{out}$ profit, followed by 60% and 30%. Analysis of the ALSI40 results shows that a mutation probability between 30% and 40% resulted in the best combination of mutation and crossover, driving the population towards the best solution across all the dataset samples.

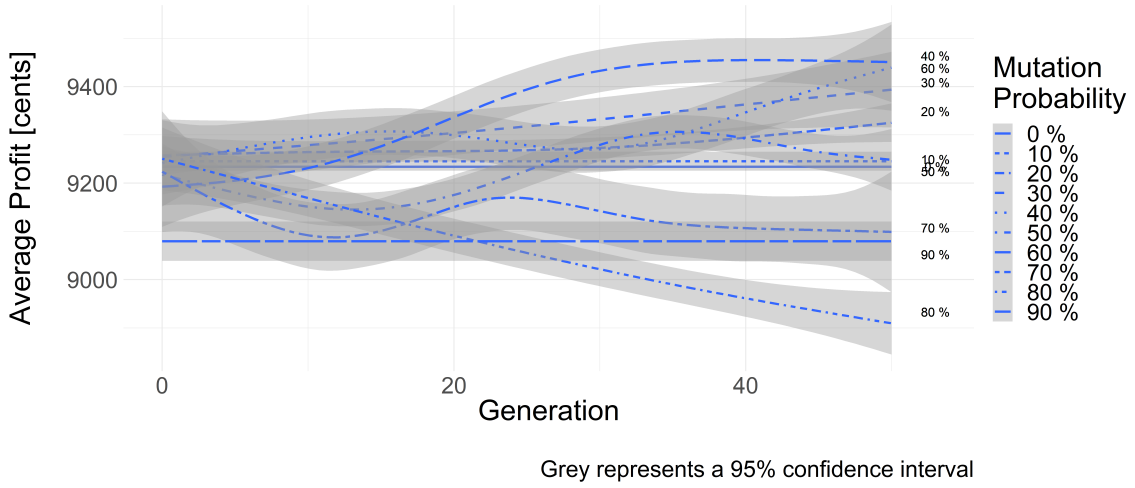



**Figure 5.12:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the ALSI40 dataset and various mutation probabilities.

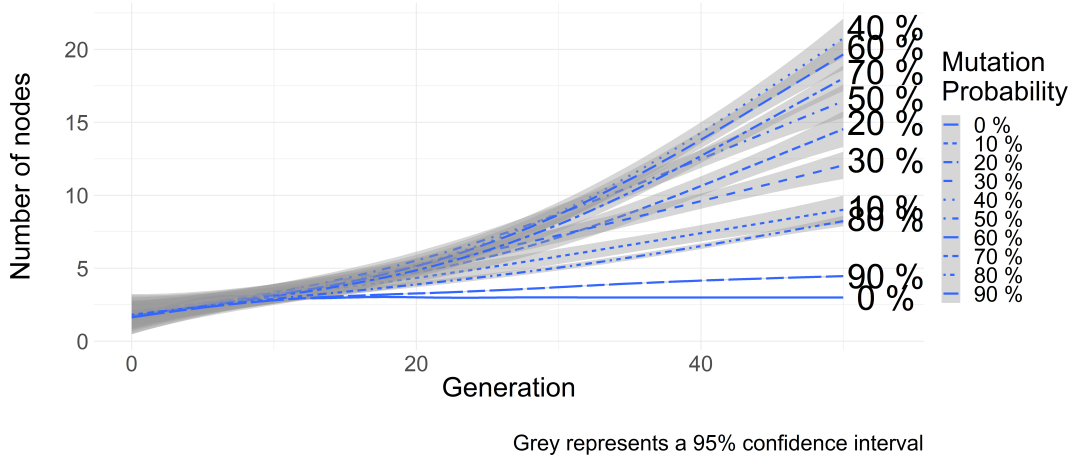
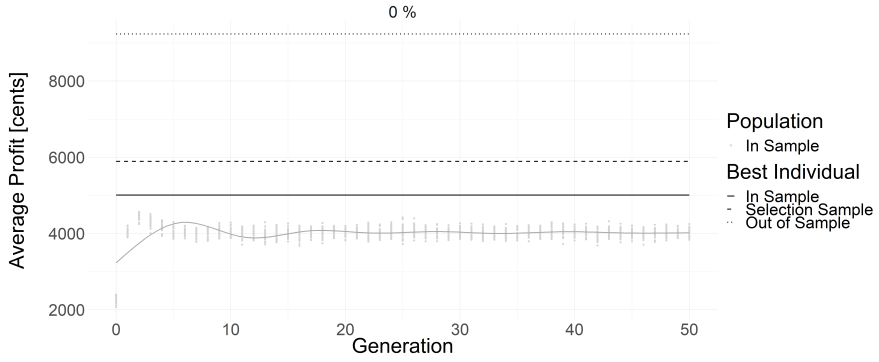
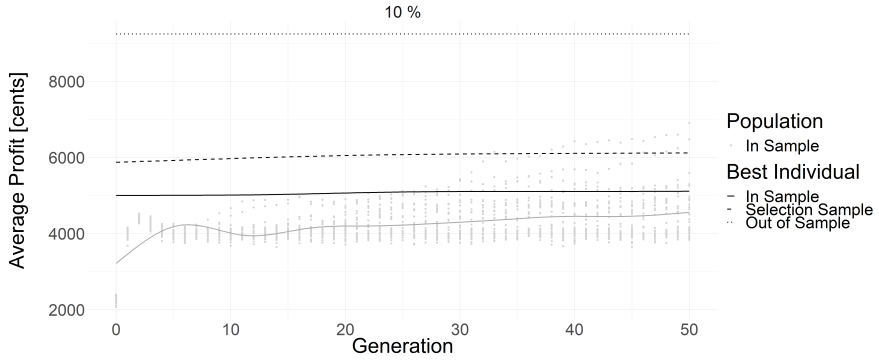



**Figure 5.13:** Average node count of an individual evolved using the ALSI40 dataset and various mutation probabilities.

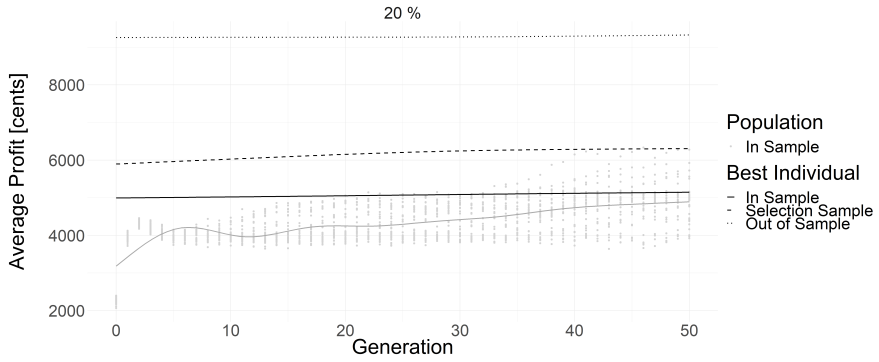



**(a)** Mutation probability set at 0%

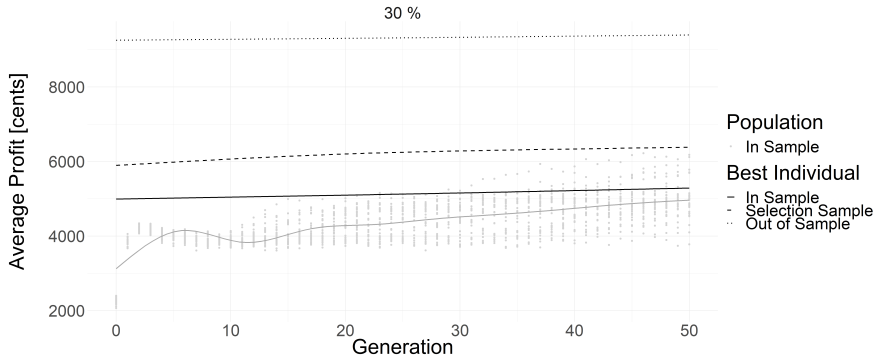



**(b)** Mutation probability set at 10%

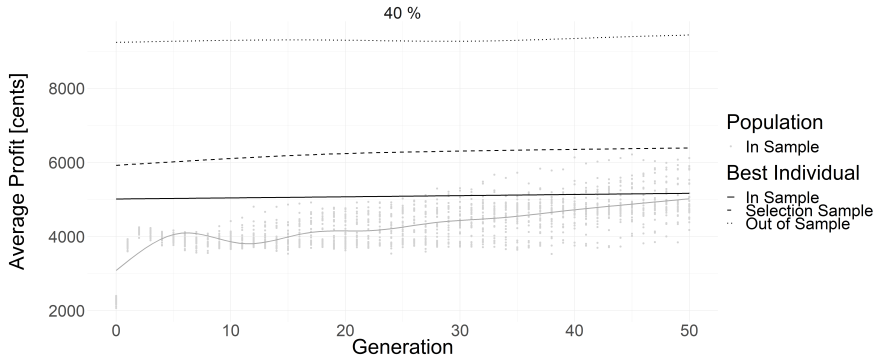
**Figure 5.14:** Scatter plot plotting returned profit using ALSI40 dataset and various mutation probabilities.



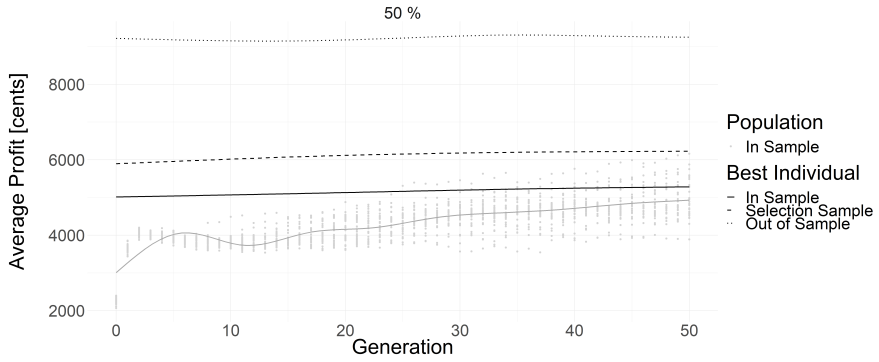

**(c)** Mutation probability set at 20%




**(d)** Mutation probability set at 30%

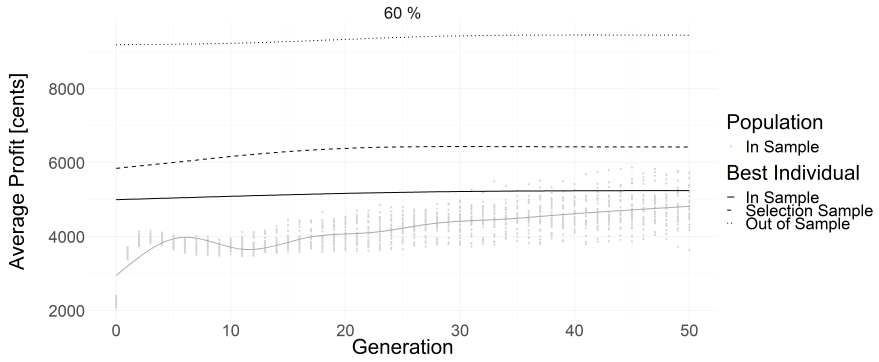



**(e)** Mutation probability set at 40%

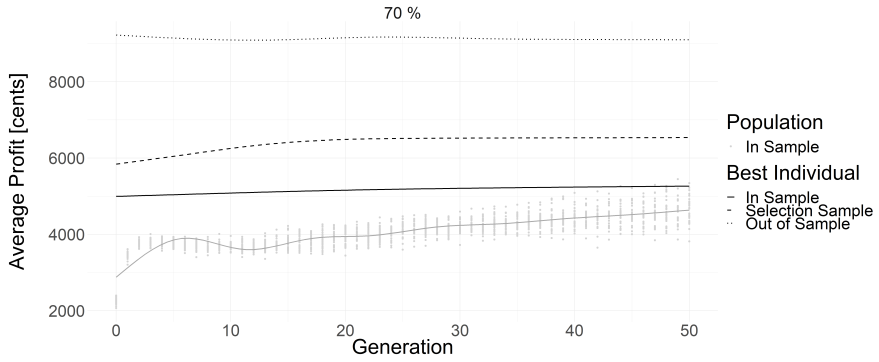



**(f)** Mutation probability set at 50%

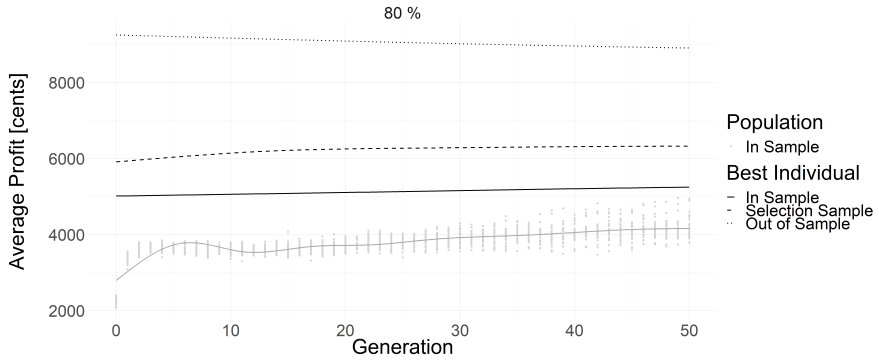
**Figure 5.14:** Scatter plot plotting returned profit using ALSI40 dataset and various mutation probabilities (Cont.).



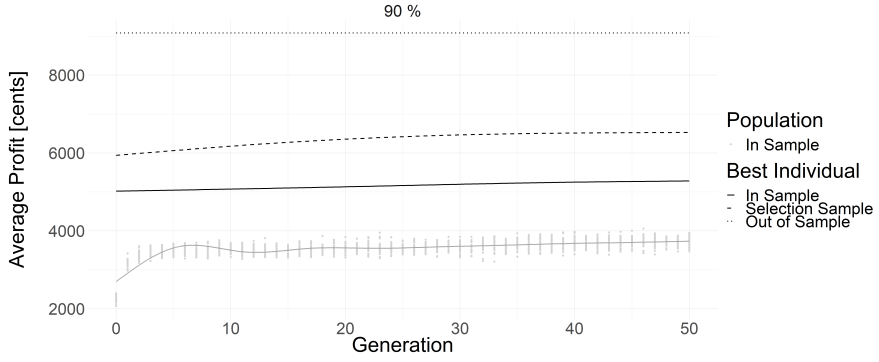

**(g)** Mutation probability set at 60%




**(h)** Mutation probability set at 70%




**(i)** Mutation probability set at 80%




**(j)** Mutation probability set at 90%

**Figure 5.14:** Scatter plot plotting returned profit using the ALSI40 dataset and various mutation probabilities (Cont.).

The $Sample_{in}$ profit results returned by the trading rules evolved using the BIL dataset are presented in Figure 5.19 and Figure 5.17. The figures show that a mutation probability of between 40% and 50% evolved trading rules that reached the greatest profit in the shortest number of generations. The second best performing trading rules were evolved with a mutation probability of 30% and 60%. However, the $Sample_{out}$ profit presented in Figure 5.19 and Figure 5.17 shows that the average $Sample_{out}$ profit decreased with increase in the mutation probability. The best $Sample_{out}$ profit was obtained by the trading rules evolved with a mutation probability of 0%, 10%, and 30%. No mutation occurred at a 0% probability, and the critical difference plot in Figure 5.15 shows no critical difference between the results obtained by the trading rules evolved using a mutation probability of 0% or 10%. Based on the BIL results, a mutation probability of 30% is preferred.

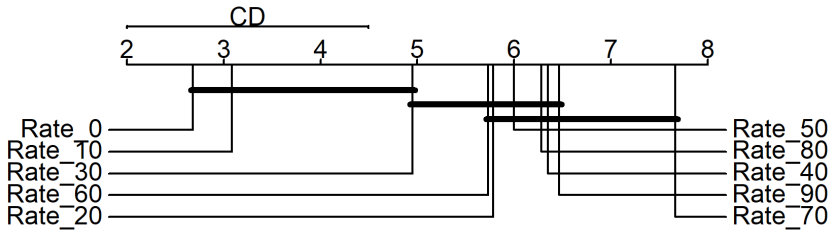



**Figure 5.15:** Critical difference plot of BIL $Sample_{out}$ results using various mutation probabilities.

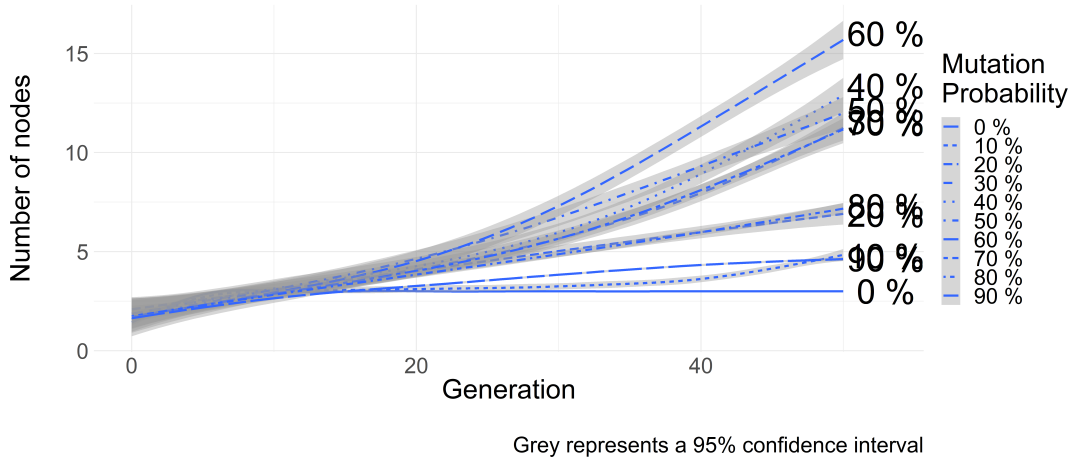



**Figure 5.16:** Average node count of an individual evolved using the BIL dataset and various mutation probabilities.

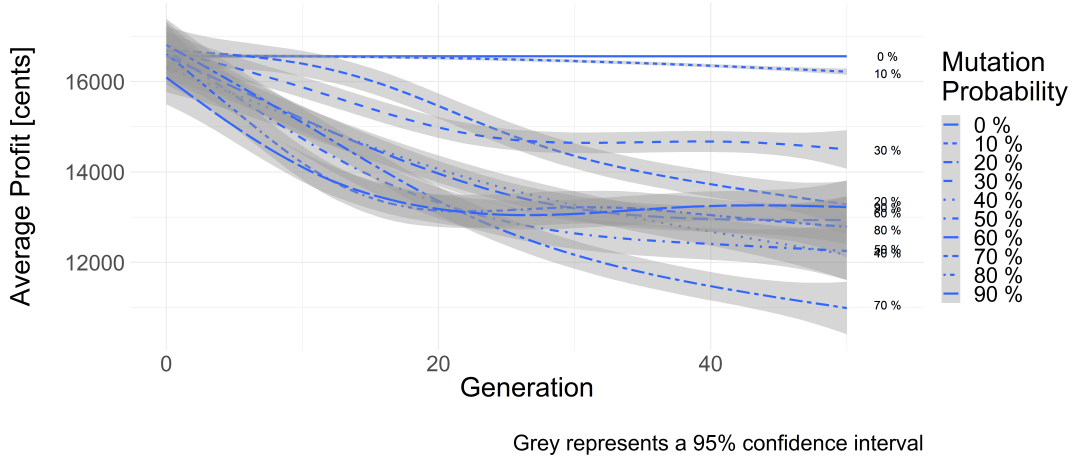



**Figure 5.17:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the BIL dataset and various mutation probabilities.

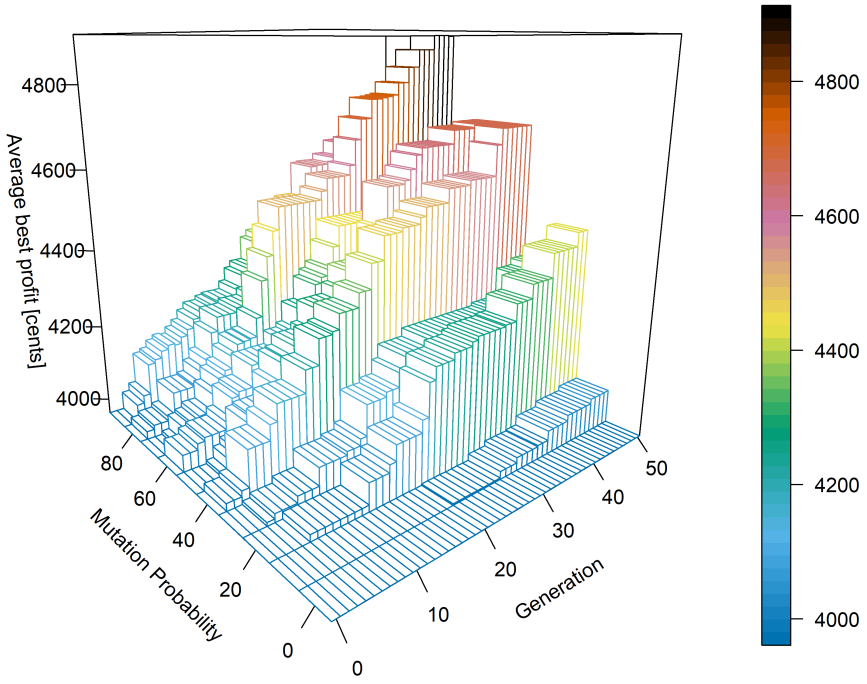
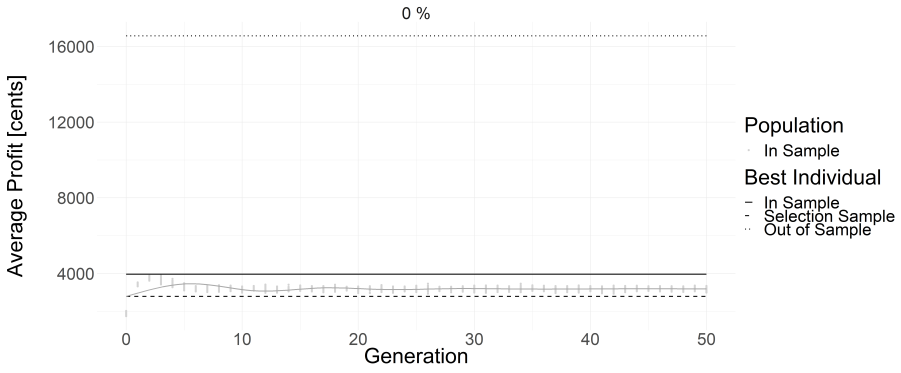
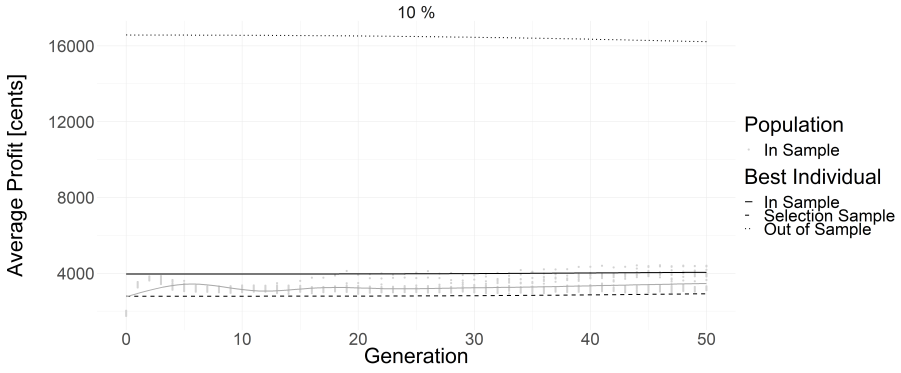


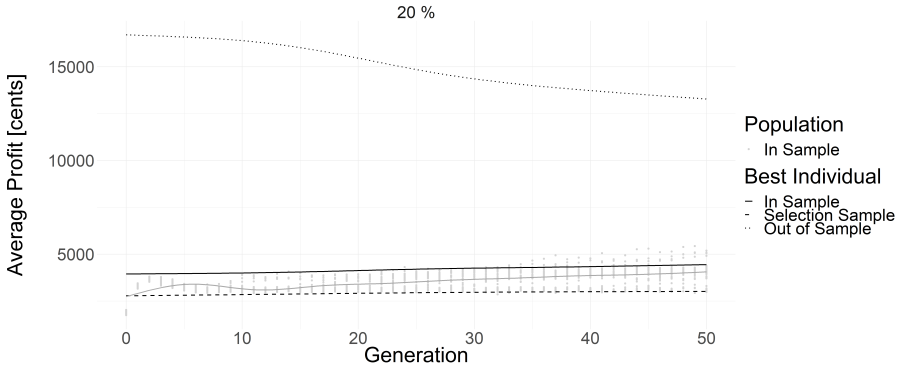

**Figure 5.18:** 3D Perspective plot of the mean BIL $Sample_{in}$ profit results using various mutation probabilities.



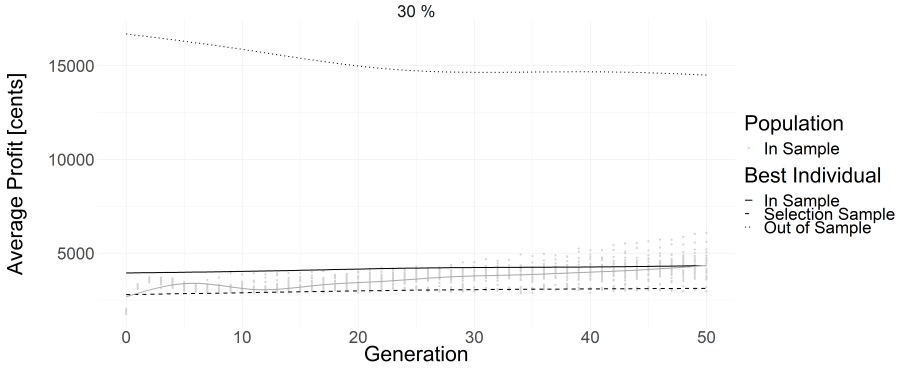

**(a)** Mutation probability set at 0%




**(b)** Mutation probability set at 10%

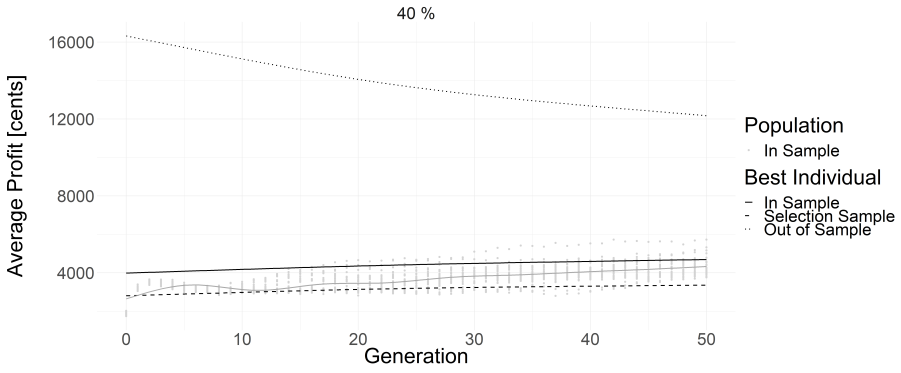



**(c)** Mutation probability set at 20%

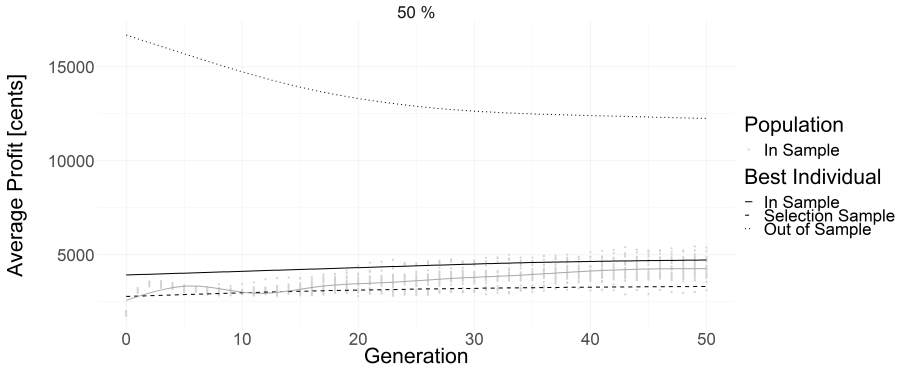



**(d)** Mutation probability set at 30%

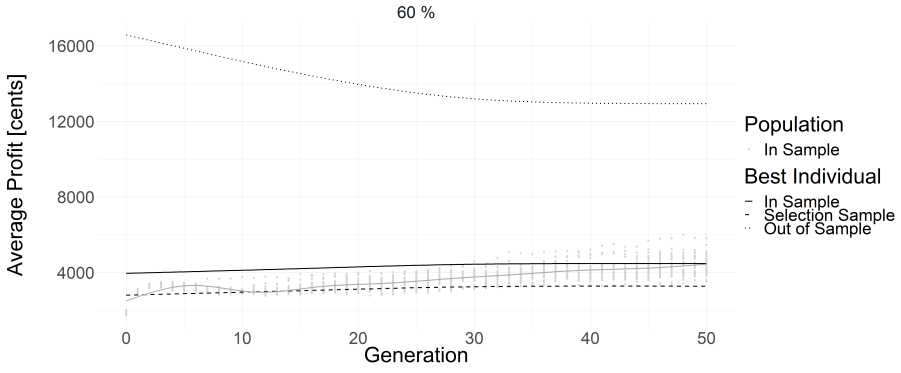
**Figure 5.19:** Scatter plot plotting returned profit using BIL dataset and various mutation probabilities.



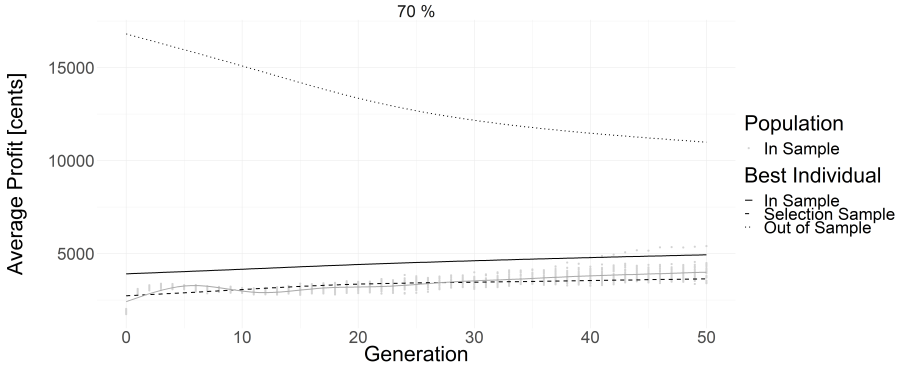

**(e)** Mutation probability set at 40%




**(f)** Mutation probability set at 50%

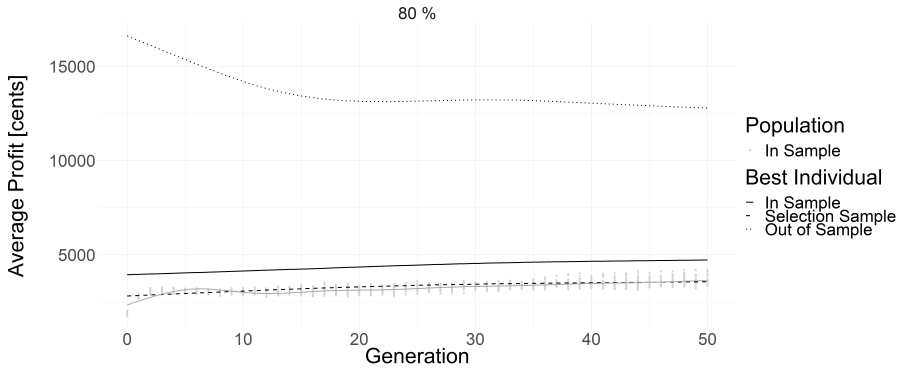



**(g)** Mutation probability set at 60%

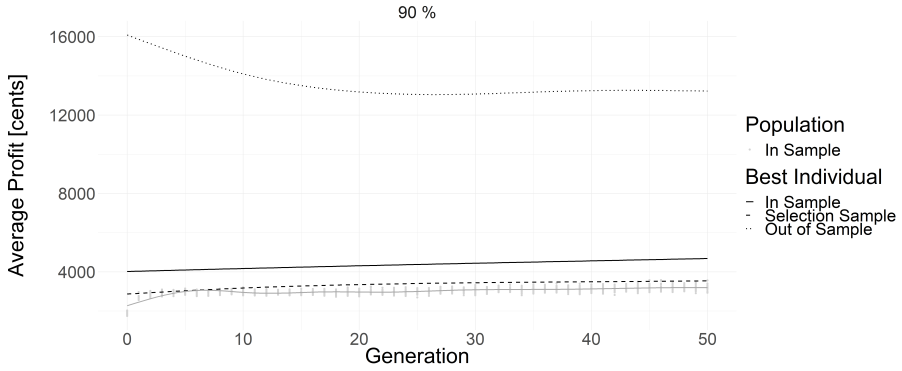



**(h)** Mutation probability set at 70%

**Figure 5.19:** Scatter plot plotting returned profit using BIL dataset and various mutation probabilities (Cont.).




**(i)** Mutation probability set at 80%




**(j)** Mutation probability set at 90%

**Figure 5.19:** Scatter plot plotting returned profit using BIL dataset and various mutation probabilities (Cont.).

Figure 5.24 and Figure 5.23 show that only those trading rules evolved with a mutation probability between 30% and 90% reached an average best $Sample_{in}$ profit greater than 35000. Further examination of the graphs shows that the average $Sample_{in}$ population profit was much lower than the average best individual's $Sample_{in}$ profit when the trading rules were evolved using a mutation probability between 60% and 90%. The difference between the average best individual $Sample_{in}$ profit and the average population $Sample_{in}$ profit observed in Figure 5.24 continues in the $Sample_{out}$ results as presented by the high $\sigma$ value recorded in Table 5.21. Therefore, the trading rules evolved with a mutation between 30% and 50% produced more consistent results than the trading rules evolved with a mutation probability between 60% and 90%.

Figure 5.21 shows that the trading rules evolved with a mutation probability of 40%, 70%, and 50% returned the largest average $Sample_{out}$ profit. The critical difference plot in Figure 5.20 shows no critical difference between the $Sample_{out}$ results obtained by the trading rules evolved with a mutation

probability between 30% and 50%. The p-values presented previously in Figure 5.8c found no significant difference between the $Sample_{out}$ results obtained by the trading rules evolved with a mutation probability between 30% and 50%. Based on the results presented for the INVNED dataset, a mutation probability between 30% and 50% is preferred.

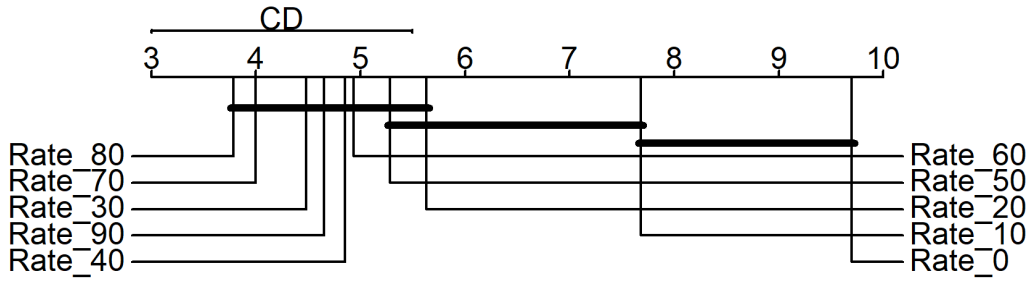



**Figure 5.20:** Critical difference plot of INVNED $Sample_{out}$ results using various mutation probabilities.

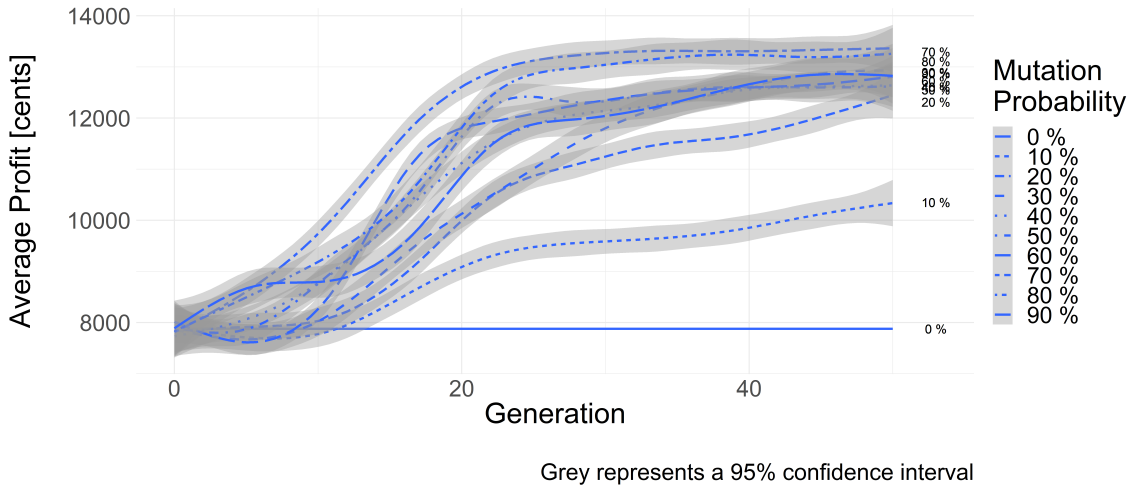



**Figure 5.21:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the INVNED dataset and various mutation probabilities.

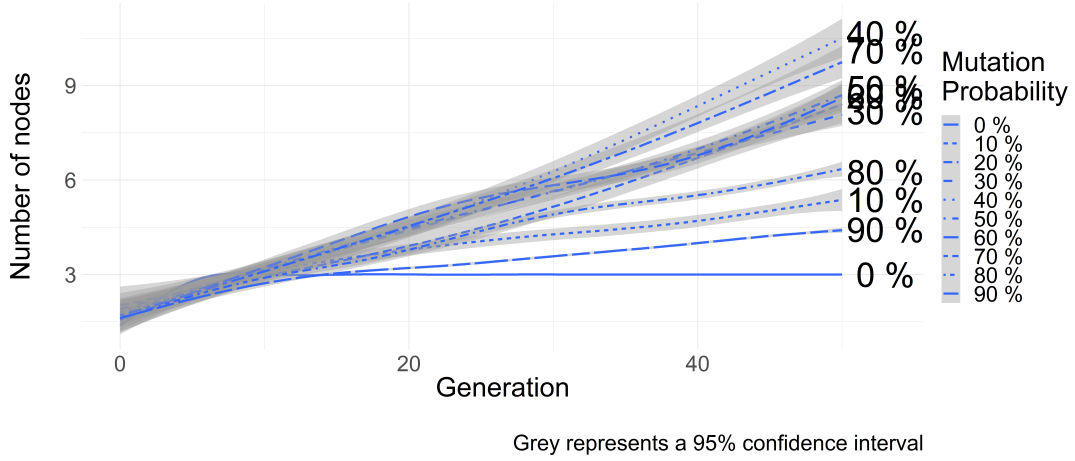



**Figure 5.22:** Average node count of an individual evolved using the INVNED dataset and various mutation probabilities.

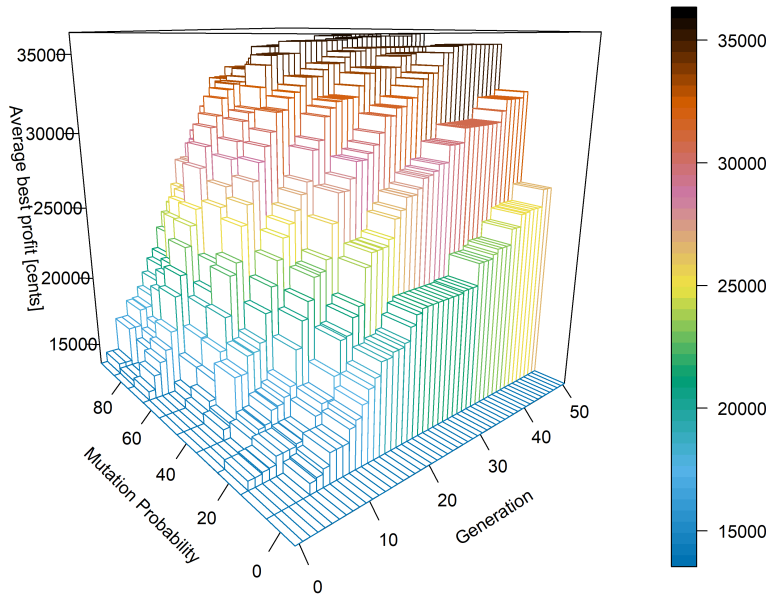
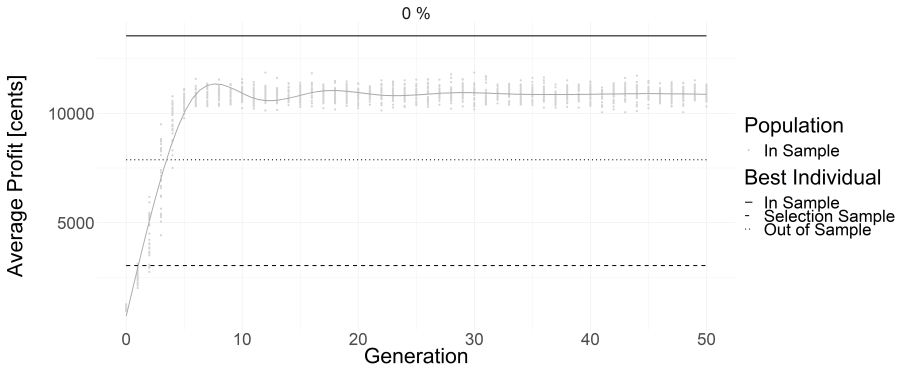
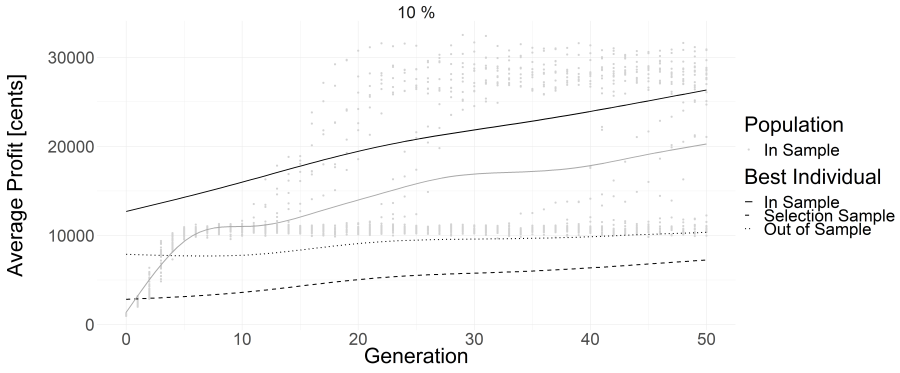


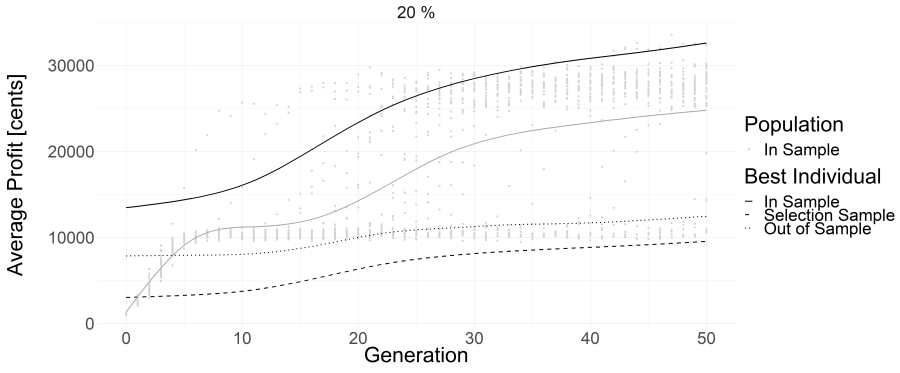

**Figure 5.23:** 3D Perspective plot of the mean INVNED $Sample_{in}$ profit results using various mutation probabilities.



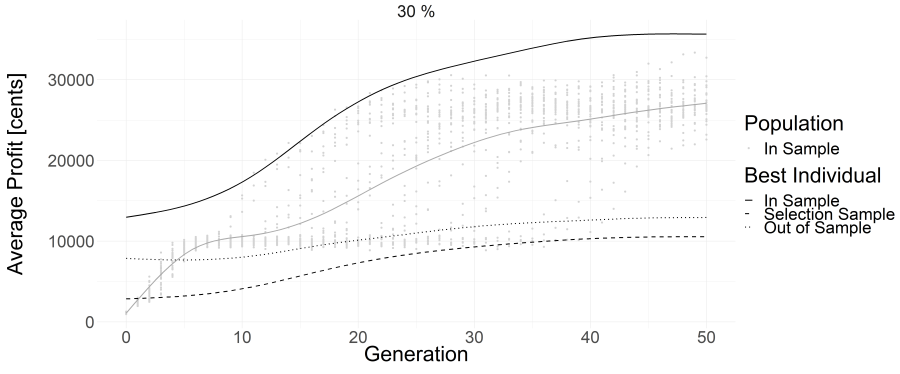

**(a)** Mutation probability set at 0%




**(b)** Mutation probability set at 10%

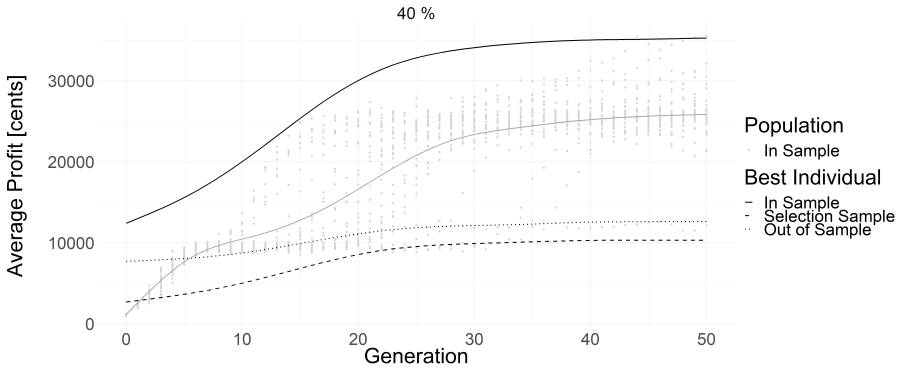



**(c)** Mutation probability set at 20%

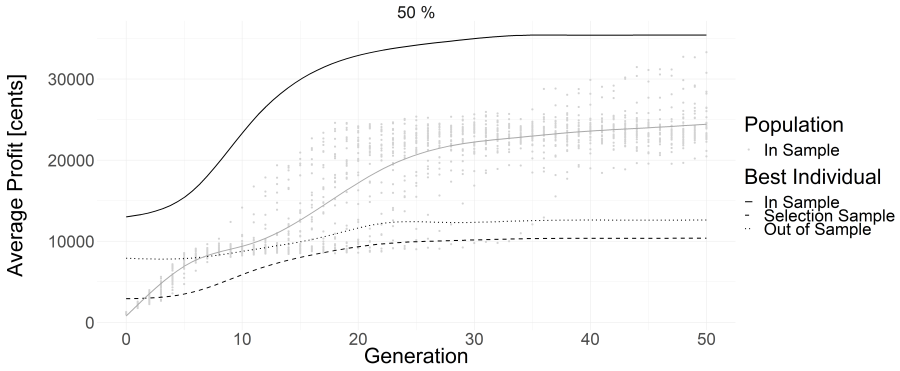



**(d)** Mutation probability set at 30%

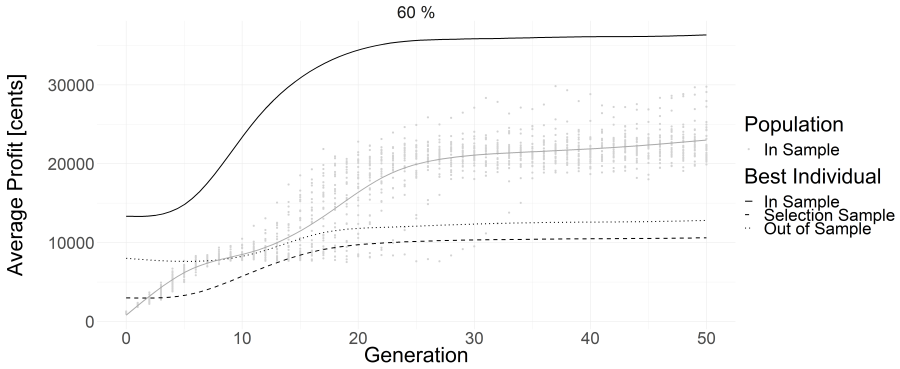
**Figure 5.24:** Scatter plot plotting returned profit using INVNED dataset and various mutation probabilities.



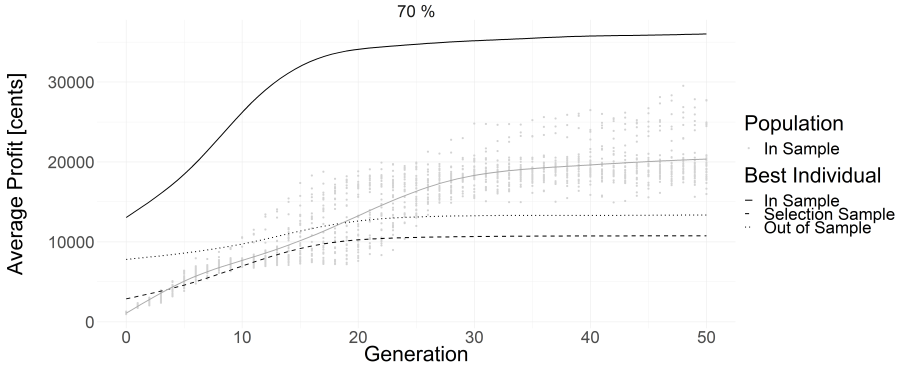

**(e)** Mutation probability set at 40%




**(f)** Mutation probability set at 50%

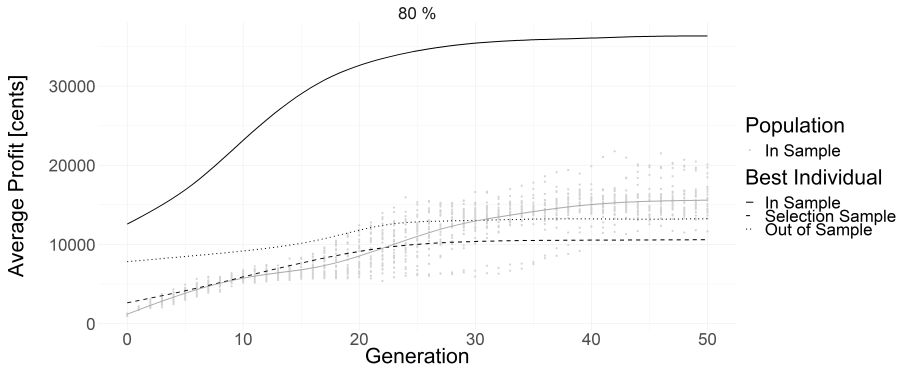



**(g)** Mutation probability set at 60%

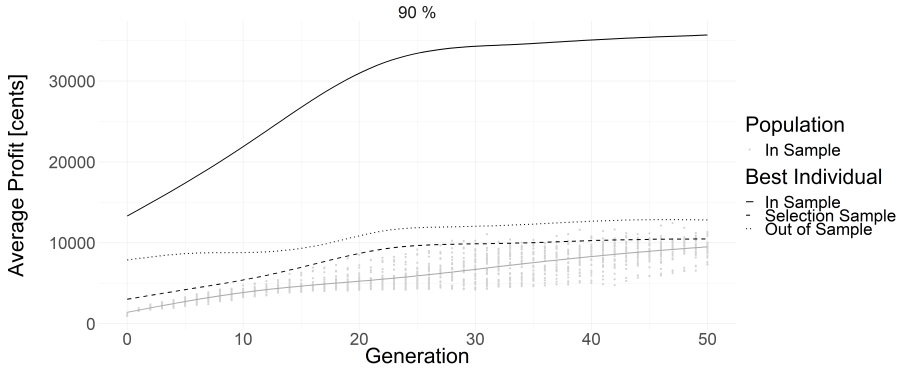



**(h)** Mutation probability set at 70%

**Figure 5.24:** Scatter plot plotting returned profit using INVNED dataset and various mutation probabilities (Cont.).




**(i)** Mutation probability set at 80%




**(j)** Mutation probability set at 90%

**Figure 5.24:** Scatter plot plotting returned profit using INVNED dataset and various mutation probabilities (Cont.).

The results of the trading rules evolved using the INVREM dataset are presented in Figure 5.27. The trading rules evolved for the INVREM dataset performed similar to the rules evolved for the INVNED dataset. Figure 5.27 shows that only those trading rules evolved with a mutation probability between 30% and 90% reached an average best $Sample_{in}$ profit greater than 30000. Like INVNED, the trading rules evolved using a mutation probability between 30% and 50% produced more consistent $Sample_{in}$ results than the trading rules evolved using a mutation probability between 60% and 90%.

Figure 5.28 shows that the best $Sample_{out}$ performing trading rules were evolved using a mutation probability of 20% followed by 30%, 40%, 50% and 60%. Although the average best individual trading rules evolved with a mutation probability of 20% did not exceed a $Sample_{in}$ profit of 30000, the average $Sample_{in}$ population profit did. Furthermore, a mutation probability of 20% resulted in a smaller $Sample_{out}$ $\sigma$ as presented in Table 5.21. Based on the observed results of the trading rules evolved for

the INVREM dataset, a mutation probability between 20% and 50% is preferred.

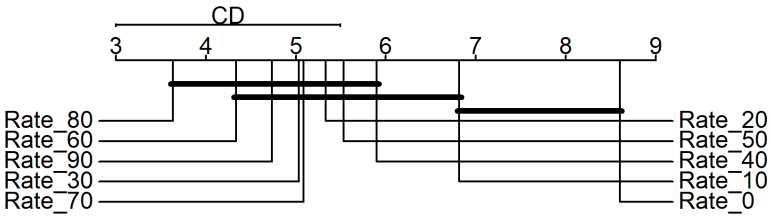



**Figure 5.25:** Critical difference plot of INVREM $Sample_{out}$ results using various mutation probabilities.

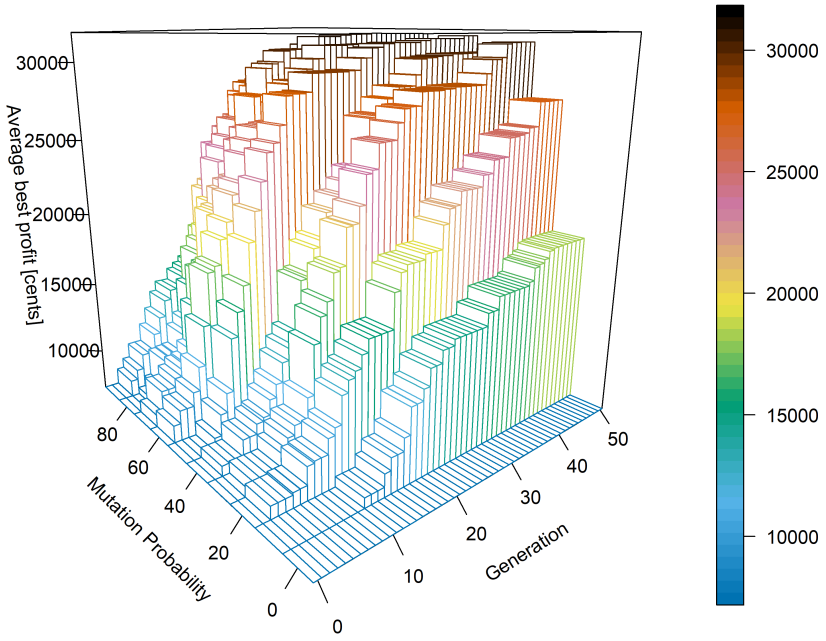
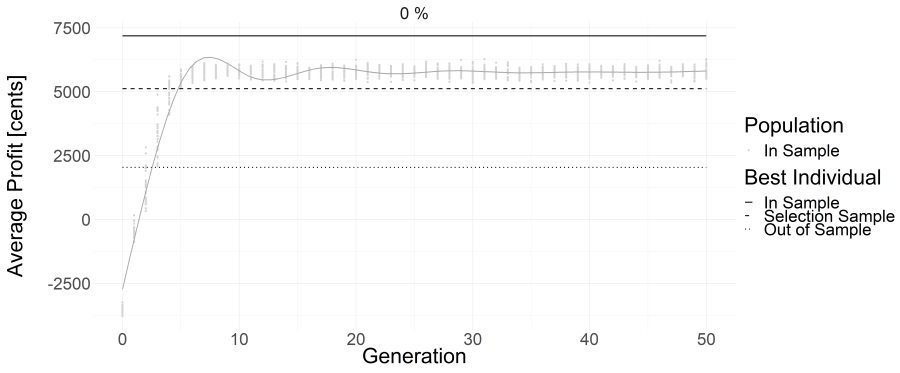
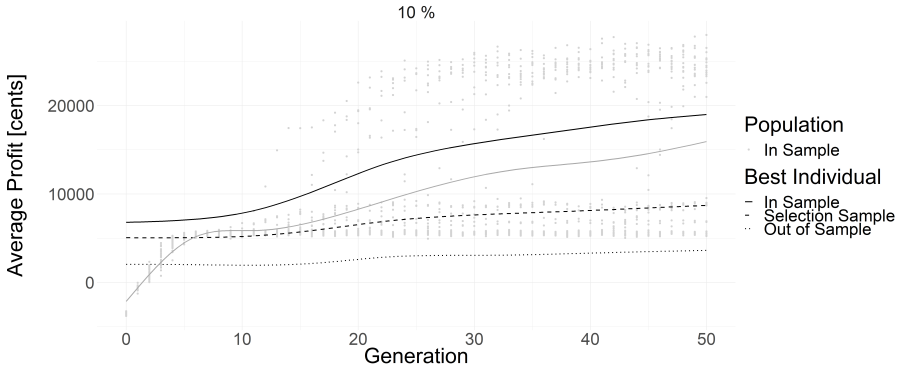


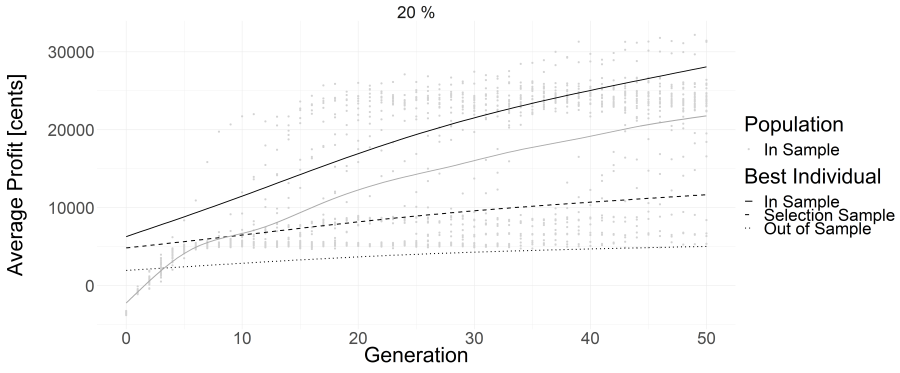

**Figure 5.26:** 3D Perspective plot of the mean INVREM $Sample_{in}$ profit results using various mutation probabilities.



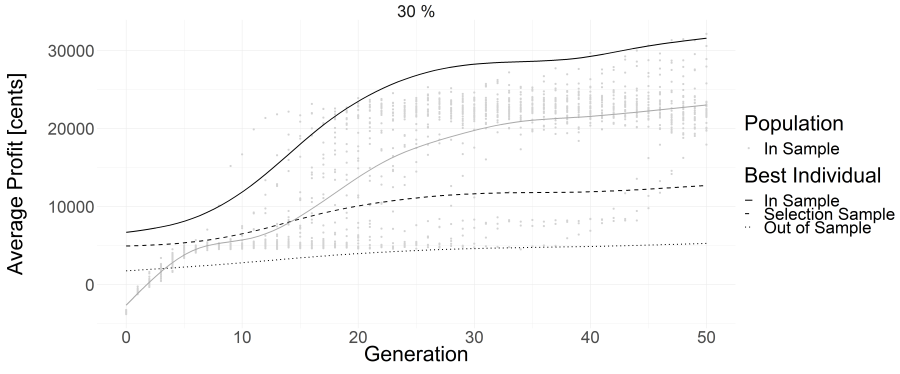

**(a)** Mutation probability set at 0%




**(b)** Mutation probability set at 10%

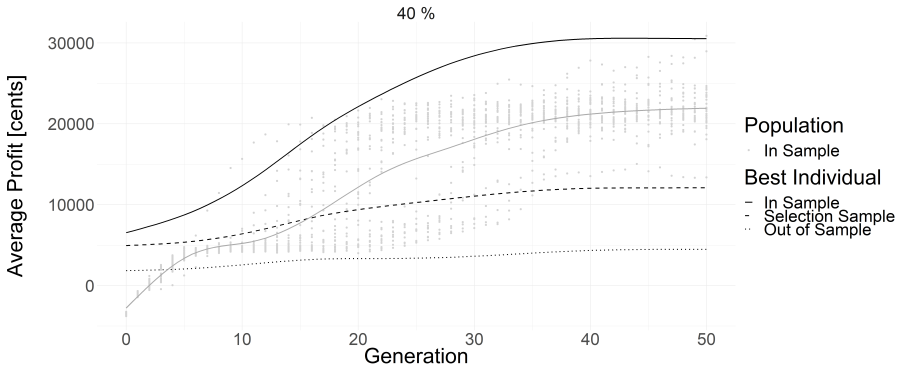



**(c)** Mutation probability set at 20%

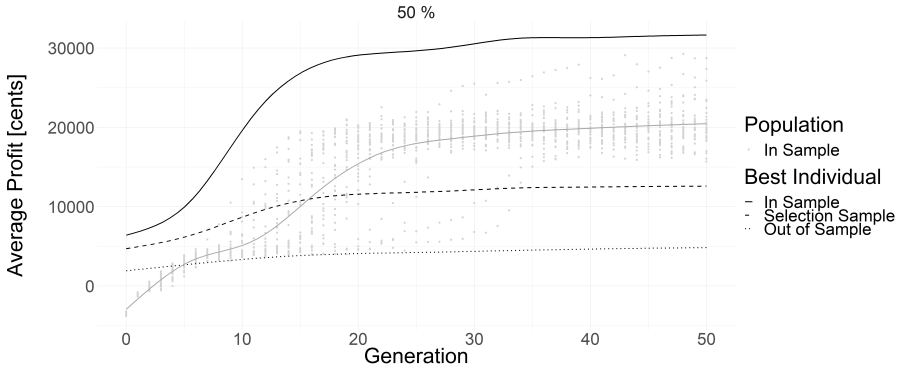



**(d)** Mutation probability set at 30%

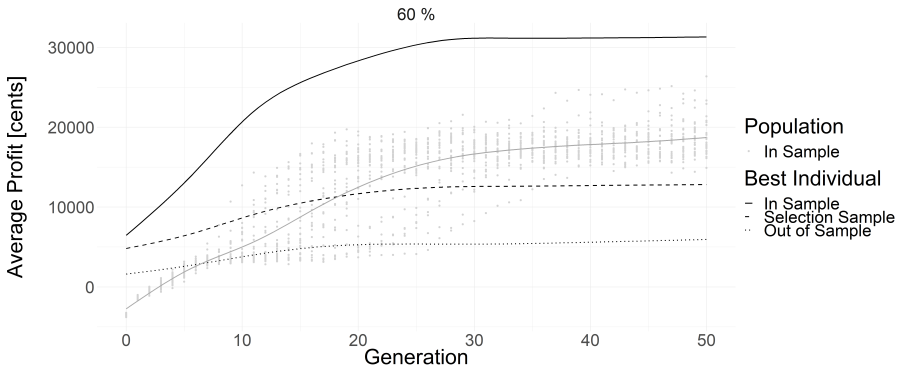
**Figure 5.27:** Scatter plot plotting returned profit using INVREM dataset and various mutation probabilities.



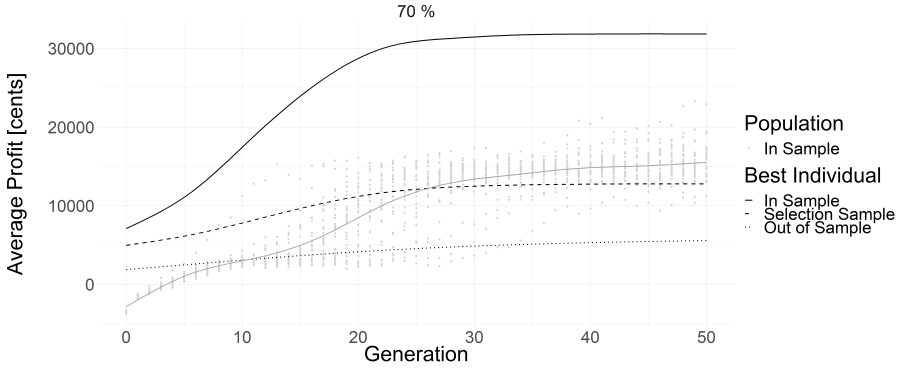

**(e)** Mutation probability set at 40%




**(f)** Mutation probability set at 50%

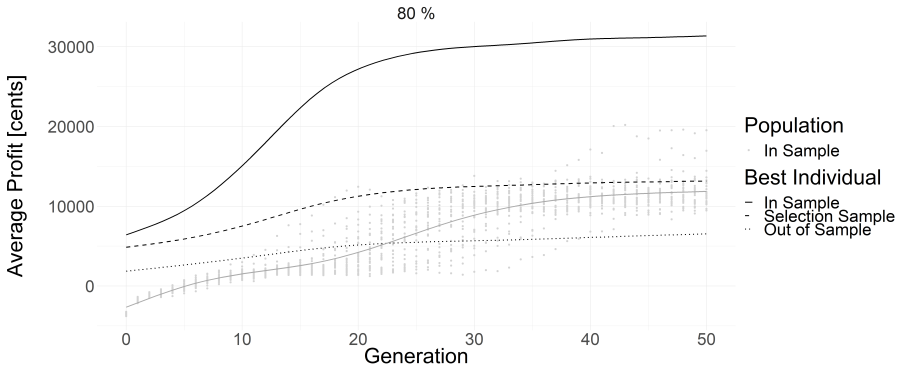



**(g)** Mutation probability set at 60%

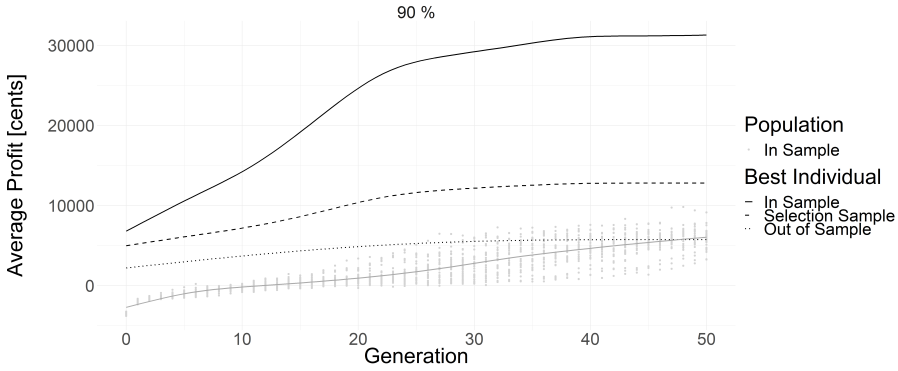



**(h)** Mutation probability set at 70%

**Figure 5.27:** Scatter plot plotting returned profit using INVREM dataset and various mutation probabilities (Cont.).




**(i)** Mutation probability set at 80%




**(j)** Mutation probability set at 90%

**Figure 5.27:** Scatter plot plotting returned profit using INVREM dataset and various mutation probabilities (Cont.).

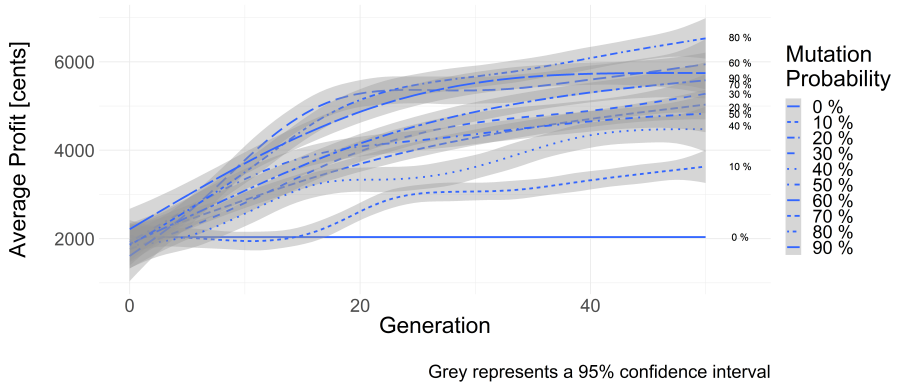



**Figure 5.28:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the INVREM dataset and various mutation probabilities.

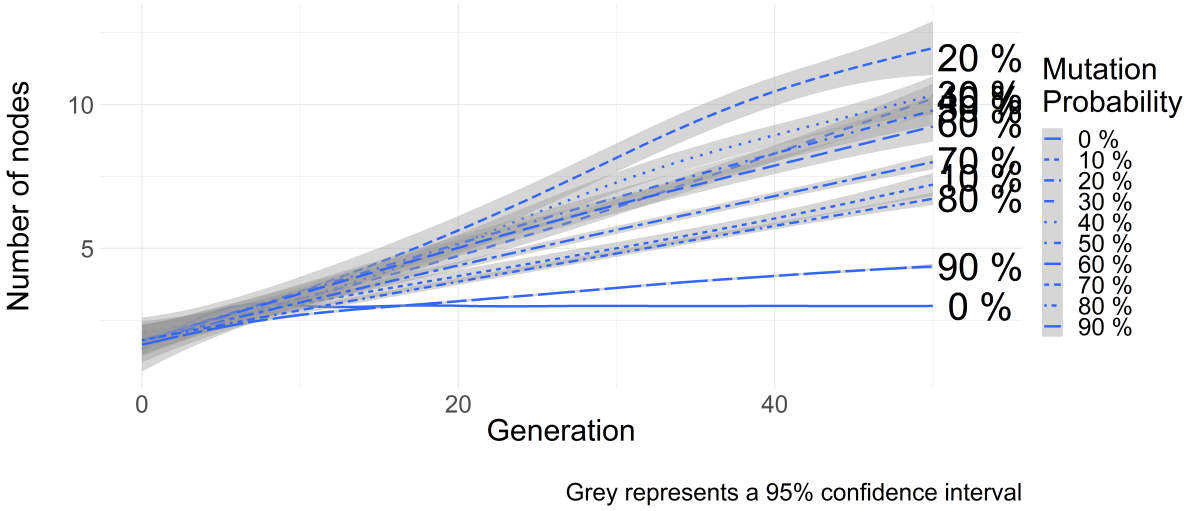



**Figure 5.29:** Average node count of an individual evolved using the INVREM dataset and various mutation probabilities.

The results of the trading rules evolved using the INVSBK dataset are presented in Figure 5.31. The figure shows that the best $Sample_{out}$ profit was returned by the trading rules evolved using a mutation probability of 60%. The critical difference graph in Figure 5.30 found no critical difference between the $Sample_{out}$ results returned by the trading rules evolved using a mutation probability between 20% and 90%. The p-values presented in Figure 5.8e found no significant difference between the results obtained by the trading rules evolved using a mutation probability between 20% and 90%. The $Sample_{in}$ results presented in Figure 5.33 and Figure 5.32 show that a mutation probability of 30% generated the best performing $Sample_{in}$ trading rules. Only the trading rules evolved with a mutation probability of 10%, 20%, and 30% produced an average population $Sample_{in}$ profit greater than 25000. A mutation probability between 30% and 90% evolved the trading rules with an average best individual $Sample_{in}$ profit greater than 25000. The difference between the average best individual $Sample_{in}$ profit and the average population $Sample_{in}$ profit is the smallest for the trading rules evolved using a mutation probability between 10% and 40%.

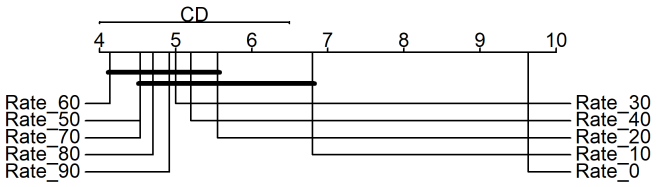



**Figure 5.30:** Critical difference plot of INVSBK $Sample_{out}$ results using various mutation probabilities.

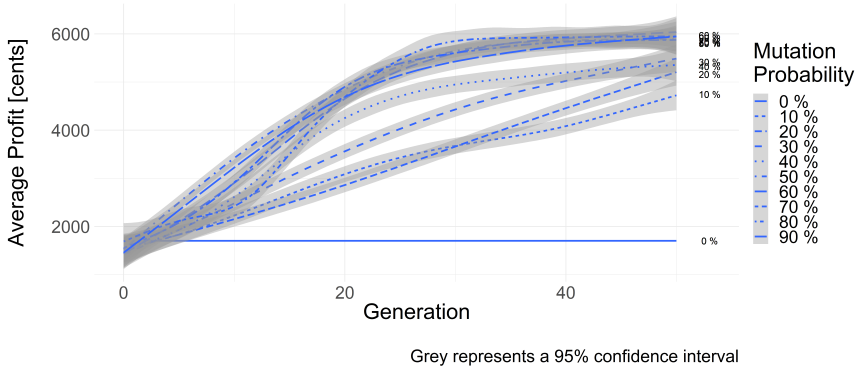



**Figure 5.31:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the INVSBK dataset and various mutation probabilities.

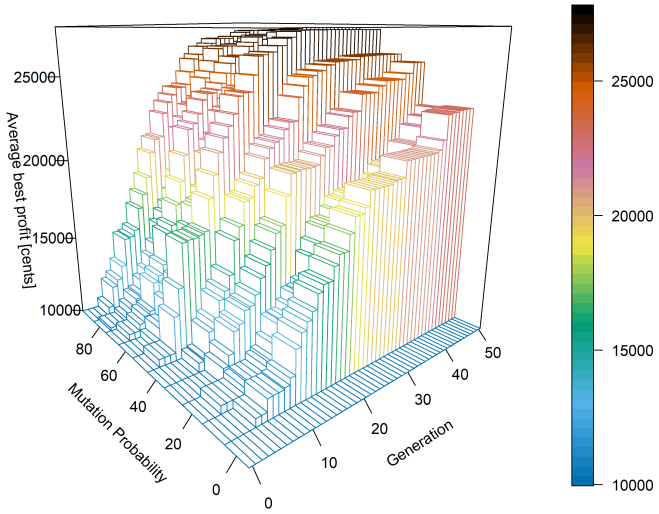
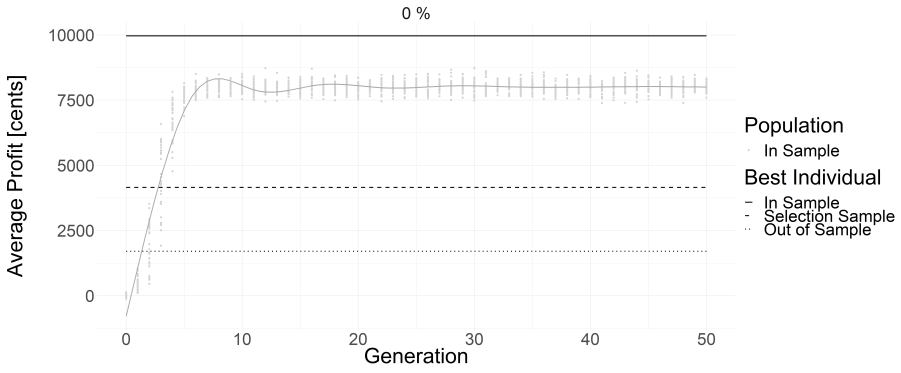
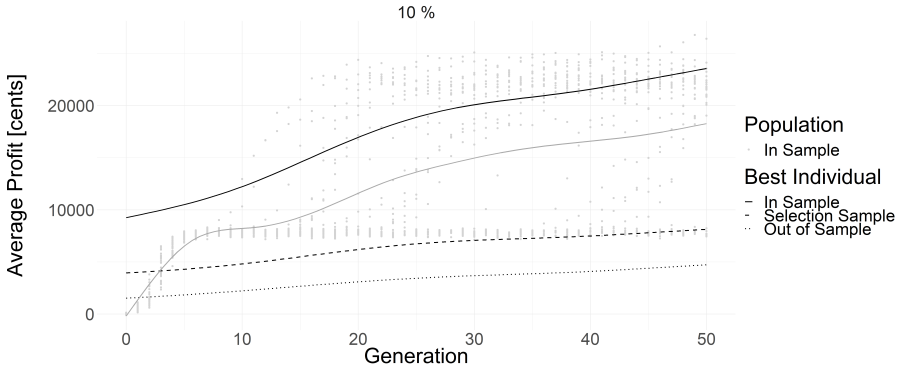


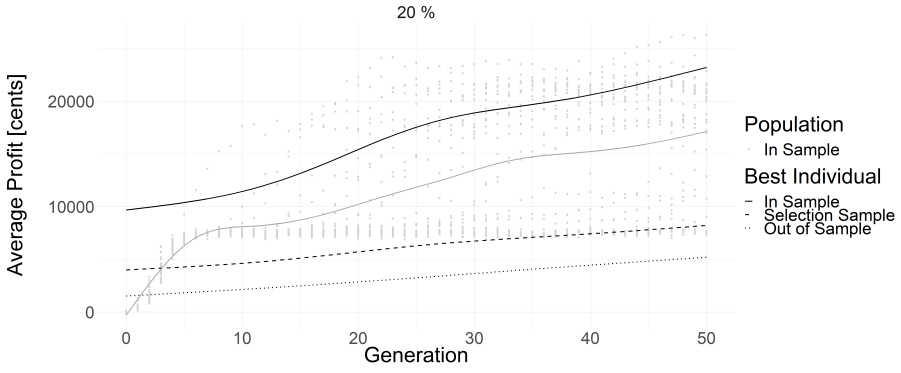

**Figure 5.32:** 3D Perspective plot of the mean INVSBK $Sample_{in}$ profit results using various mutation probabilities.



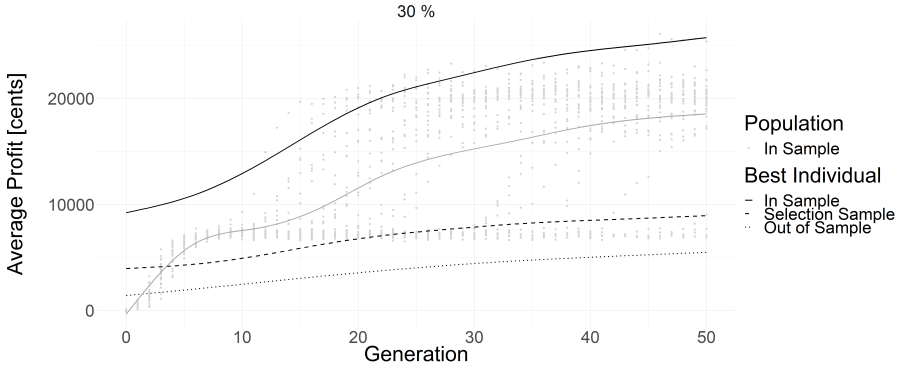

**(a)** Mutation probability set at 0%




**(b)** Mutation probability set at 10%

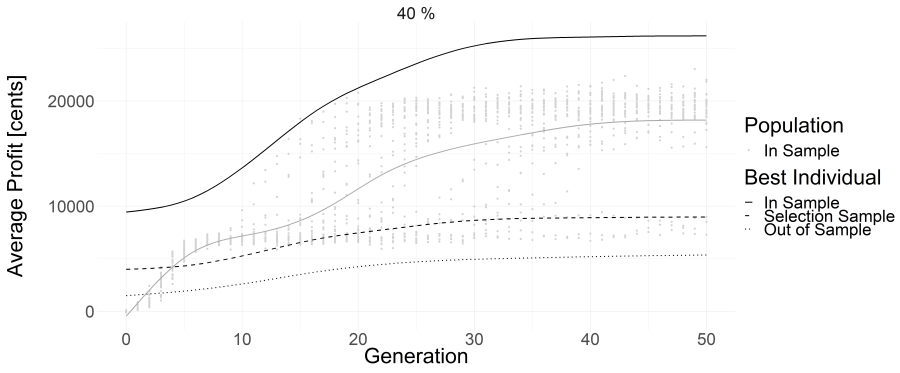



**(c)** Mutation probability set at 20%

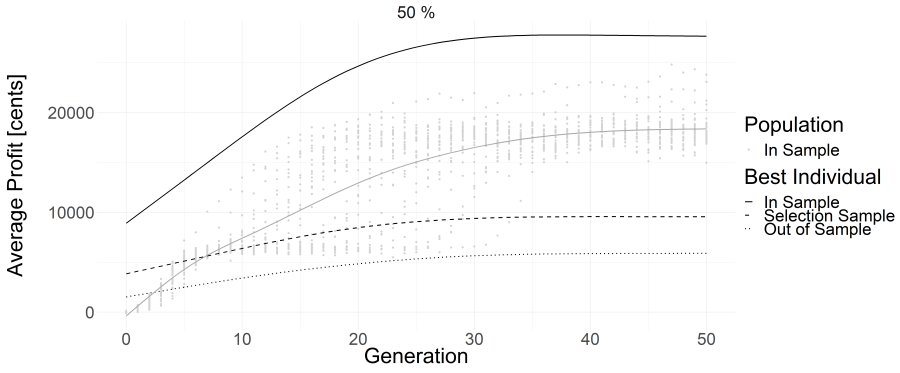



**(d)** Mutation probability set at 30%

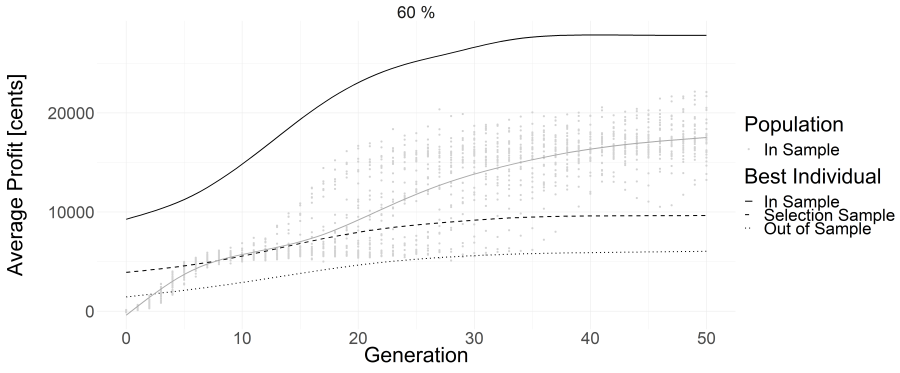
**Figure 5.33:** Scatter plot plotting returned profit using INVSBK dataset and various mutation probabilities.



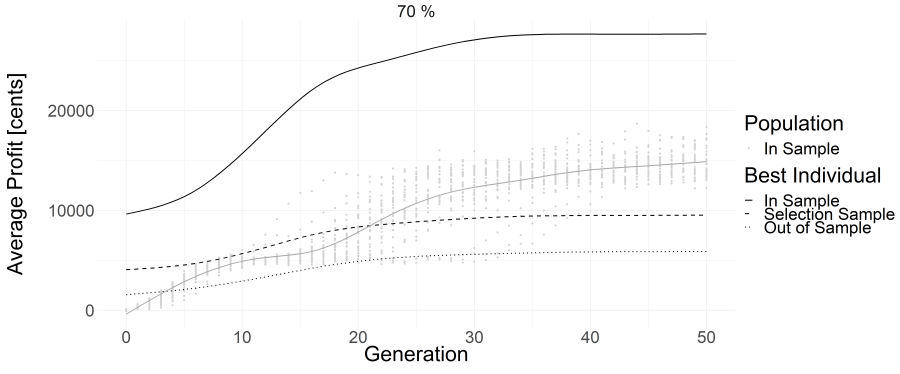

**(e)** Mutation probability set at 40%




**(f)** Mutation probability set at 50%

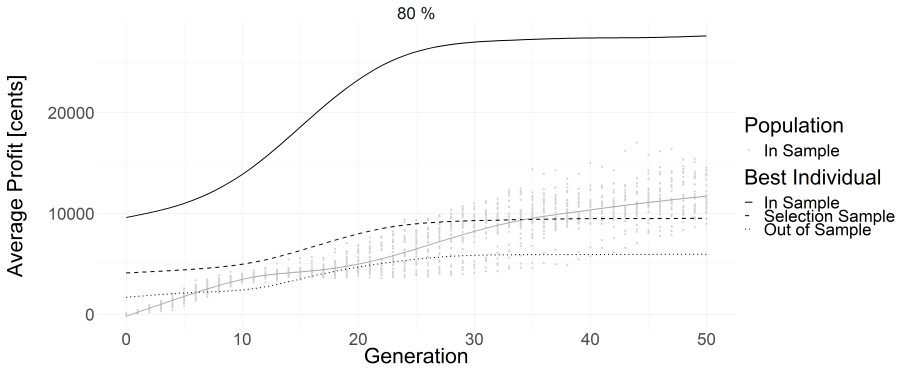



**(g)** Mutation probability set at 60%

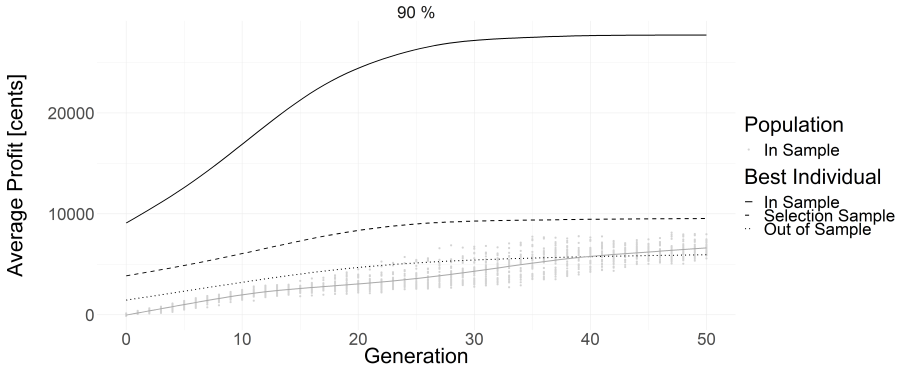



**(h)** Mutation probability set at 70%

**Figure 5.33:** Scatter plot plotting returned profit using INVSBK dataset and various mutation probabilities (Cont.).




**(i)** Mutation probability set at 80%




**(j)** Mutation probability set at 90%

**Figure 5.33:** Scatter plot plotting returned profit using INVSBK dataset and various mutation probabilities (Cont.).

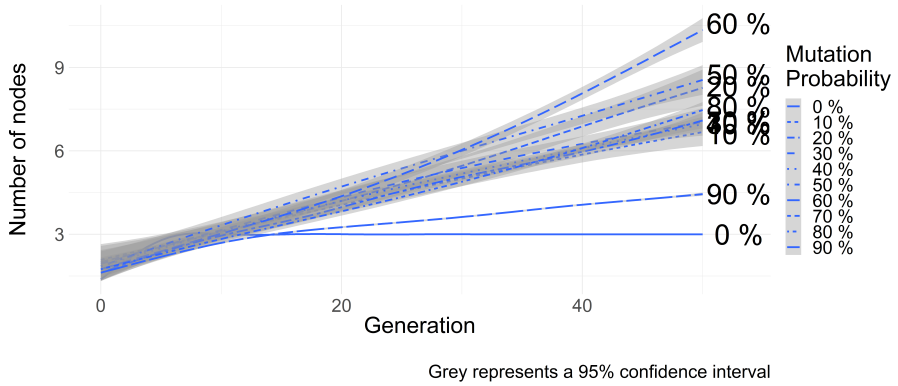



**Figure 5.34:** Average node count of an individual evolved using the INVSBK dataset and various mutation probabilities.

The best $Sample_{out}$ profit was returned by the trading rules evolved using a mutation probability of 70% as presented in Figure 5.38. The critical difference graph in Figure 5.35 found no critical difference between any of the $Sample_{out}$ results. Figure 5.37 and Figure 5.36 show that the trading rules evolved with a mutation probability of 70% is preferred, because, only the trading rules evolved using a mutation probability between 70% and 80% produced average best $Sample_{sel}$ profits greater than 10000, all trading rules produced a $Sample_{in}$ profit greater than 5000, trading rules evolved using a mutation probability of 70% produced the highest $Sample_{out}$ results. A high mutation probability of 70% is not in-line with the observations of the previous datasets. These results are in contradiction of a low mutation probability rule of thumb.

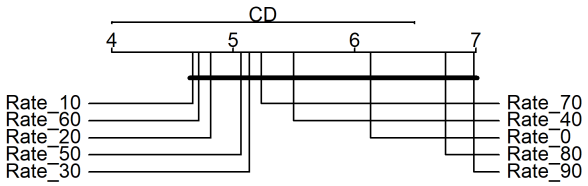



**Figure 5.35:** Critical difference plot of LON $Sample_{out}$ results using various mutation probabilities.

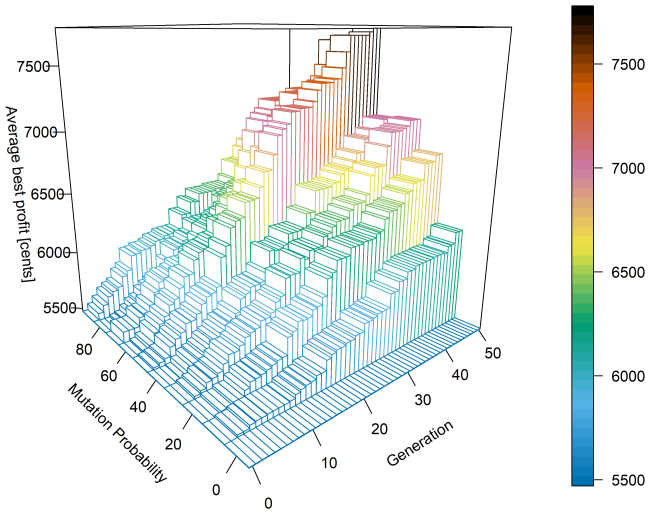
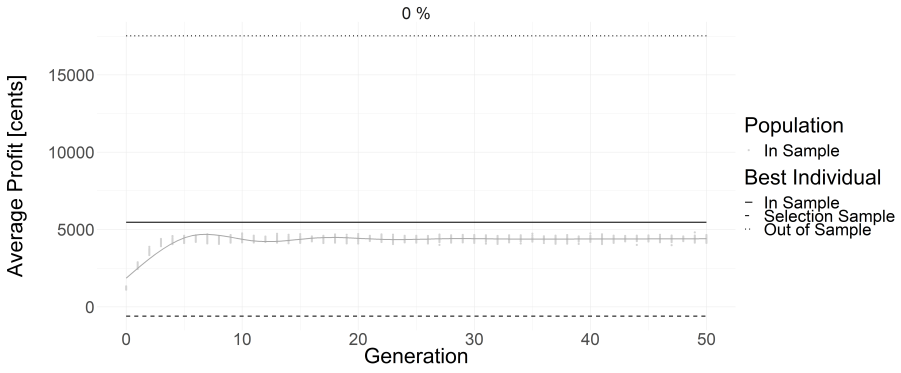
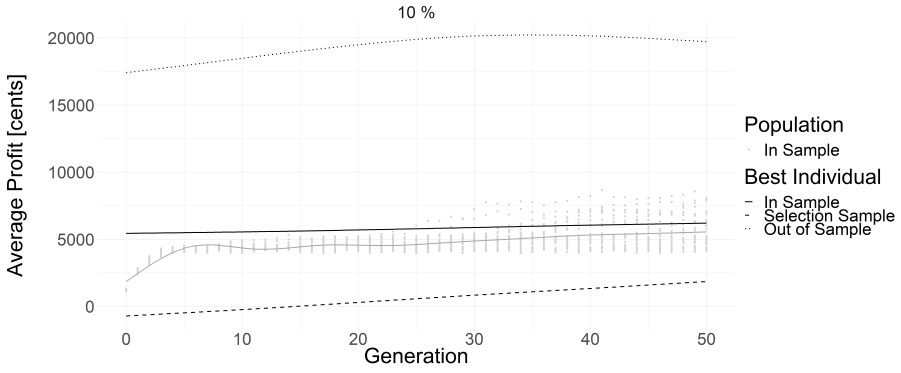


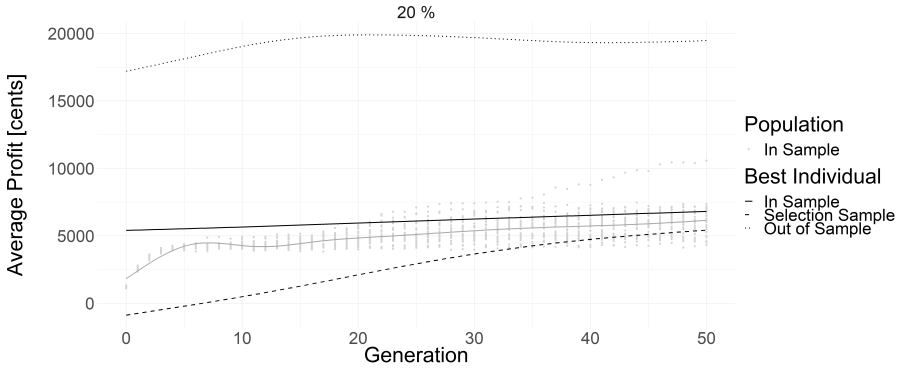

**Figure 5.36:** 3D Perspective plot of the mean LON $Sample_{in}$ profit results using various mutation probabilities.



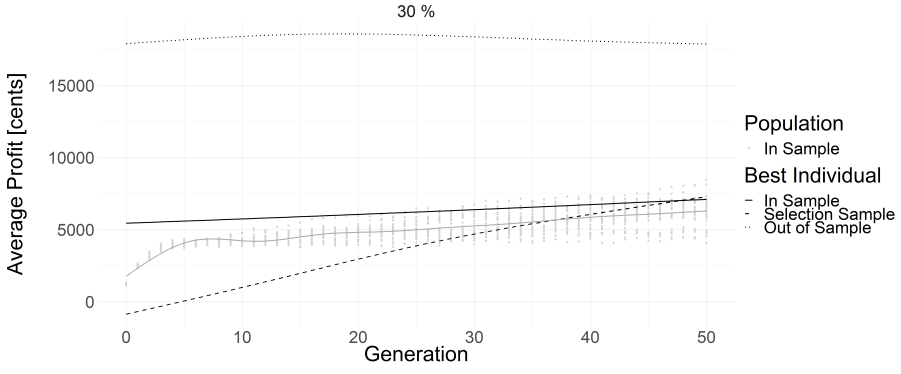

**(a)** Mutation probability set at 0%




**(b)** Mutation probability set at 10%

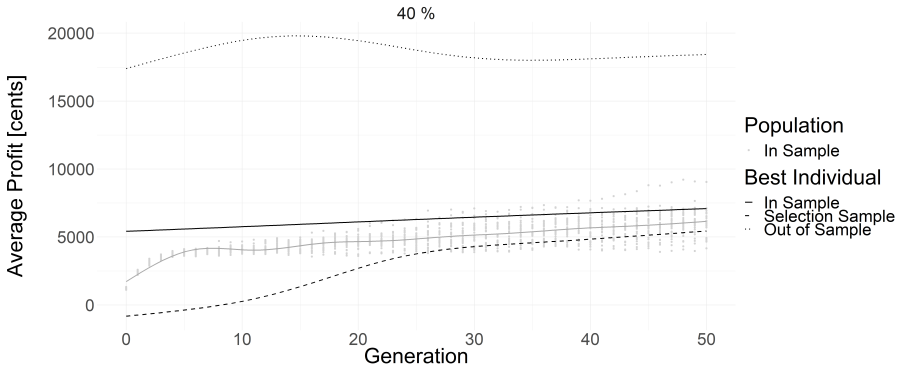



**(c)** Mutation probability set at 20%

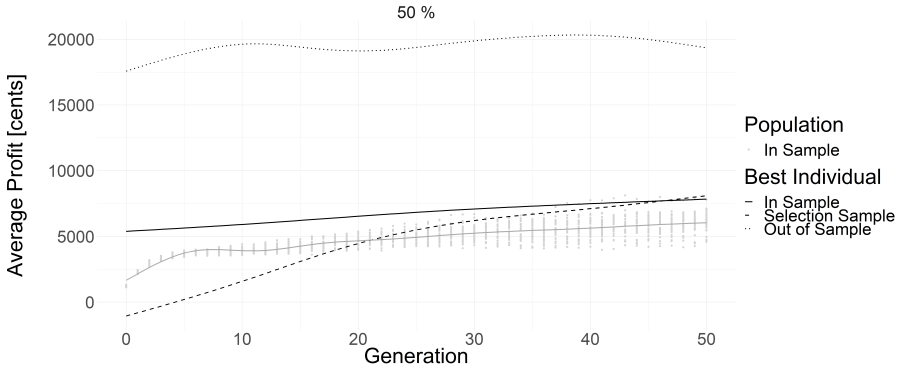



**(d)** Mutation probability set at 30%

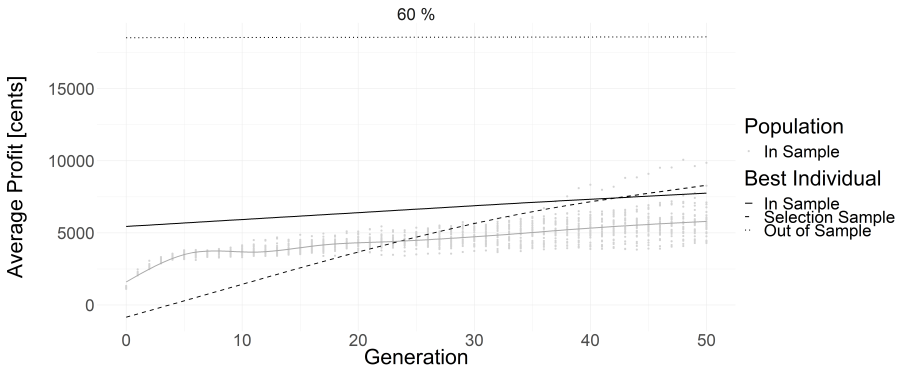
**Figure 5.37:** Scatter plot plotting returned profit using LON dataset and various mutation probabilities.



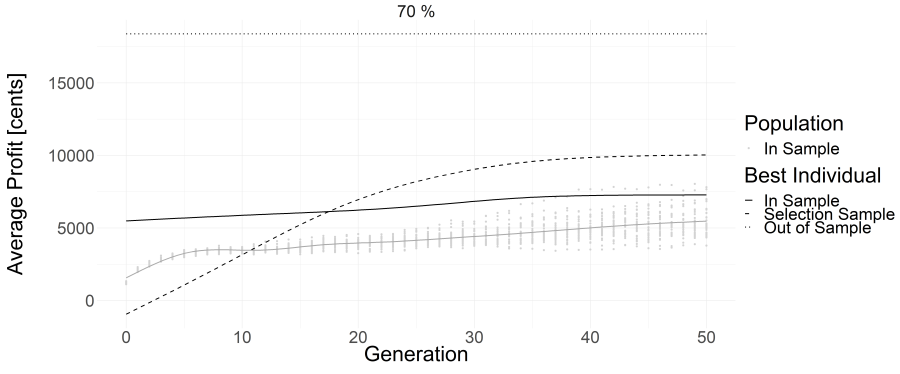

**(e)** Mutation probability set at 40%




**(f)** Mutation probability set at 50%

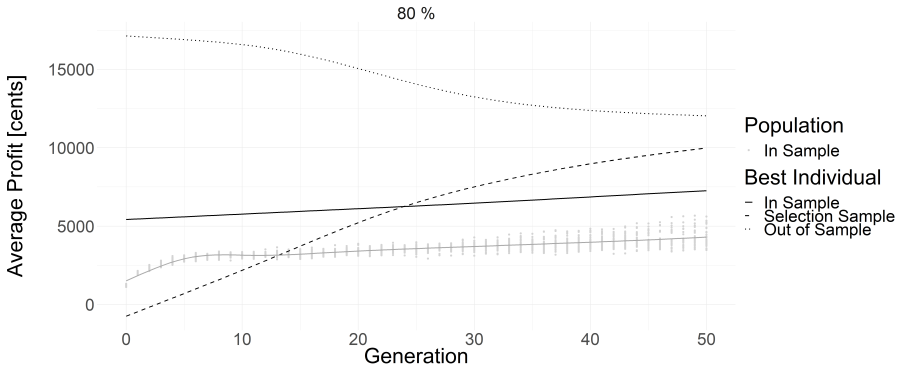



**(g)** Mutation probability set at 60%

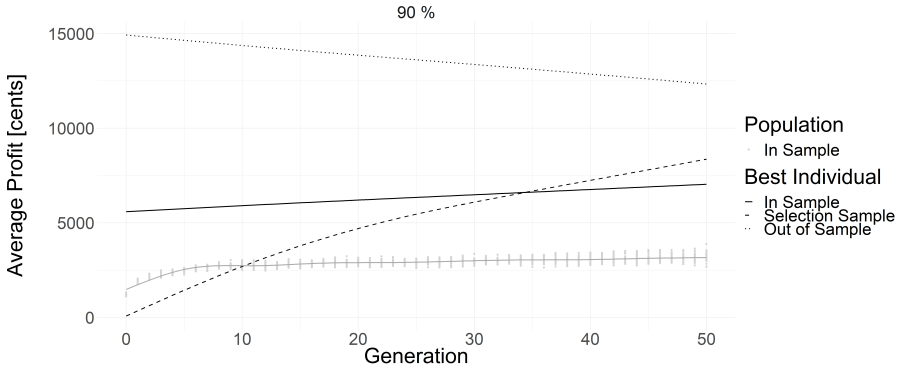



**(h)** Mutation probability set at 70%

**Figure 5.37:** Scatter plot plotting returned profit using LON dataset and various mutation probabilities (Cont.).




**(i)** Mutation probability set at 80%




**(j)** Mutation probability set at 90%

**Figure 5.37:** Scatter plot plotting returned profit using LON dataset and various mutation probabilities (Cont.).

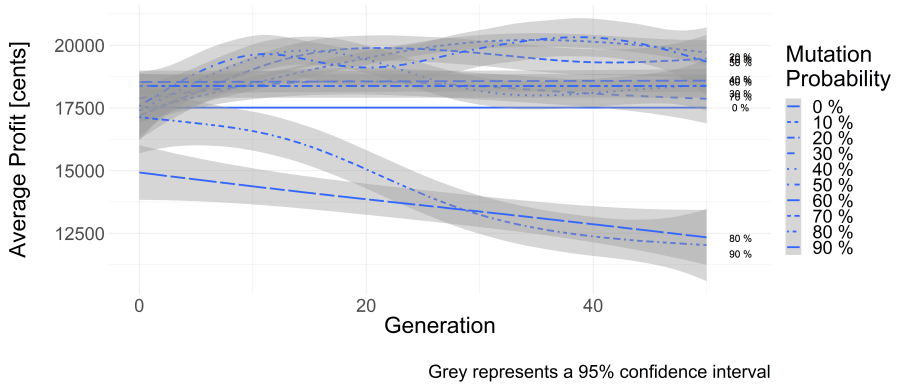



**Figure 5.38:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the LON dataset and various mutation probabilities.

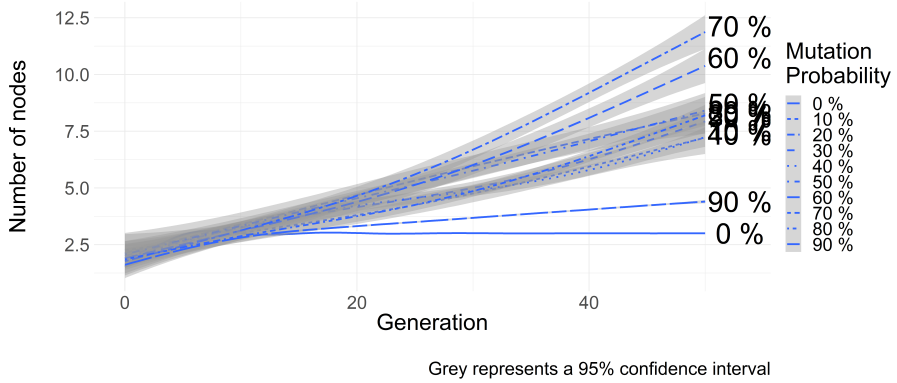



**Figure 5.39:** Average node count of an individual evolved using the LON dataset and various mutation probabilities.

Figure 5.42 shows that a mutation probability of 40% produced the trading rules that returned the largest $Sample_{out}$ profit using the NED dataset. Figure 5.40 shows no critical difference between the $Sample_{out}$ results returned by the trading rules evolved using a mutation probability of 40% and the results returned by the trading rules evolved using a mutation probability between 10% and 90%. A critical difference was found between the $Sample_{out}$ results of the trading rules evolved using a mutation probability of 10% and 90%. The critical difference plot confirms the p-values presented previously in Table 5.8g that shows no significant difference was found between the $Sample_{out}$ results of the trading rules evolved using a mutation probability of 20% and 80%. Figure 5.41 and Figure 5.43 show that the best $Sample_{in}$ results were obtained by the trading rules evolved using a mutation probability of 40%. The trading rules evolved with a mutation probability 40% returned an average $Sample_{in}$ profit greater than 2500 quicker than any other simulation, and trading rules evolved with a mutation probability between 10% and 40% reached an average population profit greater than 3000. Therefore, based on these observations, a mutation probability of 40% is preferred for the trading rules evolved for the NED dataset.

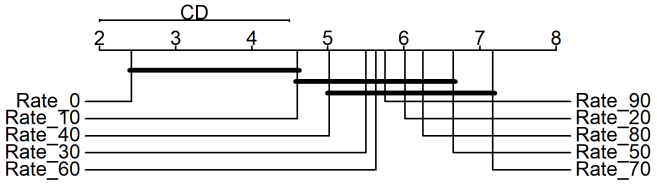
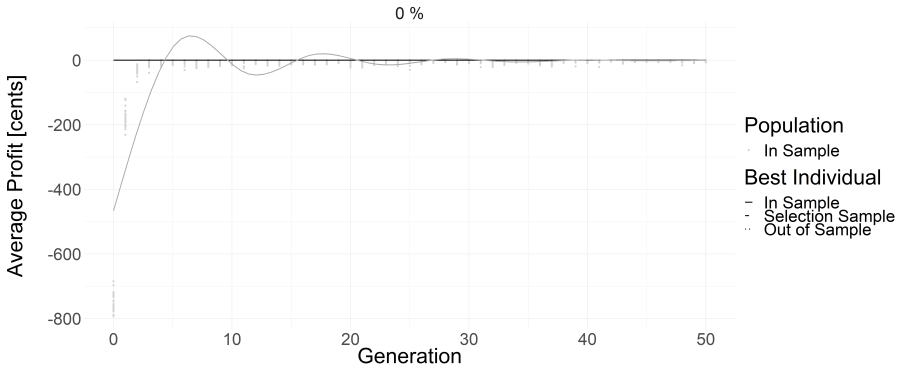
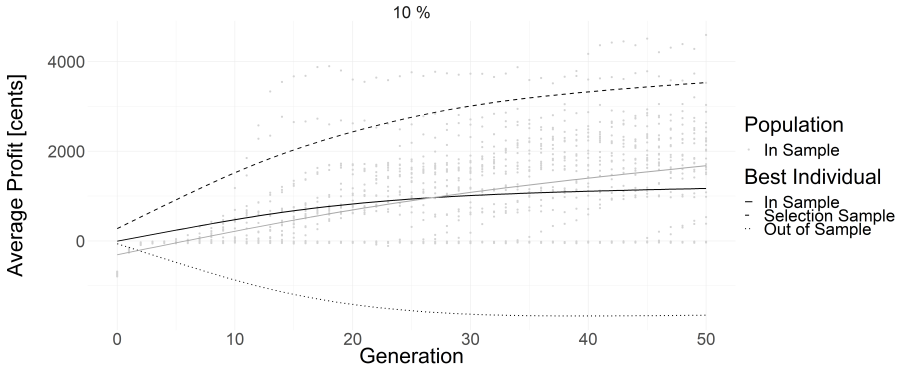


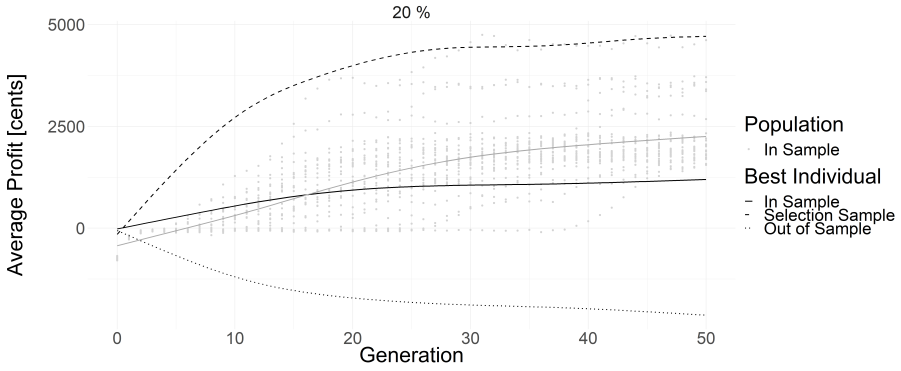

**Figure 5.40:** Critical difference plot of NED $Sample_{out}$ results using various mutation probabilities.



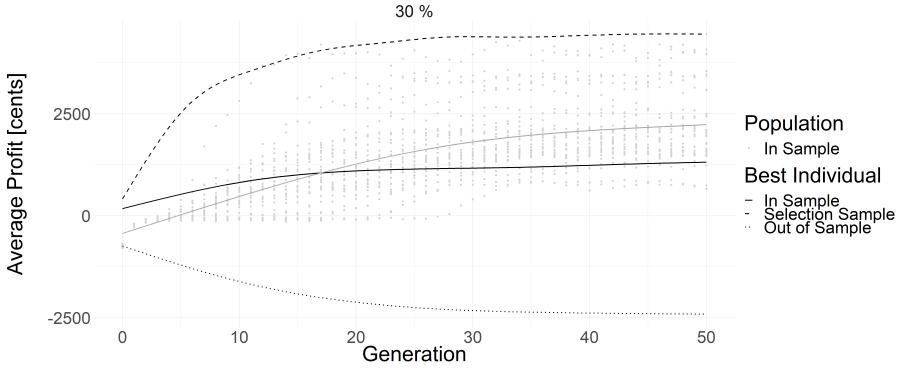

**(a)** Mutation probability set at 0%




**(b)** Mutation probability set at 10%

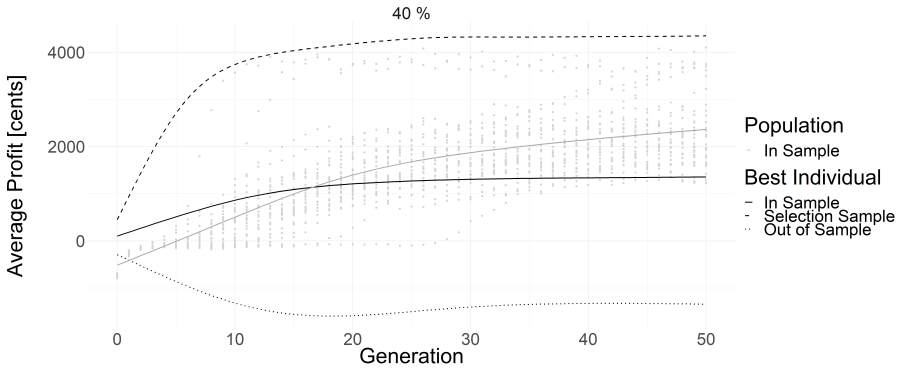



**(c)** Mutation probability set at 20%

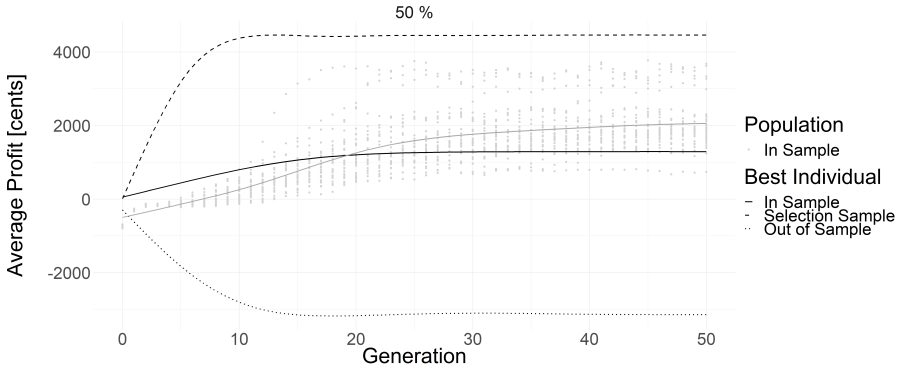



**(d)** Mutation probability set at 30%

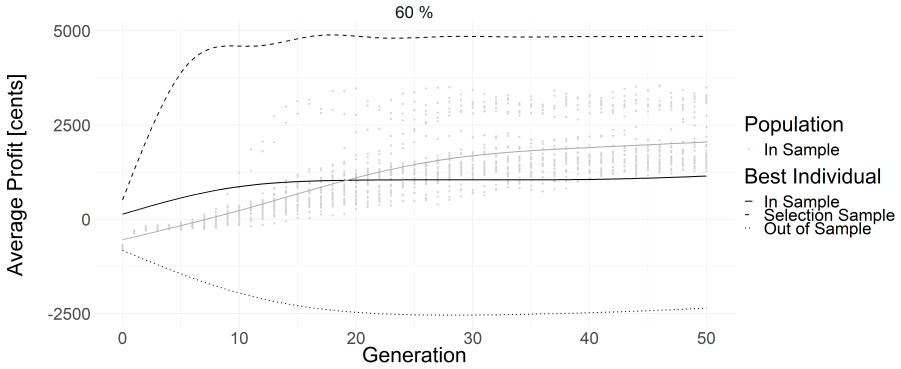
**Figure 5.41:** Scatter plot plotting returned profit using NED dataset and various mutation probabilities.



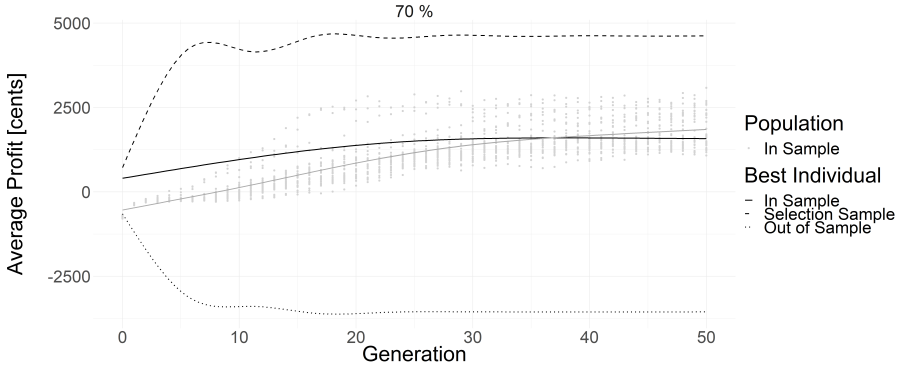

**(e)** Mutation probability set at 40%




**(f)** Mutation probability set at 50%

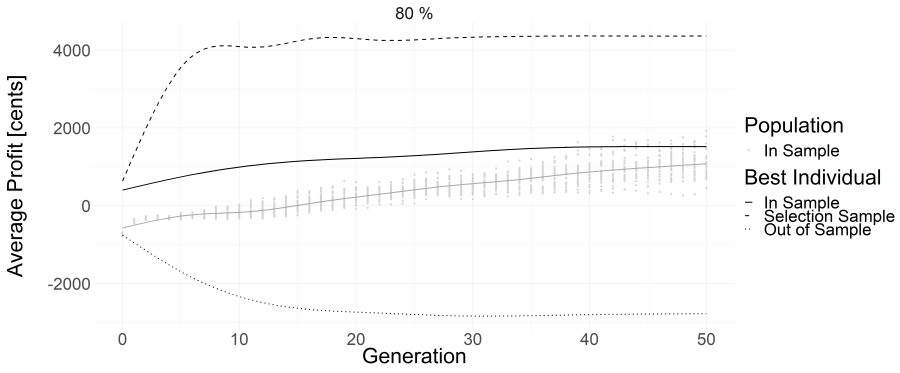



**(g)** Mutation probability set at 60%

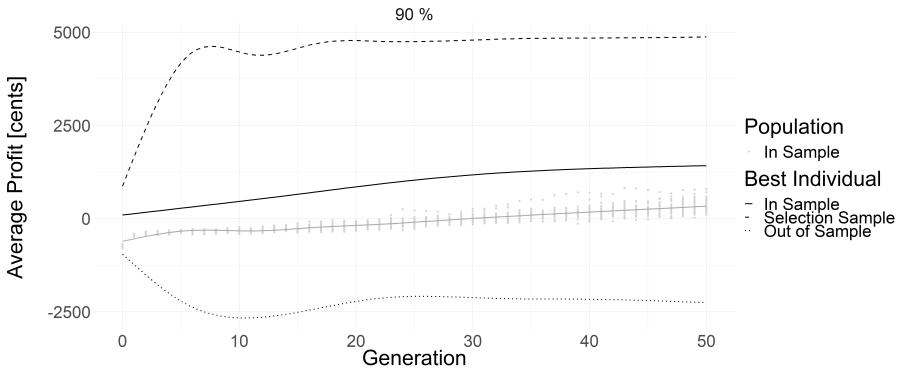



**(h)** Mutation probability set at 70%

**Figure 5.41:** Scatter plot plotting returned profit using NED dataset and various mutation probabilities (Cont.).




**(i)** Mutation probability set at 80%




**(j)** Mutation probability set at 90%

**Figure 5.41:** Scatter plot plotting returned profit using NED dataset and various mutation probabilities (Cont.).

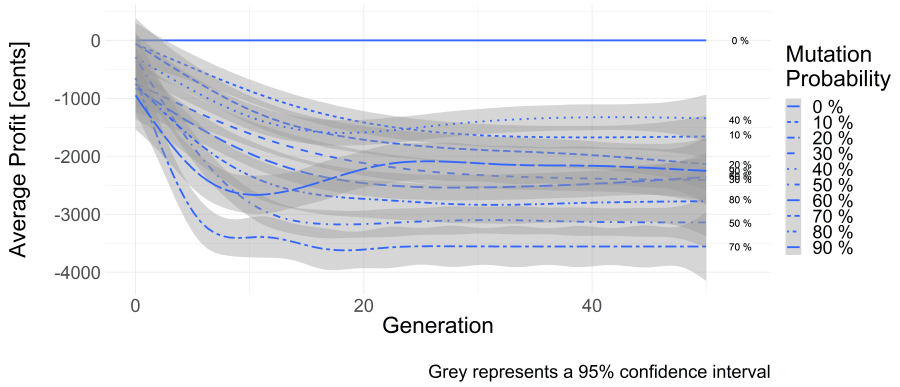



**Figure 5.42:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the NED dataset and various mutation probabilities.

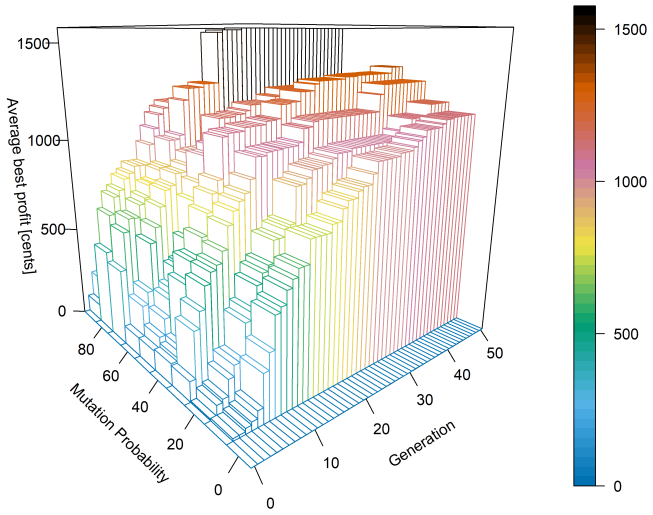



**Figure 5.43:** 3D Perspective plot of the mean NED $Sample_{in}$ profit results using various mutation probabilities.

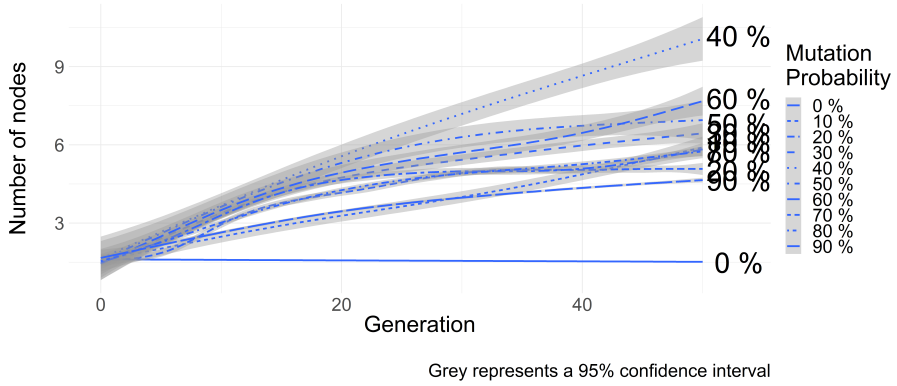



**Figure 5.44:** Average node count of an individual evolved using the NED dataset and various mutation probabilities.

Figure 5.48 shows that a mutation probability of 80% evolved trading rules that returned the highest $Sample_{out}$ profit for the RCH dataset. The second highest profit was returned by the trading rules evolved using a mutation probability between 30% and 70%. The critical difference plot in Figure 5.45 shows no critical difference between the $Sample_{out}$ results returned by the trading rules evolved using a mutation probability of 80% and those results returned by the trading rules evolved using a mutation

probability between 30% and 70%. Figure 5.47 and Figure 5.46 show that the average $Sample_{in}$ profit returned by the trading rules evolved using a mutation probability between 30% and 70% were higher than the profit returned by the trading rules evolved using a mutation probability of 80%. Figure 5.47 and Figure 5.46 show that only the trading rules evolved with a mutation probability between 20% and 50% returned an $Sample_{in}$ profit greater than 1500. Therefore, based on these observations, a mutation probability between 30% and 50% is preferred for the RCH dataset.

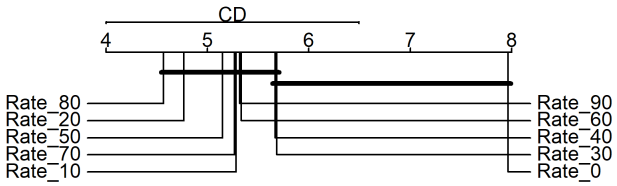



**Figure 5.45:** Critical difference plot of RCH $Sample_{out}$ results using various mutation probabilities.

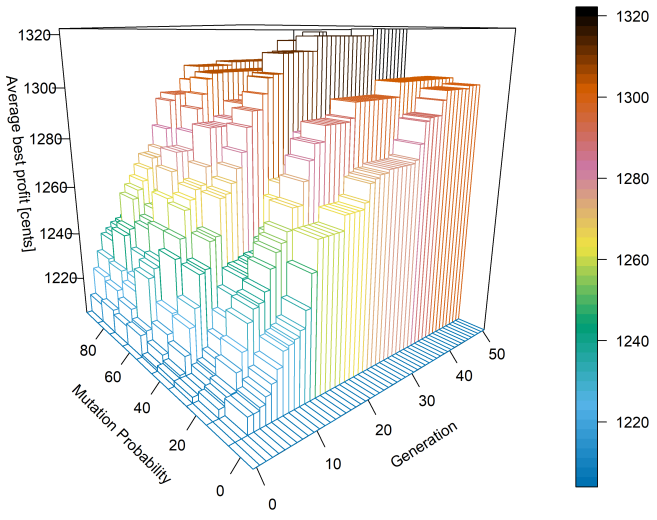
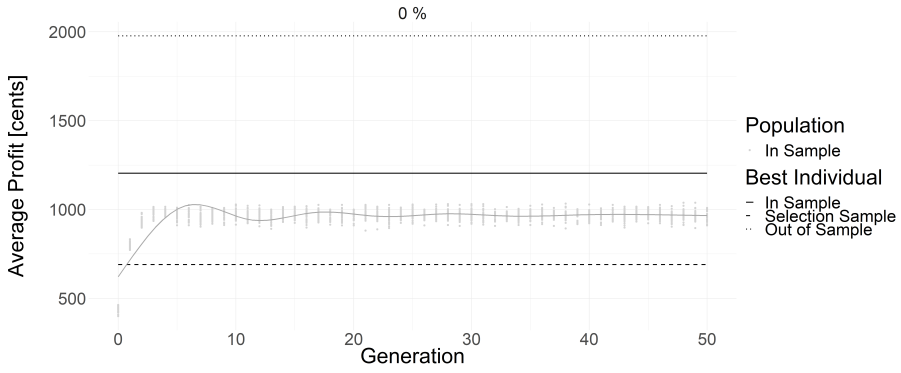
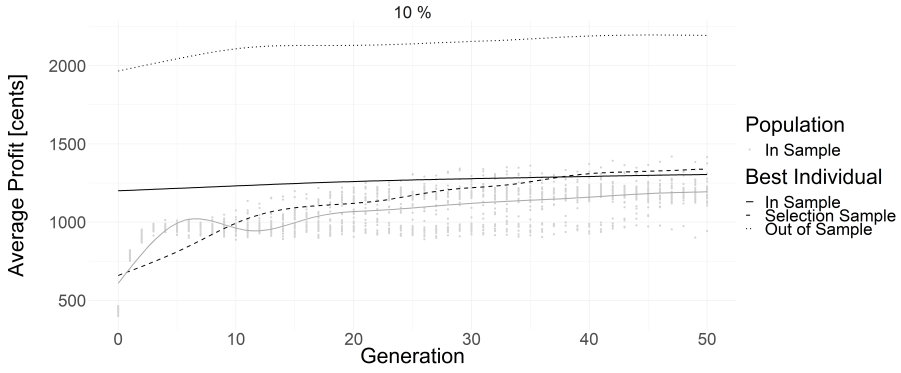


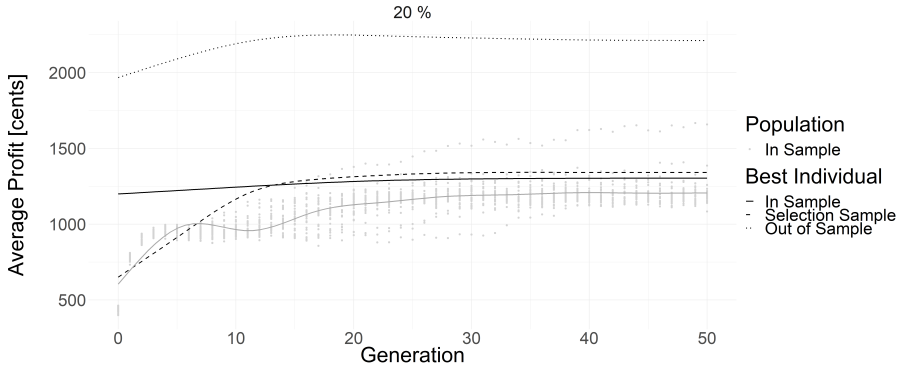

**Figure 5.46:** 3D Perspective plot of the mean RCH $Sample_{in}$ profit results using various mutation probabilities.



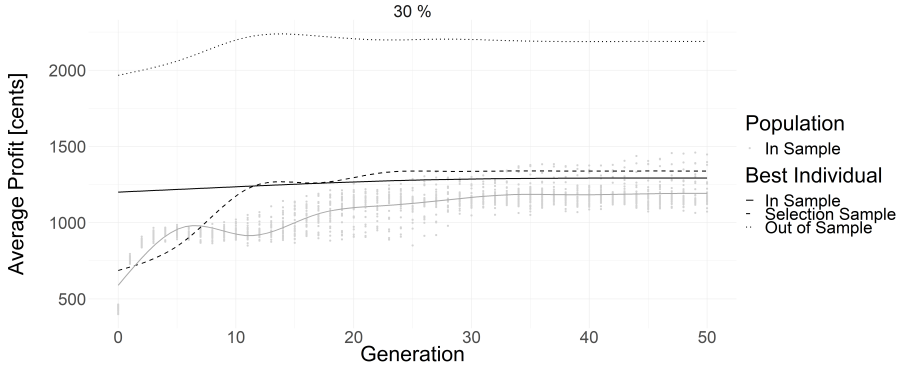

**(a)** Mutation probability set at 0%




**(b)** Mutation probability set at 10%

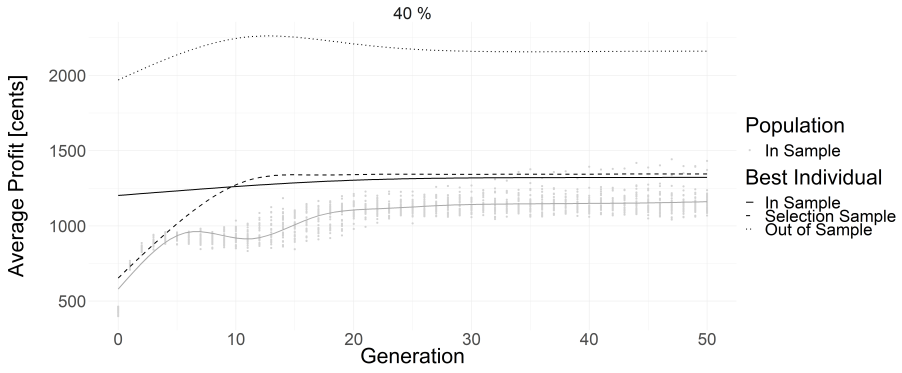



**(c)** Mutation probability set at 20%

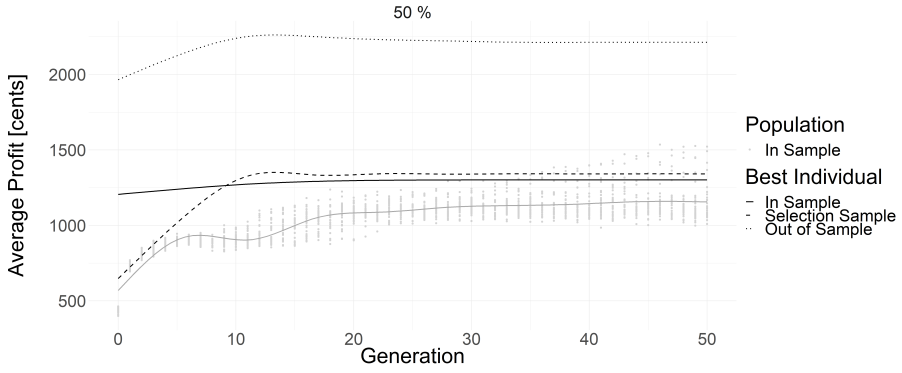



**(d)** Mutation probability set at 30%

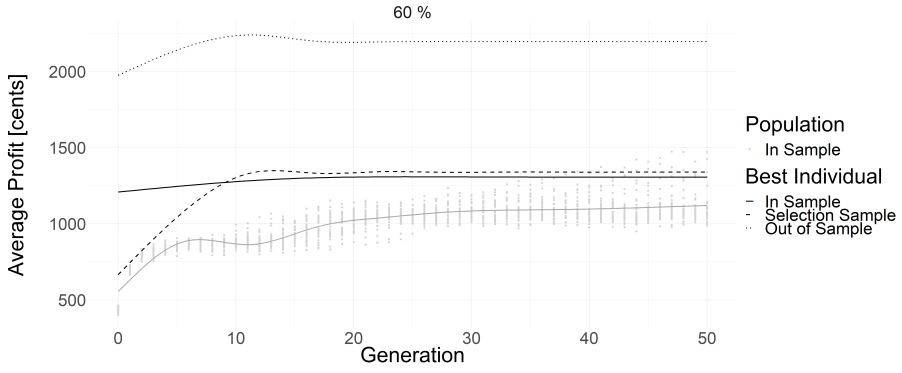
**Figure 5.47:** Scatter plot plotting returned profit using RCH dataset and various mutation probabilities.



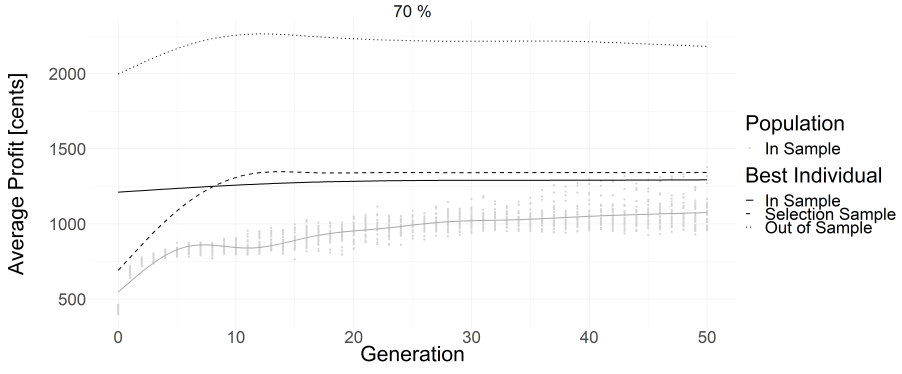

**(e)** Mutation probability set at 40%




**(f)** Mutation probability set at 50%

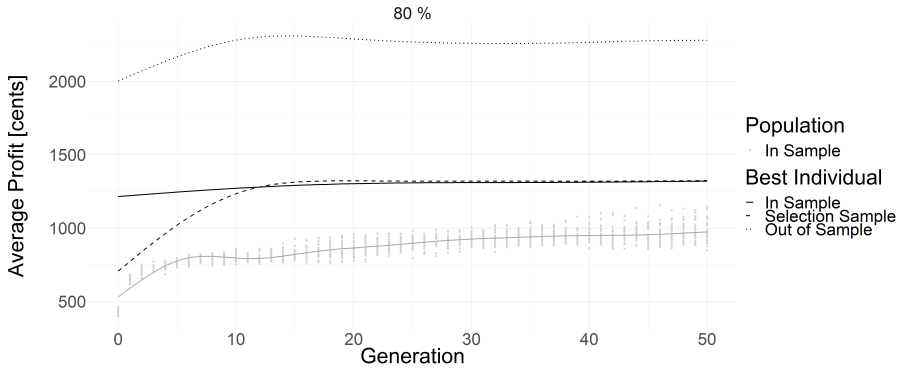



**(g)** Mutation probability set at 60%

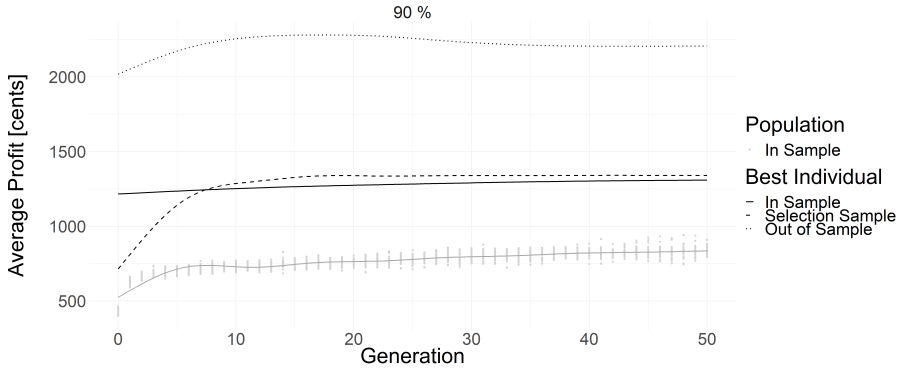



**(h)** Mutation probability set at 70%

**Figure 5.47:** Scatter plot plotting returned profit using RCH dataset and various mutation probabilities (Cont.).




**(i)** Mutation probability set at 80%




**(j)** Mutation probability set at 90%

**Figure 5.47:** Scatter plot plotting returned profit using RCH dataset and various mutation probabilities (Cont.).

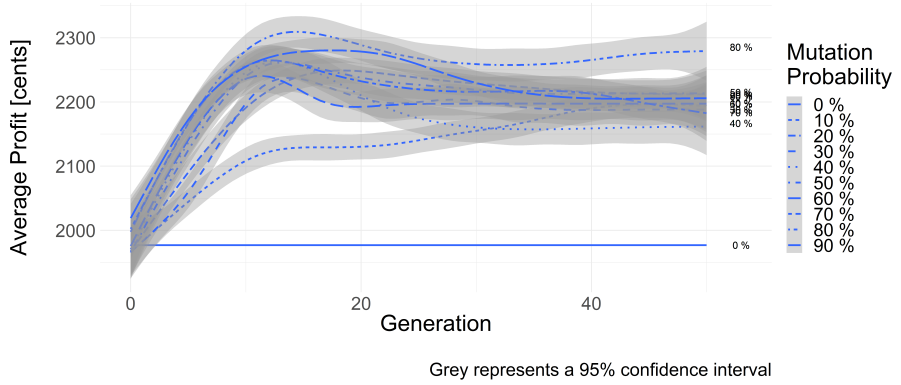



**Figure 5.48:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the RCH dataset and various mutation probabilities.

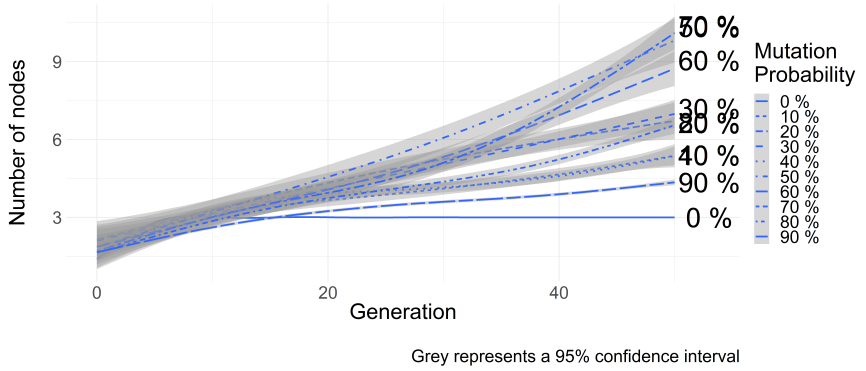



**Figure 5.49:** Average node count of an individual evolved using the RCH dataset and various mutation probabilities.

The results presented in this section vary depending on the dataset used. The results presented in this section found that a 70% mutation probability resulted in the best performing trading rules when trading LON. This result contradicts the results returned by the trading rules evolved for other datasets. Perhaps a variable mutation probability that begins high and decreases over time is preferred when evolving trading rules. Further research is required to study the effects of a variable mutation probability on the results returned by evolved stock market traders.

A mutation probability of 30% and 40% evolved the best trading rules when trading BIL and NED respectively. A mutation probability between 30% and 40% was presented as the preferred mutation probability when evolving trading rules with ALSI40, INVNED, INVREM, INVSBK, and RCH datasets. Therefore, based on these observations, a mutation probability of 35% was selected as the preferred mutation probabilities the remainder of this thesis.

### 5.5.2 Population Size Sensitivity Analysis

The results of the $2^4 30$ factorial design showed that the population size has the largest impact on the outcome of the evolutionary process. Each individual within a population represents a potential trading rule. More individuals result in a greater initial genetic diversity within the population and more ways in which crossover can optimise the trading rule. However, at which point does the number of individuals in a population become large enough to contain all the variations of genetic elements so that any new individual within the population has no effect on the outcome? At which point is the population too small to contain a sufficient set of genetic elements for crossover to provide any benefit?

Allen and Karjalainen [43] used a population size of 500. Simulations are run in this section to determine the preferred population size. The simulations were configured to use the control parameter

values in Table 5.3. Rank selection was used as the selection strategy. The first simulation was configured with a population size of 20. Each subsequent simulation doubled the population size until 6400 individuals were configured for a total of eight simulations. The results show that a population size of 1600 is preferred.

The simulations were repeated for each share presented in Table 5.2. Following the empirical process introduced in Section 5.2.1, the null-hypothesis was assumed. The null-hypothesis assumes that the results returned by traders evolved using various configured population sizes are not significantly different. The Iman and Davenport [142] test results presented in Table 5.22 rejected the null-hypothesis for most of the samples. For those samples where the null-hypothesis was rejected, a post-hoc analysis was done and is presented in Figure 5.50. Figure 5.50 shows a significant difference in performance between a low population size and a large population size. Each set of results were analysed individually to determine the effect that population size had on the profit generated by the evolved trading rules.

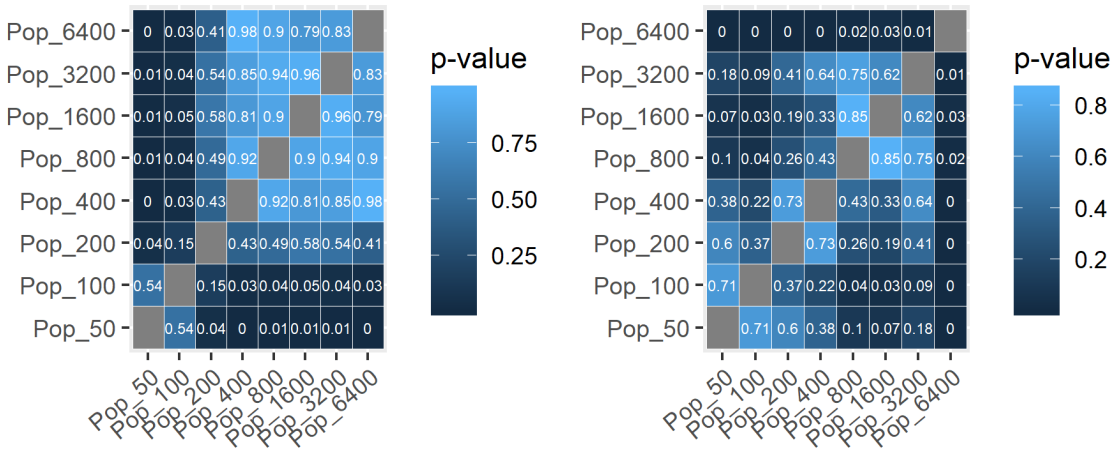
Figure 5.50a shows that a significant difference exists between the $Sample_{out}$ results obtained by the trading rules evolved using the AGL dataset and a population size of 50, and any population size greater than 50. Figure 5.52 shows that the lowest average $Sample_{out}$ profit was generated by the trading rules evolved using a population size of 50 and 100. The critical difference plot in Figure 5.51 does not show a critical difference between the $Sample_{out}$ results, but does show how similar the $Sample_{out}$ results are for trading rules evolved using a population size of 200, 400, 800, 1600, 3200, and 6400.

Figure 5.53 shows that the trading rules evolved using the AGL dataset and a population size greater than 800 returned a $Sample_{in}$ profit greater than 10000 within 50 generations. Therefore, based on these observations, a population size between 800 and 6400 is preferred for evolving trading rules for the AGL dataset.

**Table 5.22:** p-value results using Iman and Davenport test with Finner's correction.

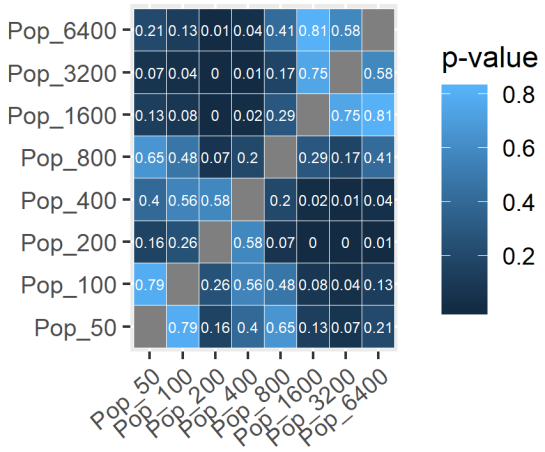
| | $Sample_{in}$ | | $Sample_{sel}$ | | $Sample_{out}$ | |
|---|---|---|---|---|---|---|
| **Share Code** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** |
| **agl** | **0.000018** | **0.000028** | **0.000328** | **0.000703** | **0.014381** | **0.026795** |
| **alsi** | **0.000000** | **0.000000** | 0.084855 | 0.097255 | 0.320479 | 0.383045 |
| **bil** | **0.000000** | **0.000000** | **0.046116** | 0.057309 | **0.000280** | **0.001048** |
| **gfi** | **0.000000** | **0.000000** | **0.012699** | **0.017277** | **0.008051** | **0.023959** |
| **imp** | 0.958187 | 0.958187 | 0.967481 | 0.967481 | 0.975481 | 0.978529 |
| **inv-ned** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **inv-rem** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **inv-sbk** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **lon** | **0.000000** | **0.000000** | **0.000001** | **0.000002** | 0.264645 | 0.342418 |
| **ned** | **0.029597** | **0.034072** | **0.005943** | **0.009885** | **0.010857** | **0.024127** |
| **rch** | **0.000111** | **0.000151** | **0.001402** | **0.002627** | 0.066452 | 0.098003 |
| **rem** | **0.045741** | **0.048927** | 0.253427 | 0.268850 | 0.972263 | 0.978529 |
| **sab** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.009722** | **0.024127** |
| **sbk** | **0.012643** | **0.015778** | **0.006889** | **0.010316** | 0.933219 | 0.955962 |
| **sol** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.031862** | 0.052538 |

Note: **bold** values represent a significant difference.

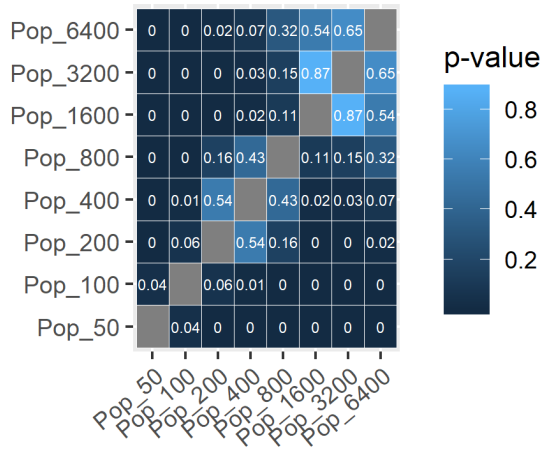



**(a)** $Sample_{out}$ AGL **(b)** $Sample_{out}$ BIL

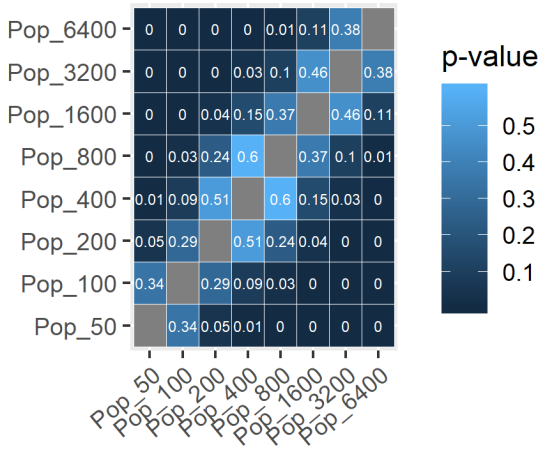
**Figure 5.50:** Friedman pairwise comparison of p-values for results.



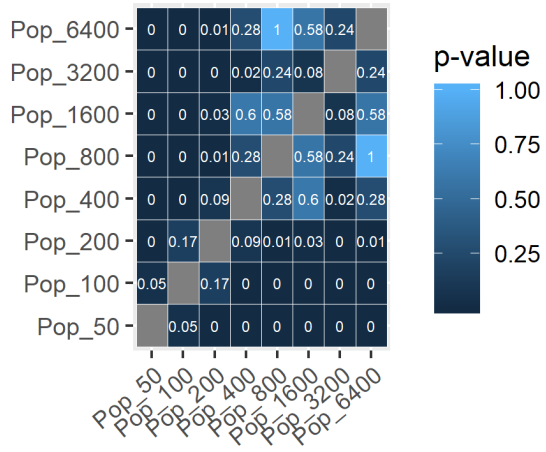

**(c)** $Sample_{out}$ GFI



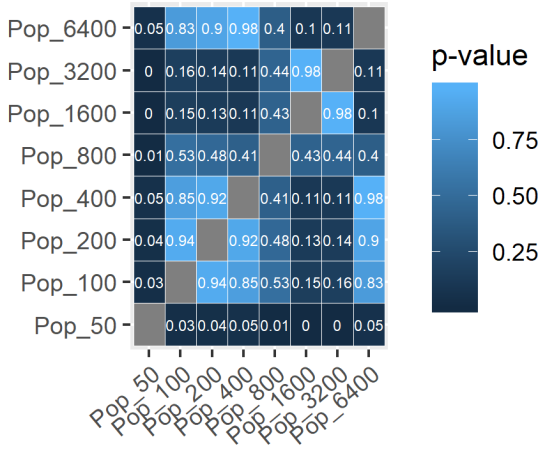

**(d)** $Sample_{out}$ INV-NED



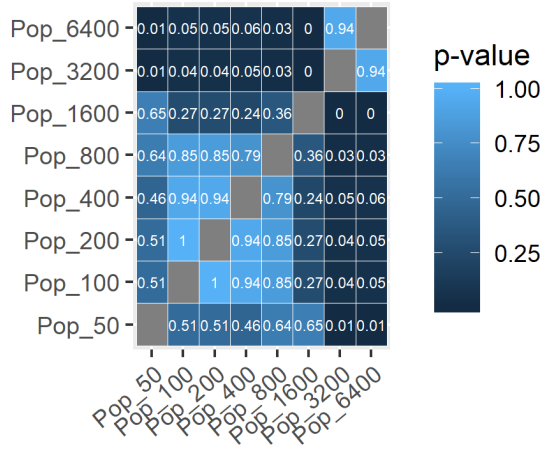

**(e)** $Sample_{out}$ INV-REM




**(f)** $Sample_{out}$ INV-SBK




**(g)** $Sample_{out}$ NED




**(h)** $Sample_{out}$ SAB

**Figure 5.50:** Friedman pairwise comparison of p-values for results (Cont.).

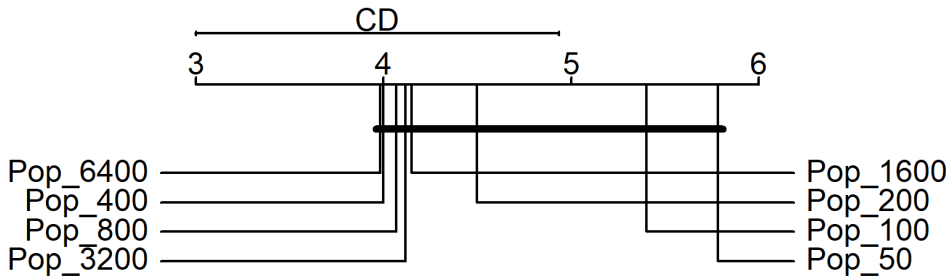



**Figure 5.51:** Critical difference plot of AGL $Sample_{out}$ results using various population sizes.

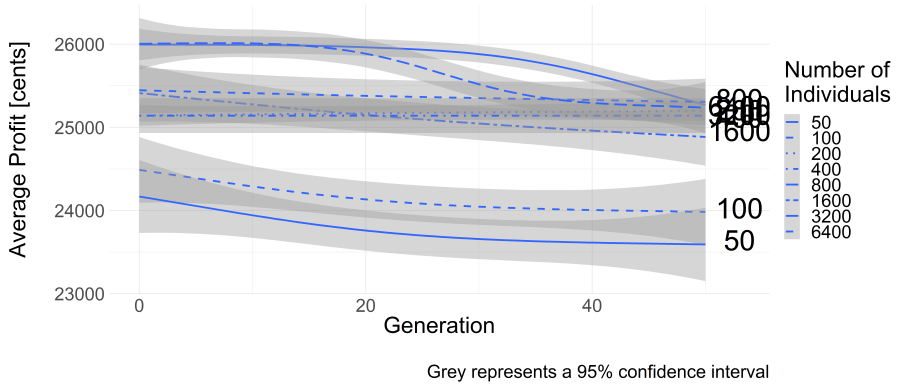
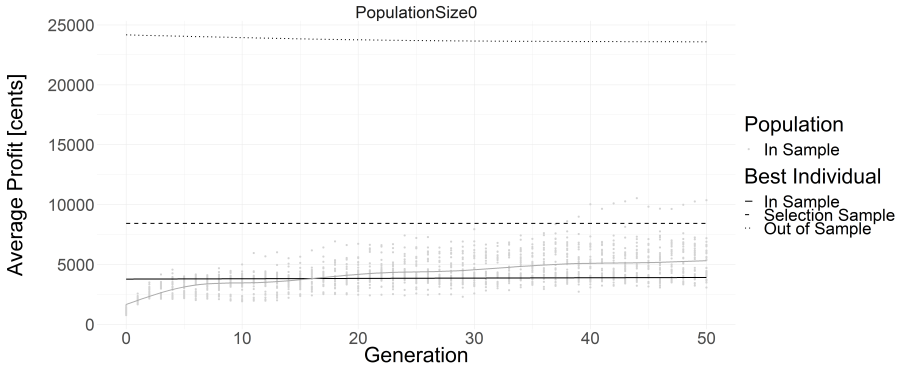
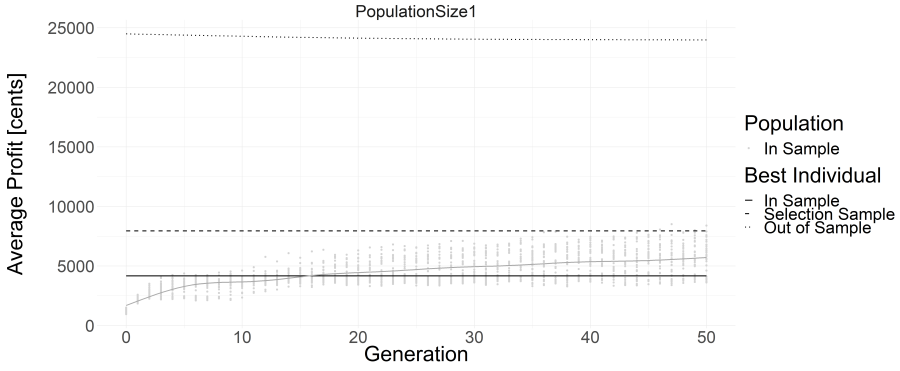



**Figure 5.52:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the AGL dataset and various population sizes.

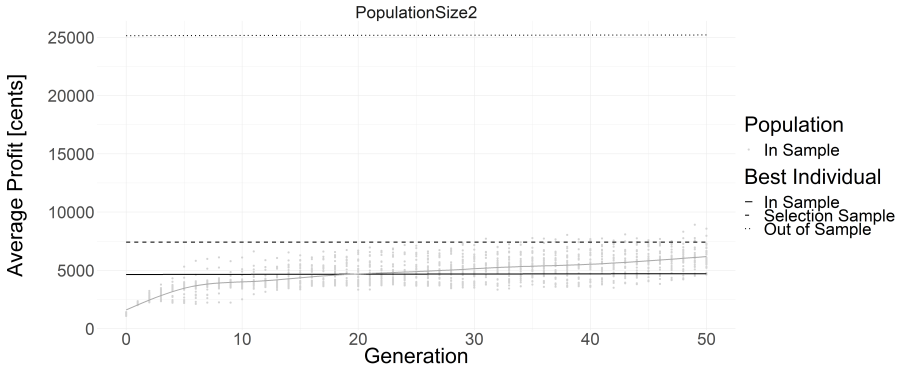



**(a)** Population size set at 50 individuals

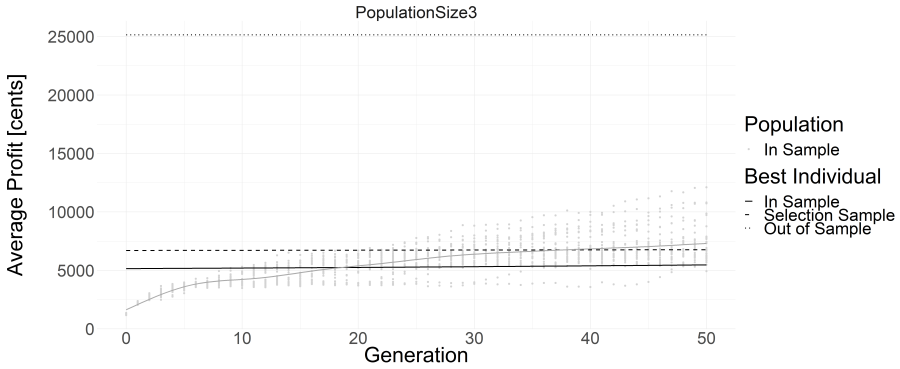
**Figure 5.53:** Scatter plot plotting returned profit using AGL dataset and various population sizes.



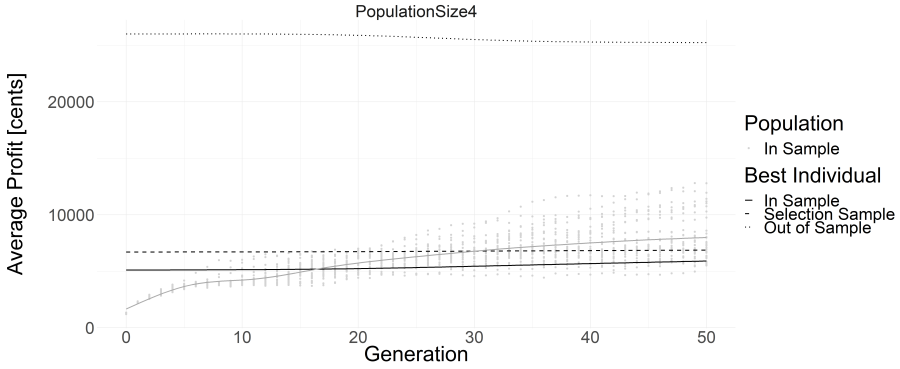

**(b)** Population size set at 100 individuals




**(c)** Population size set at 200 individuals

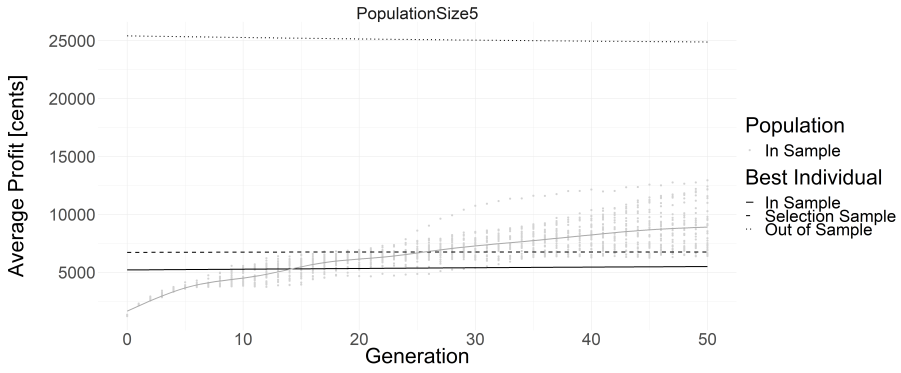



**(d)** Population size set at 400 individuals

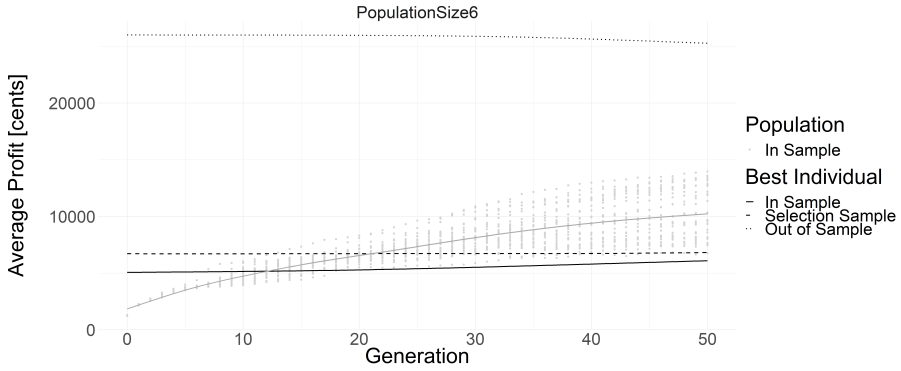



**(e)** Population size set at 800 individuals

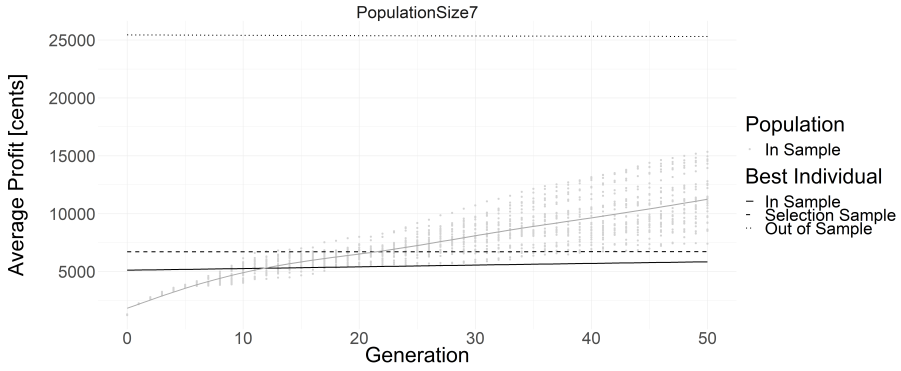
**Figure 5.53:** Scatter plot plotting returned profit using AGL dataset and various population sizes (Cont.).




**(f)** Population size set at 1600 individuals




**(g)** Population size set at 3200 individuals




**(h)** Population size set at 6400 individuals

**Figure 5.53:** Scatter plot plotting returned profit using AGL dataset and various population sizes (Cont.).

Figure 5.50b shows that a significant difference exists between the $Sample_{out}$ results obtained by the trading rules evolved using the BIL dataset and a population size of 6400, and trading rules evolved using any population size less than 6400. The p-value results are confirmed by the critical difference

plot in Figure 5.54. Figure 5.56 shows that the largest profit was returned after 50 generations by the trading rules evolved using a population size of 100. Figure 5.56 shows that the trading rules evolved with 6400 individuals performed the worst after 50 generations. The scatter plot in Figure 5.55 shows that the trading rules that returned a $Sample_{in}$ profit greater than 5000 were those trading rules evolved using more than 200 individuals. Therefore, based on these observations, a population size between 200 and 3200 is preferred when evolving trading rules for the BIL dataset.

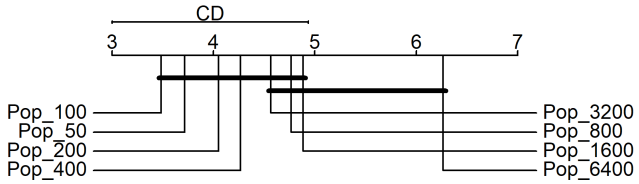
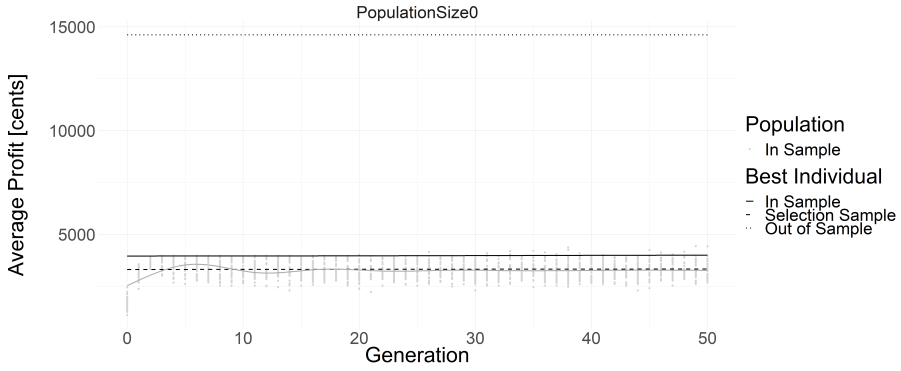
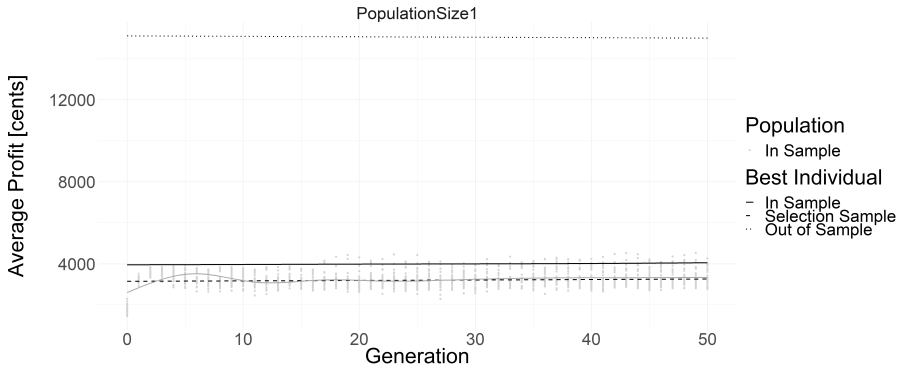



**Figure 5.54:** Critical difference plot of BIL $Sample_{out}$ results using various population sizes.

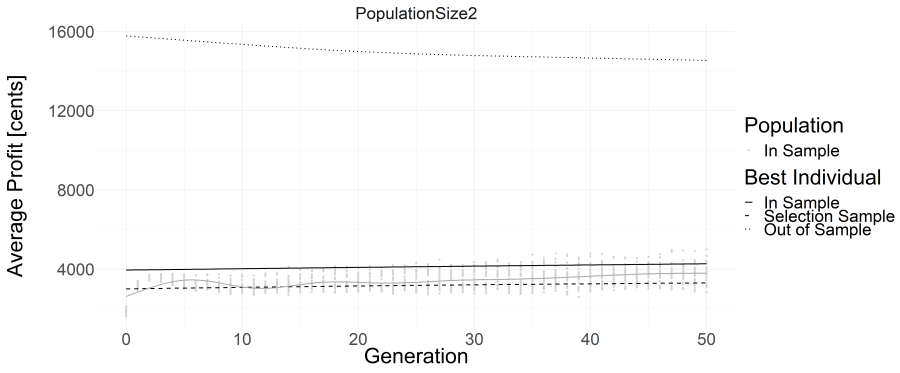



**(a)** Population size set at 50 individuals

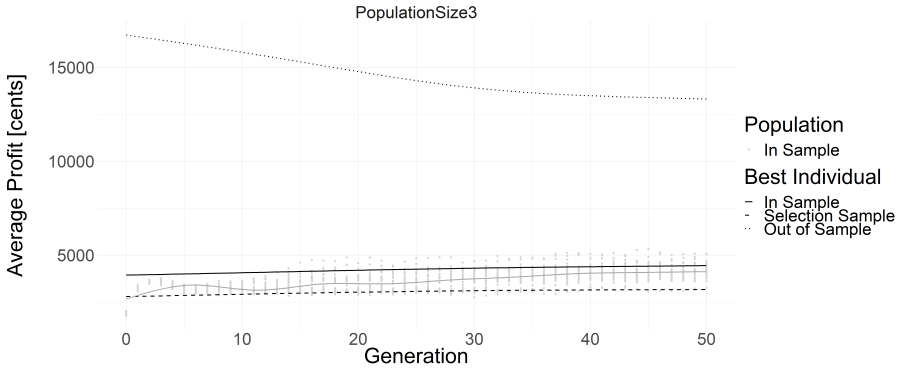



**(b)** Population size set at 100 individuals

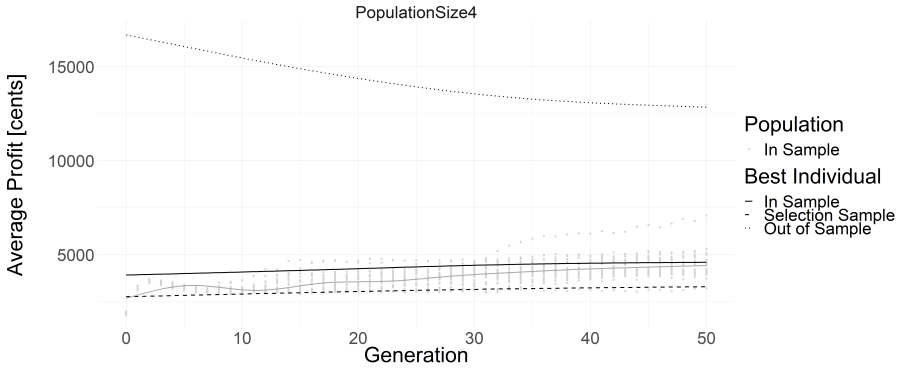
**Figure 5.55:** Scatter plot plotting returned profit using BIL dataset and various population sizes.



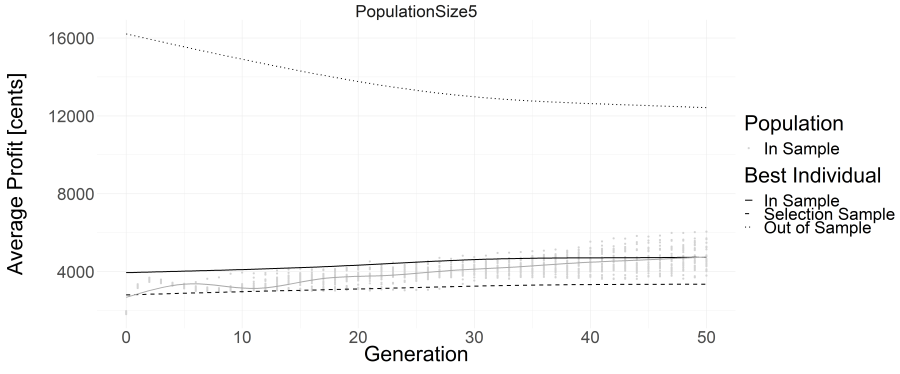

**(c)** Population size set at 200 individuals




**(d)** Population size set at 400 individuals

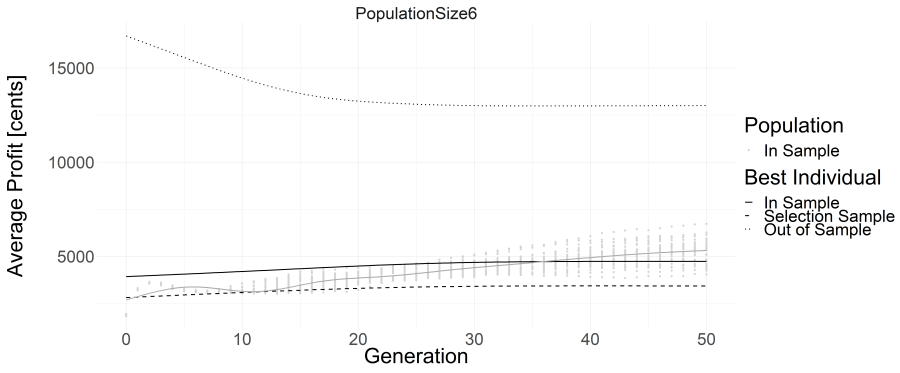



**(e)** Population size set at 800 individuals

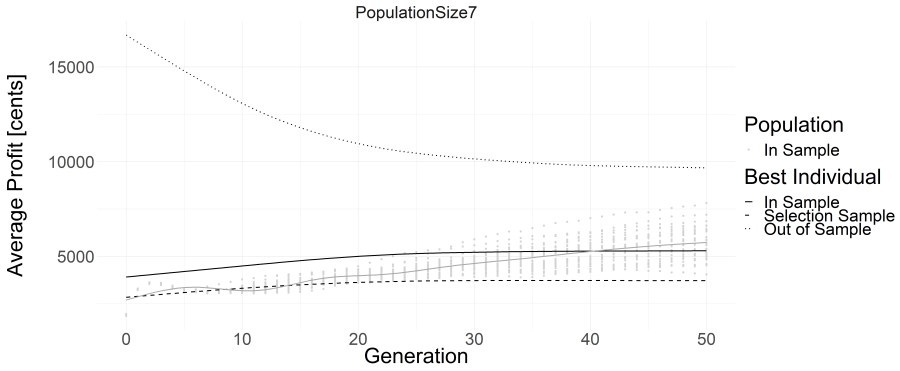



**(f)** Population size set at 1600 individuals

**Figure 5.55:** Scatter plot plotting returned profit using BIL dataset and various population sizes (Cont.).




**(g)** Population size set at 3200 individuals




**(h)** Population size set at 6400 individuals

**Figure 5.55:** Scatter plot plotting returned profit using BIL dataset and various population sizes (Cont.).

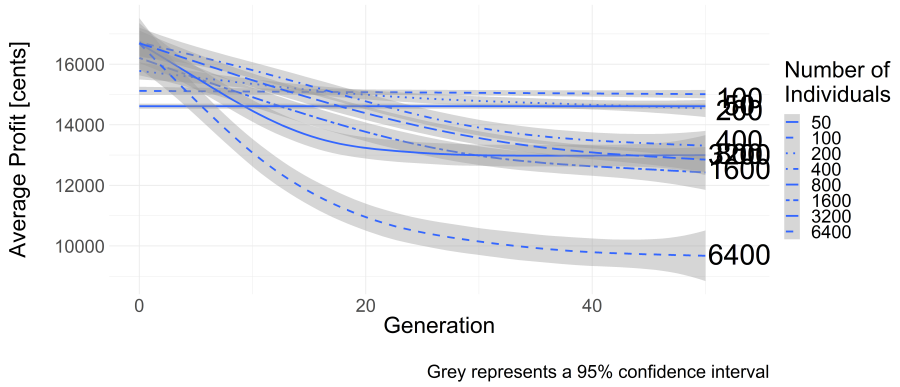



**Figure 5.56:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the BIL dataset and various population sizes.

Figure 5.59 shows three different profit bands returned by the trading rules evolved for the GFI dataset, i.e. the lowest $Sample_{out}$ profit returned by the trading rules evolved using a population size of 200 and 400, the second lowest $Sample_{out}$ results returned by the trading rules evolved using a population size of 50, 100, and 800, and the lowest profit band using a population size of 6400. The best $Sample_{out}$ performing trading rules were evolved using a population size of 1600, 3200, 6400. The $Sample_{out}$ results are consistent with the p-value results presented Figure 5.50c. The p-value results show that the $Sample_{out}$ results returned by the trading rules evolved using a population size of 200 and 400 are significantly different to the results returned by the trading rules evolved using a population size of 3200. The critical difference plot in Figure 5.57 confirms these observations and presents two groups of $Sample_{out}$ results: $Sample_{out}$ results returned by the trading rules evolved using a population size of 50, 100, 200, and 400, and $Sample_{out}$ results returned by the trading rules evolved using a population size of 1600, 3200, and 6400. The $Sample_{in}$ results presented as a scatter plot in Figure 5.58 show that only the trading rules evolved with a population size of 3200 and 6400 returned a $Sample_{in}$ profit greater than 7500. Figure 5.58 also shows that the trading rules evolved using a population size greater than 800 reached an average best $Sample_{in}$ profit of 2500 before 15 generations. Therefore, based on these observations, a population size of 1600, 3200, or 6400 is preferred when evolving trading rules for the GFI dataset.

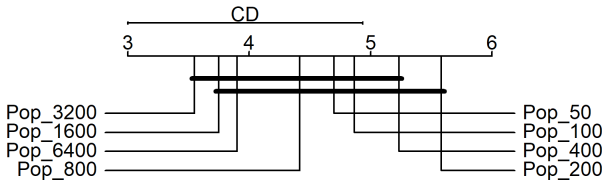
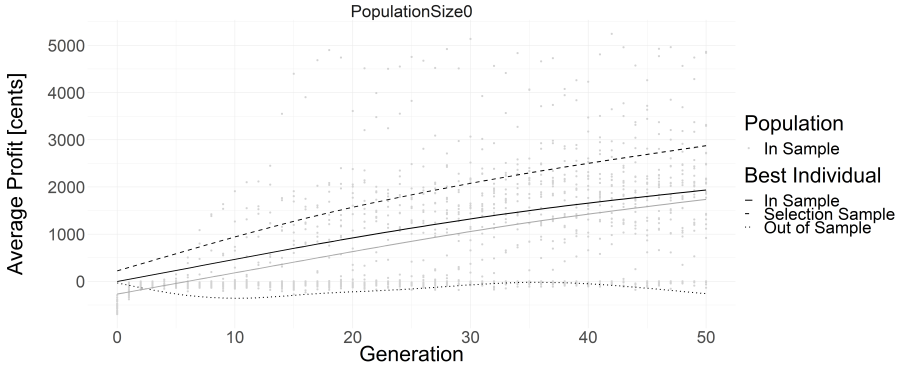
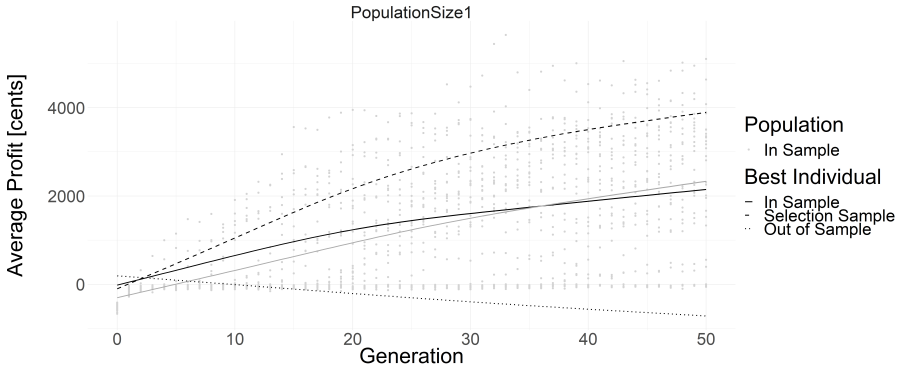


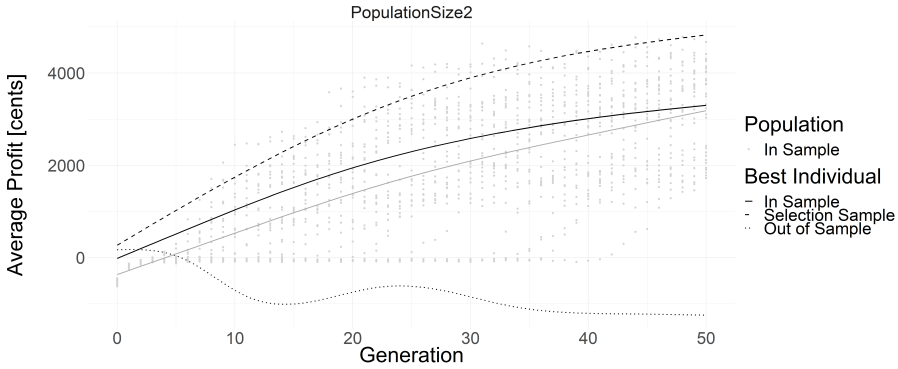

**Figure 5.57:** Critical difference plot of GFI $Sample_{out}$ results using various population sizes.



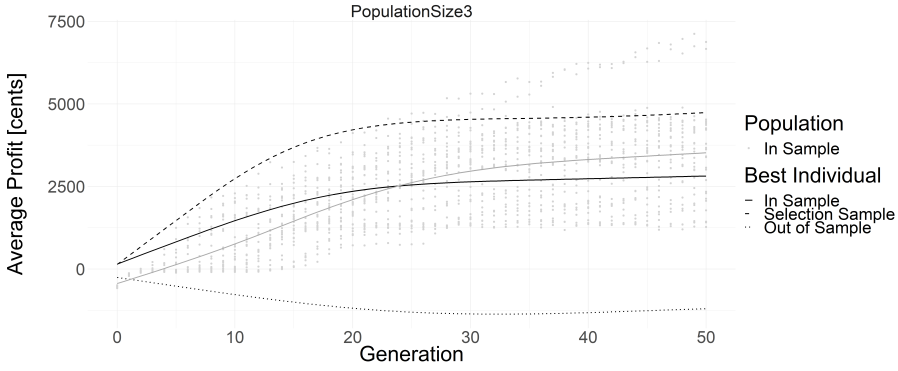

**(a)** Population size set at 50 individuals




**(b)** Population size set at 100 individuals

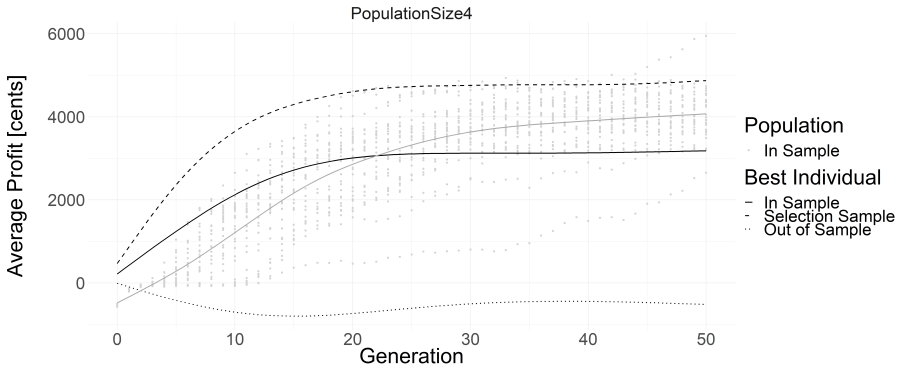



**(c)** Population size set at 200 individuals

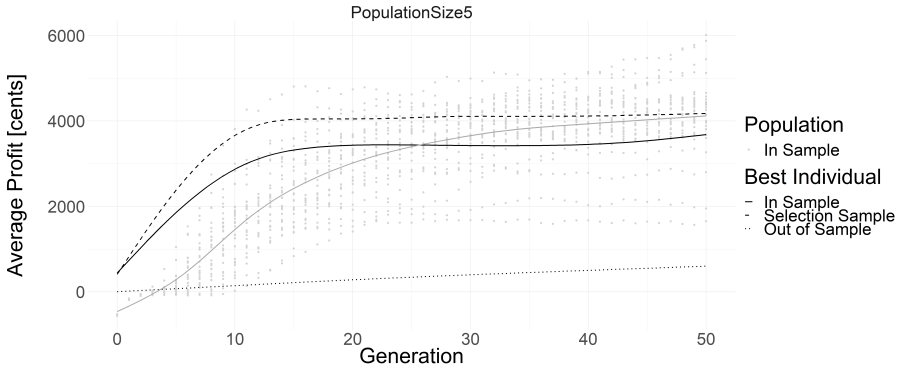



**(d)** Population size set at 400 individuals

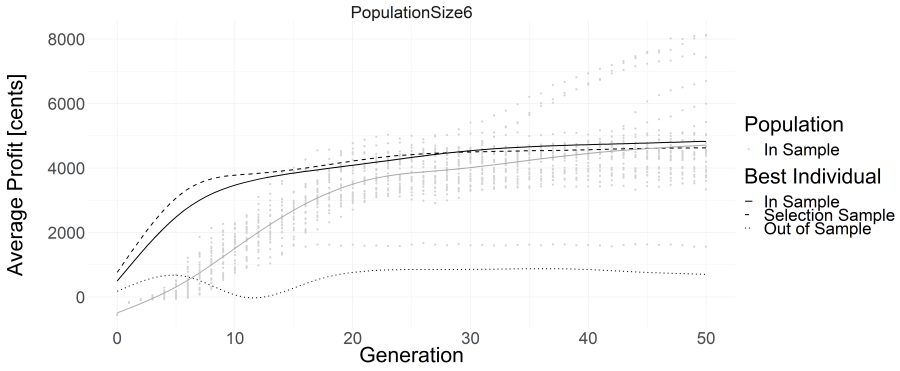
**Figure 5.58:** Scatter plot plotting returned profit using GFI dataset and various population sizes.



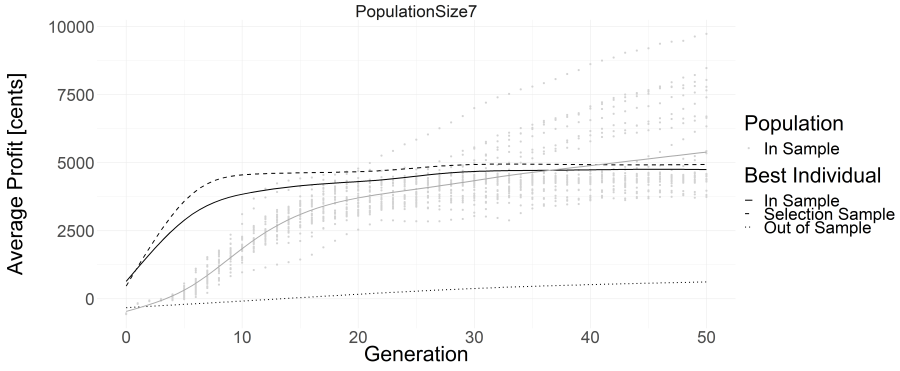

**(e)** Population size set at 800 individuals




**(f)** Population size set at 1600 individuals




**(g)** Population size set at 3200 individuals




**(h)** Population size set at 6400 individuals

**Figure 5.58:** Scatter plot plotting returned profit using GFI dataset and various population sizes (Cont.).

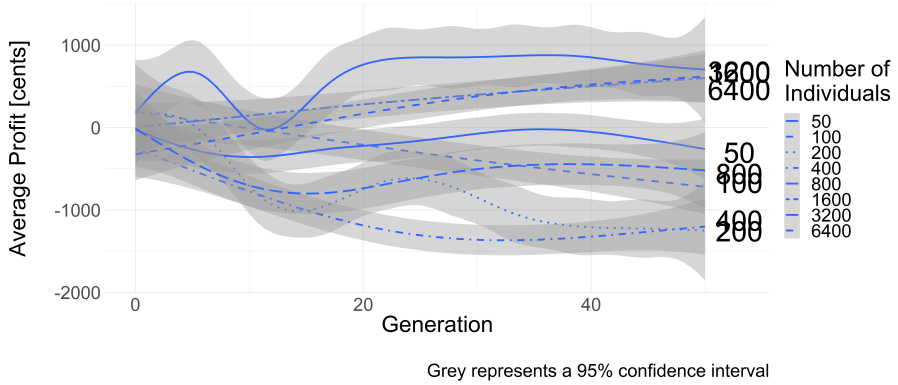



**Figure 5.59:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the GFI dataset and various population sizes.

Figure 5.62 shows that the trading rules evolved with a population size of 1600, 3200, and 6400 for the INVNED dataset returned the largest $Sample_{out}$ results within 50 generations. The graph also shows that the trading rules evolved using a population size of 50 returned the lowest profit. The p-value results presented in Figure 5.50d fail to show a significant difference between the $Sample_{out}$ results returned by the trading rules evolved using a population size of 400, 800, 1600, 3200, and 6400. The critical different plot presented in Figure 5.60 confirms the p-value results. The critical difference plot also shows the similarity of the $Sample_{out}$ results returned by the trading rules evolved using a population size of 1600, 3200 and 6400. The Friedman pairwise test found with a confidence level greater than 73% that the $Sample_{out}$ results returned by the trading rules evolved using a population size of 1600, 3200 and 6400 are not significantly different.

Analysis of the $Sample_{in}$ results presented in Figure 5.61 shows that the trading rules evolved using the INVNED dataset and a population size of 800, 1600, 3200, or 6400 individuals returned an average best individual profit greater than 30000 within 15 generations. Based on these results, a population size of 1600, 3200, or 6400 is preferred when evolving trading rules for the INVNED dataset.

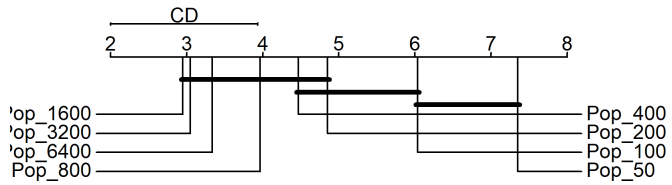
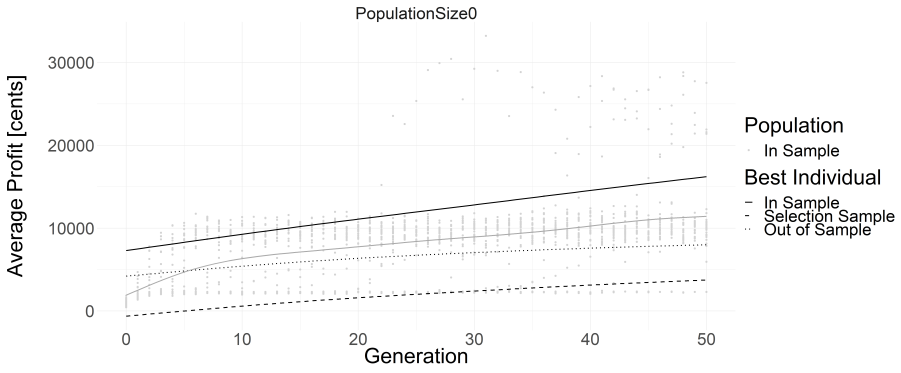
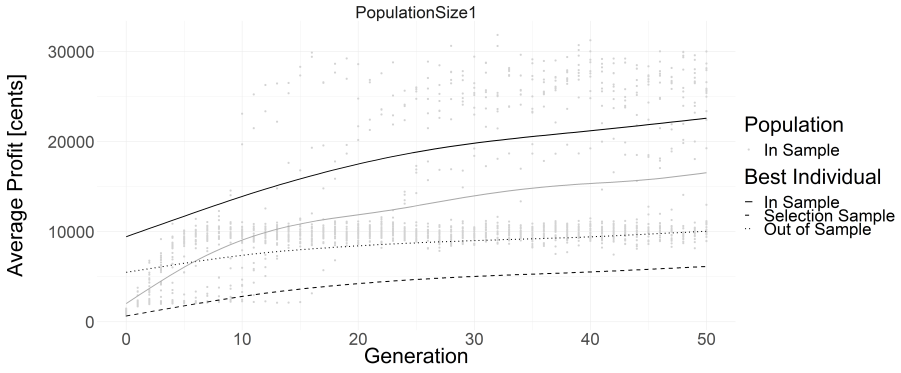


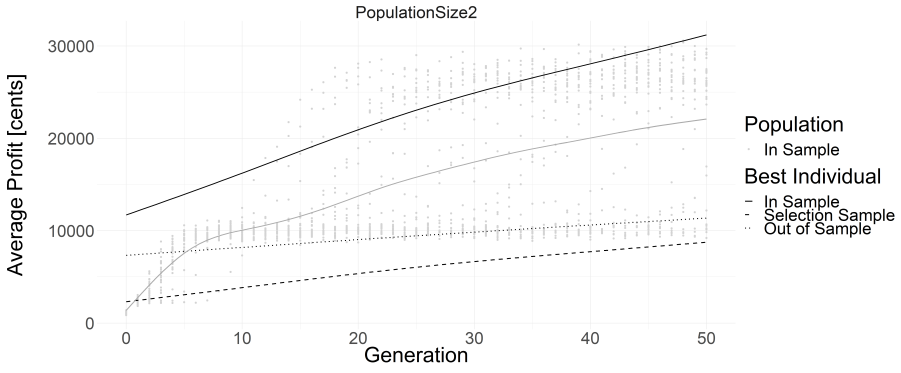

**Figure 5.60:** Critical difference plot of INVNED $Sample_{out}$ results using various population sizes.



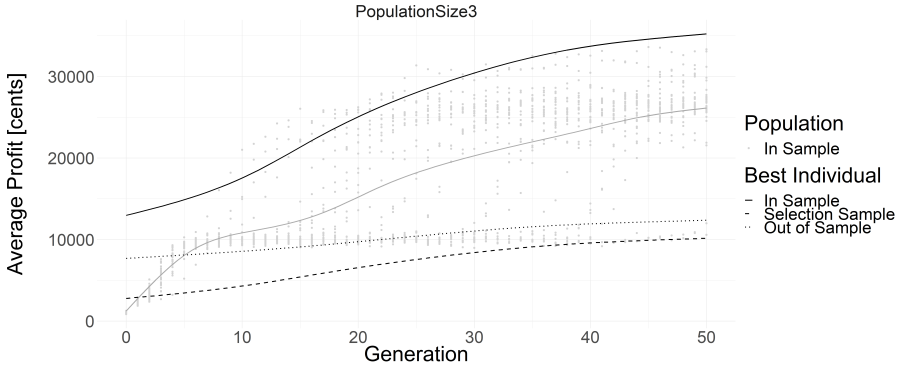

**(a)** Population size set at 50 individuals




**(b)** Population size set at 100 individuals

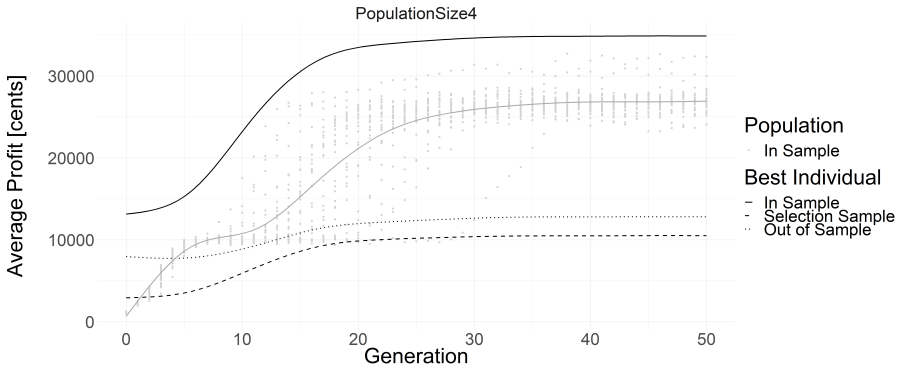



**(c)** Population size set at 200 individuals

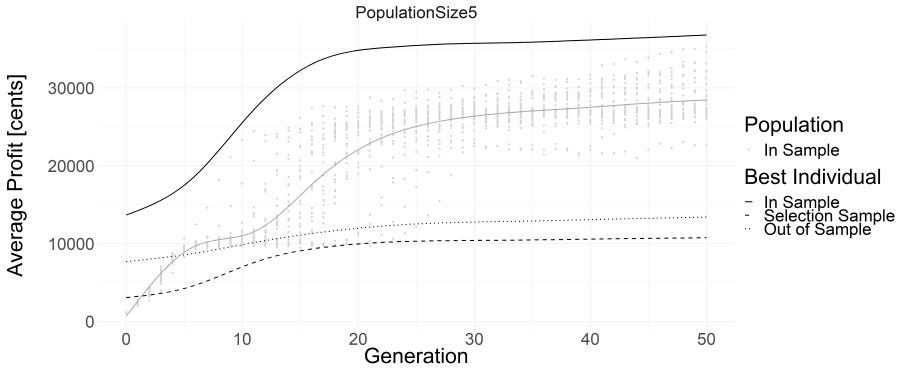



**(d)** Population size set at 400 individuals

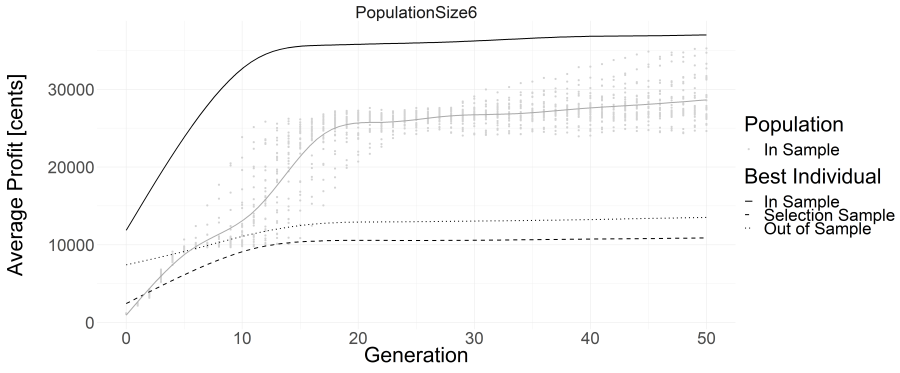
**Figure 5.61:** Scatter plot plotting returned profit using INVNED dataset and various population sizes.



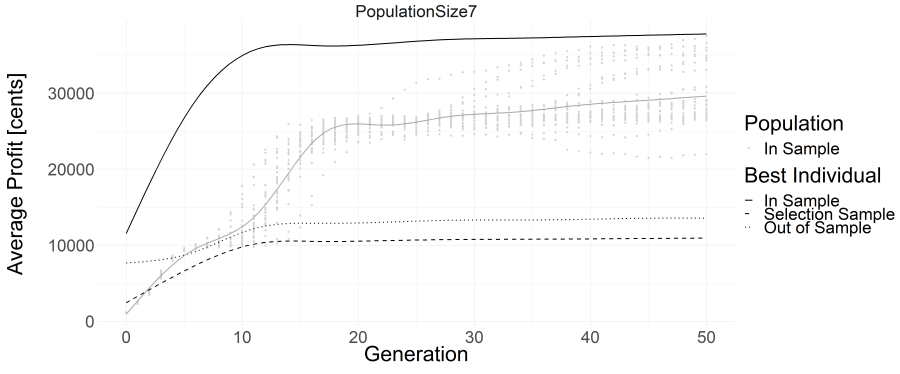

**(e)** Population size set at 800 individuals




**(f)** Population size set at 1600 individuals




**(g)** Population size set at 3200 individuals




**(h)** Population size set at 6400 individuals

**Figure 5.61:** Scatter plot plotting returned profit using INVNED dataset and various population sizes (Cont.).

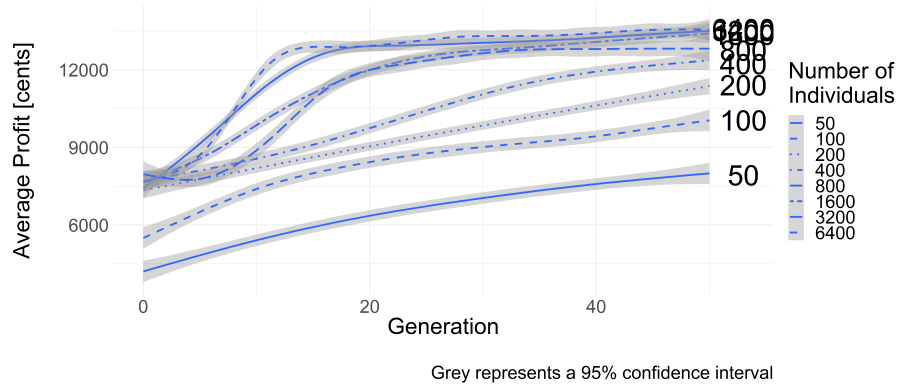



**Figure 5.62:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the INVNED dataset and various population sizes.

The profit results returned by the trading evolved evolved for the INVREM dataset are presented in Figure 5.64, and are similar to the trading results for the INVNED dataset presented in Figure 5.61. The best performing $Sample_{out}$ trading rules presented in Figure 5.65 were evolved using a population size of 6400, followed by 3200 and 1600. The critical difference graph in Figure 5.63 shows no critical difference between the $Sample_{out}$ results returned by the trading rules evolved using a population size of 1600, 3200, and 6400. Figure 5.64 shows that only the trading rules evolved with a population size of 1600, 3200, and 6400 returned an average best individual $Sample_{out}$ profit greater than 2000 within 10 generations. Based on these observations, a population size of 1600, 3200, or 6400 is preferred when evolving trading rules for the INVNED dataset.

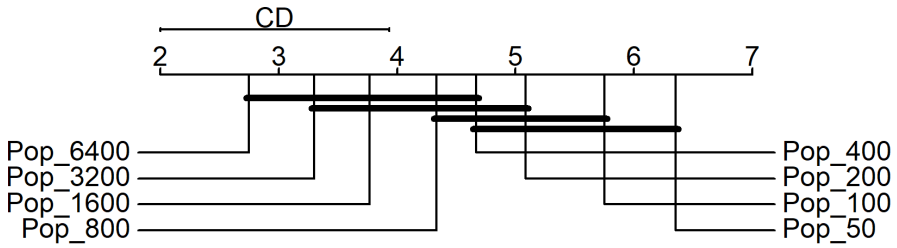
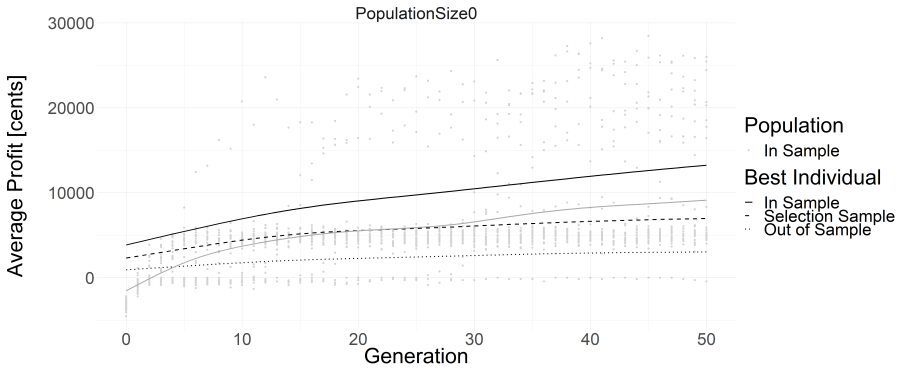
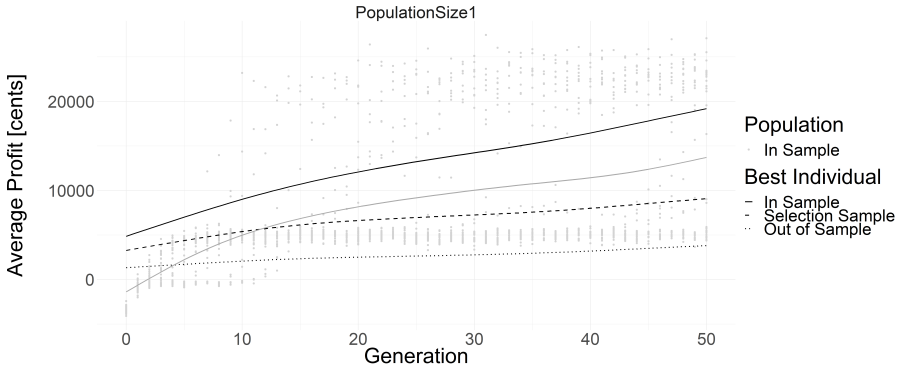


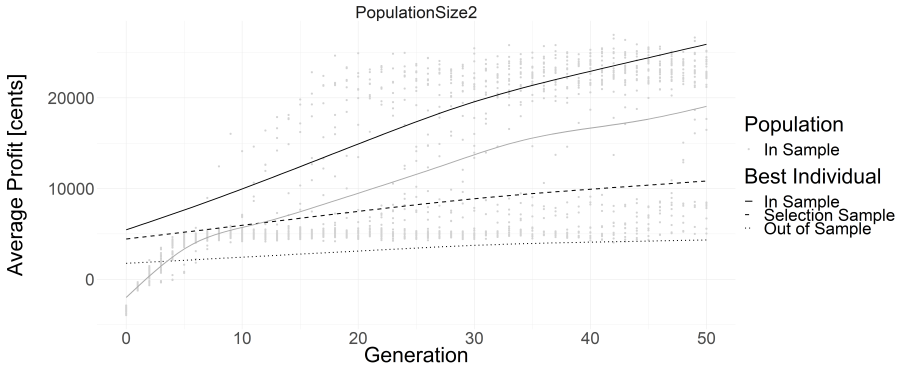

**Figure 5.63:** Critical difference plot of INVREM $Sample_{out}$ results using various population sizes.



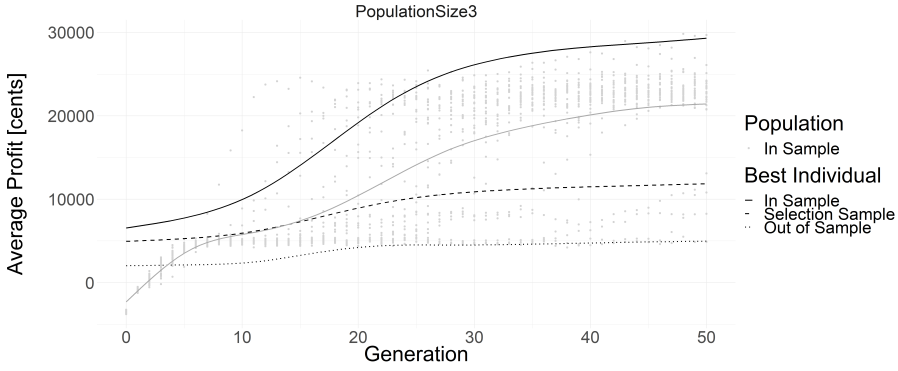

**(a)** Population size set at 50 individuals




**(b)** Population size set at 100 individuals

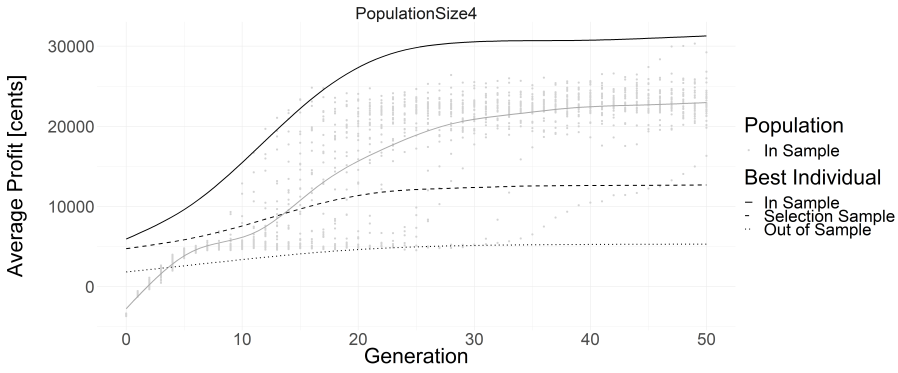



**(c)** Population size set at 200 individuals

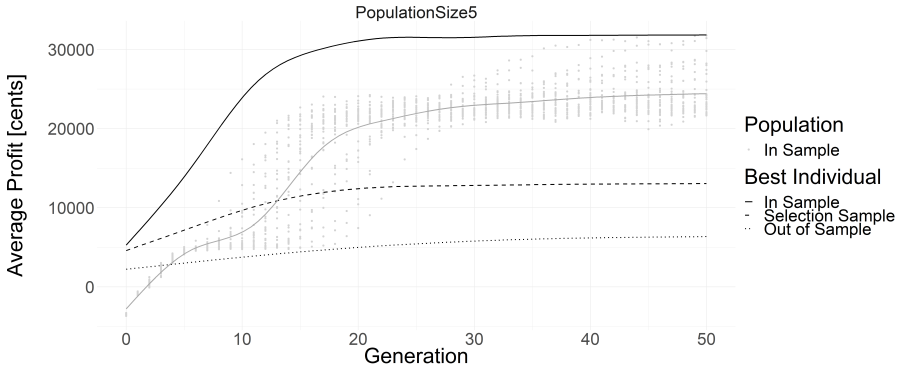



**(d)** Population size set at 400 individuals

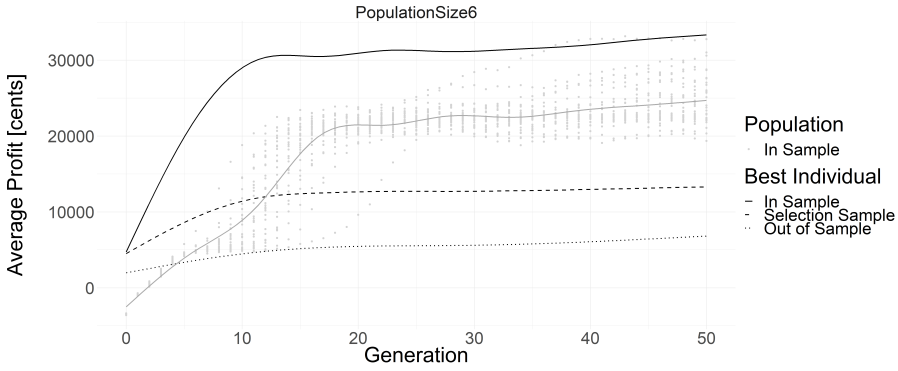
**Figure 5.64:** Scatter plot plotting returned profit using INVREM dataset and various population sizes.



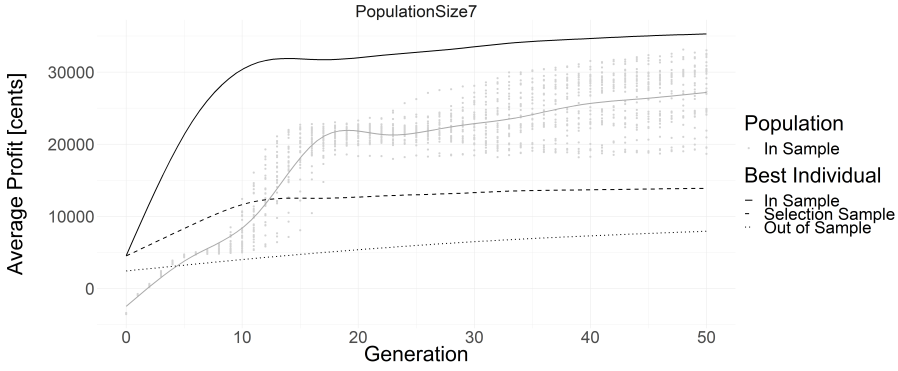

**(e)** Population size set at 800 individuals




**(f)** Population size set at 1600 individuals




**(g)** Population size set at 3200 individuals




**(h)** Population size set at 6400 individuals

**Figure 5.64:** Scatter plot plotting returned profit using INVREM dataset and various population sizes (Cont.).

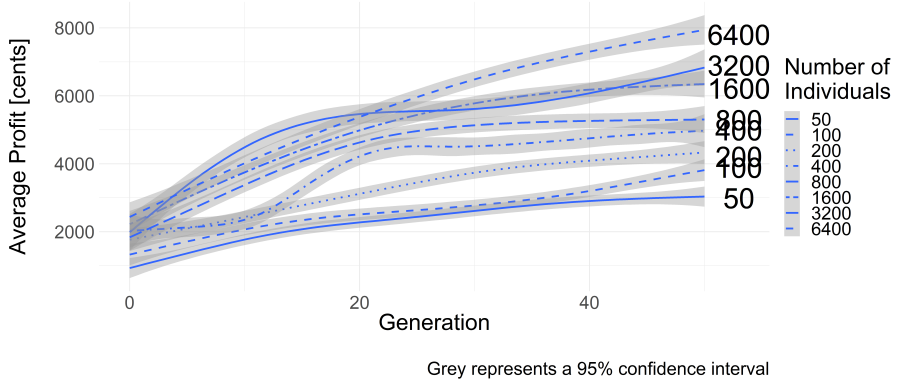



**Figure 5.65:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the INVREM dataset and various population sizes.

The results for trading rules evolved for the INVSBK dataset are presented in Figure 5.67. The best performing $Sample_{out}$ trading rules presented in Figure 5.68 were evolved using a population size of 3200, and 6400, followed by 800 and 1600. Figure 5.50f shows no significant difference was found between the $Sample_{out}$ results returned by the trading rules evolved using a population size of 400, 800, 1600, 2400, 3200, and 6400. The critical difference graph in Figure 5.66 shows no critical difference between the $Sample_{out}$ results returned by the trading rules evolved using a population size of 400, 800, 1600, 3200, and 6400. Figure 5.67 shows that only the trading rules evolved with a population size of 1600, 3200, and 6400 returned an average best individual $Sample_{out}$ profit greater than 2500 within 15 generations. Therefore, based on these observations, a population size of 1600, 3200, or 6400 is preferred when evolving trading rules for the INVSBK dataset.

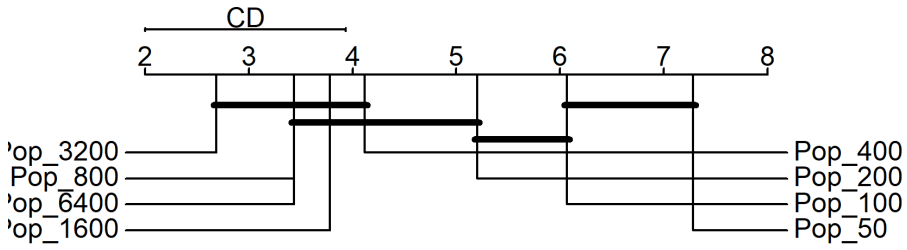
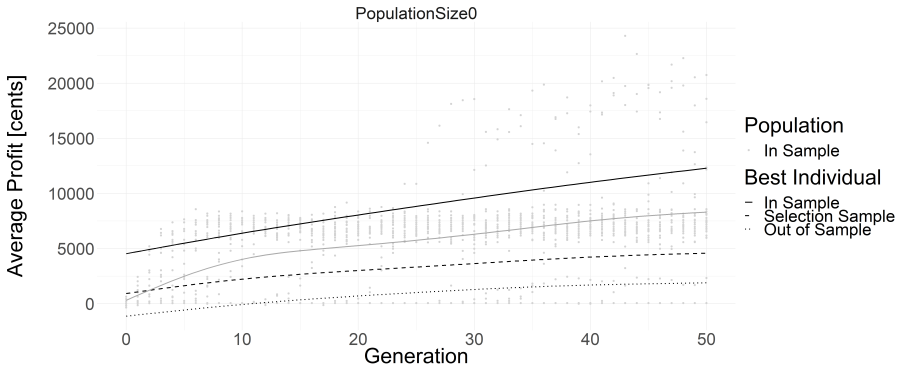
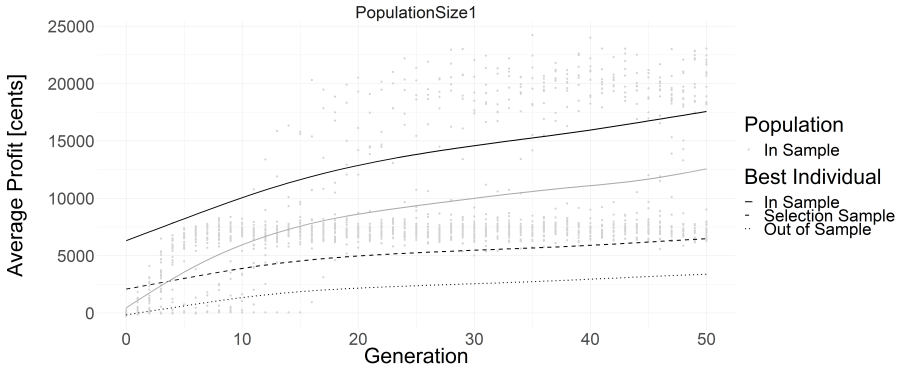


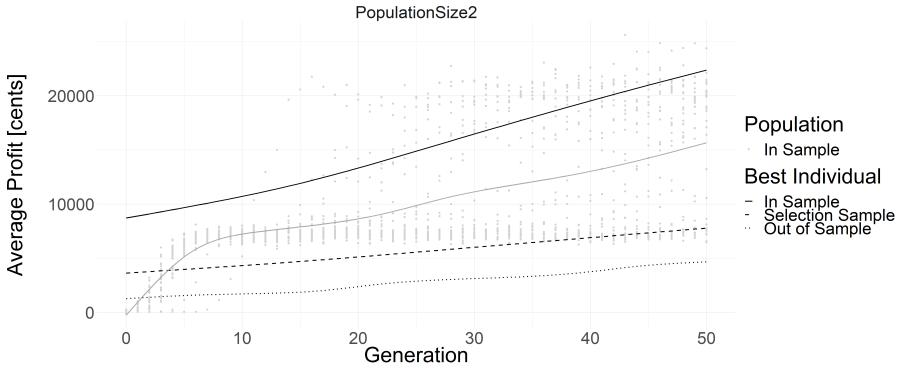

**Figure 5.66:** Critical difference plot of INVSBK $Sample_{out}$ results using various population sizes.



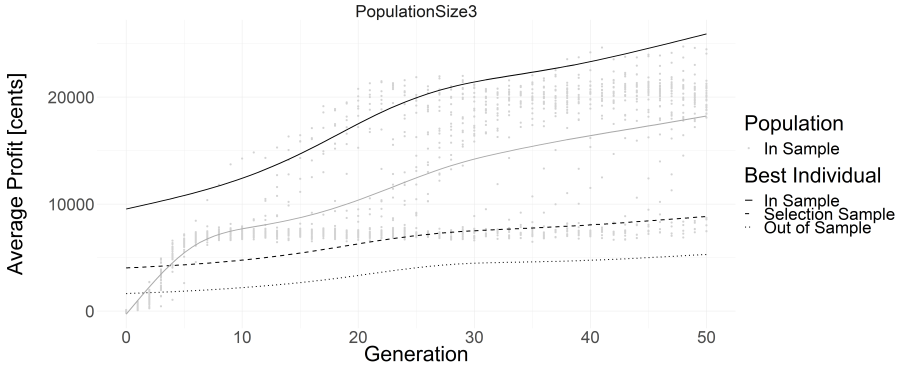

**(a)** Population size set at 50 individuals




**(b)** Population size set at 100 individuals

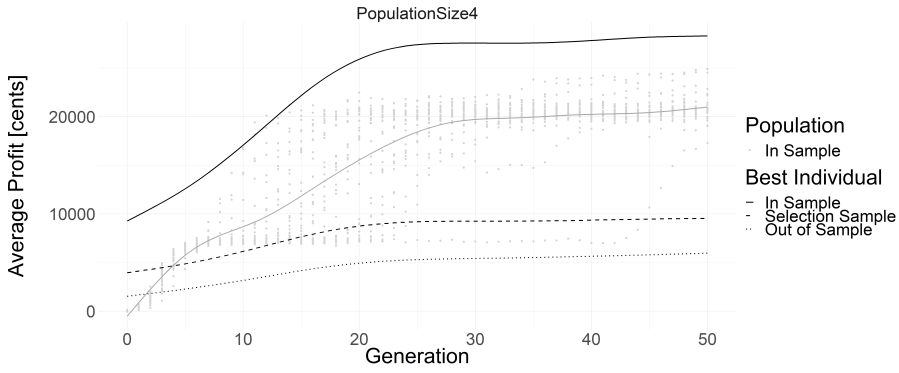



**(c)** Population size set at 200 individuals

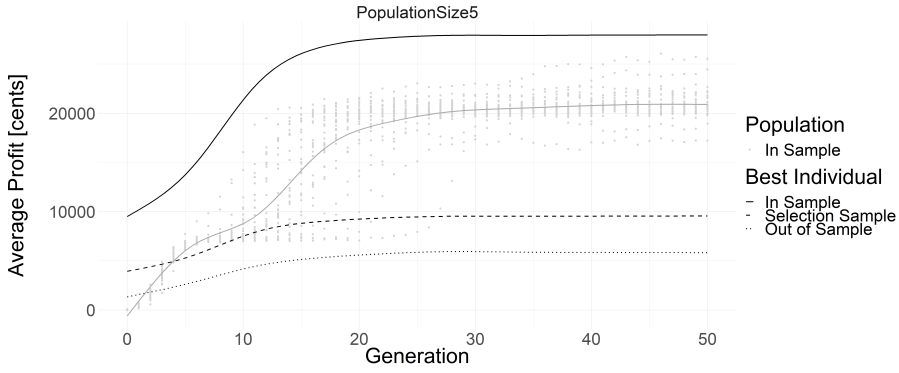



**(d)** Population size set at 400 individuals

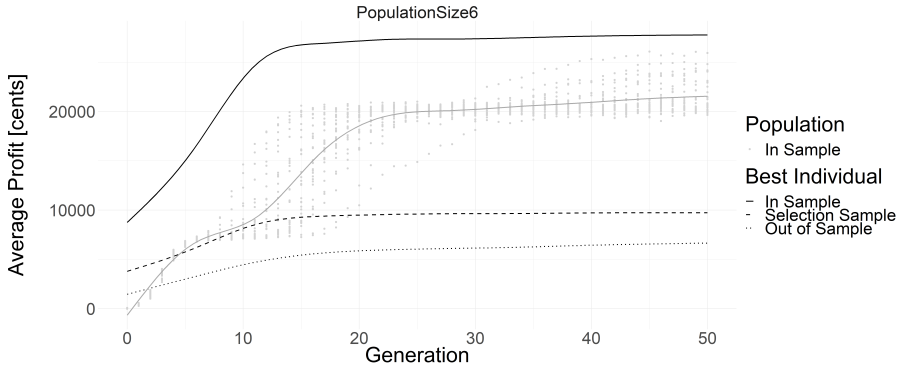
**Figure 5.67:** Scatter plot plotting returned profit using INVSBK dataset and various population sizes.



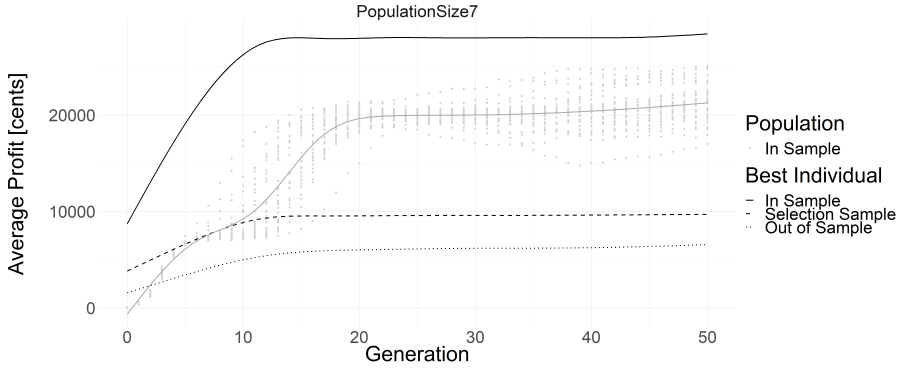

**(e)** Population size set at 800 individuals




**(f)** Population size set at 1600 individuals




**(g)** Population size set at 3200 individuals




**(h)** Population size set at 6400 individuals

**Figure 5.67:** Scatter plot plotting returned profit using INVSBK dataset and various population sizes (Cont.).

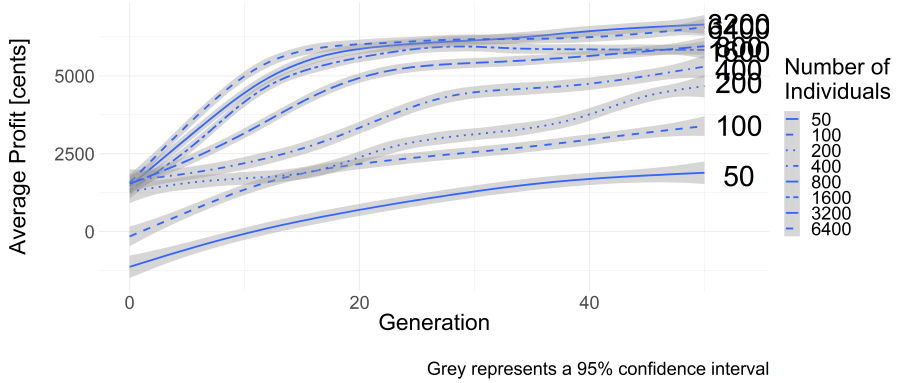



**Figure 5.68:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the INVSBK dataset and various population sizes.

The profit for the trading rules evolved using the NED dataset are presented in Figure 5.71. The profit results show that the trading rules evolved using a population size of 50 returned the largest $Sample_{out}$. The critical difference plot presented in Figure 5.69 shows that the $Sample_{out}$ results returned by the trading rules evolved using the NED dataset and a population size of 50 are critically different from the results returned by the trading rules evolved using a population size of 100, 200, 400, 800, 1600, and 3200. Figure 5.70 shows that the $Sample_{in}$ and $Sample_{sel}$ profit results returned by the trading rules evolved using a population size of 50 were the lowest. These observations show that a low population size of 50 is preferred when evolving trading rules for the NED dataset and contradicts the results observed in the previous datasets.

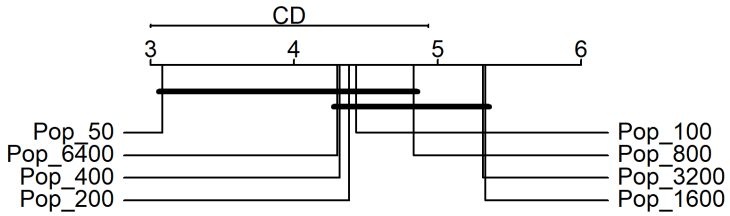
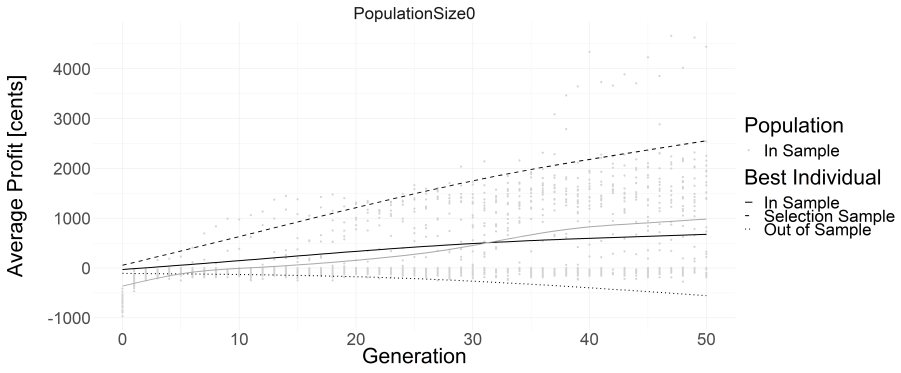
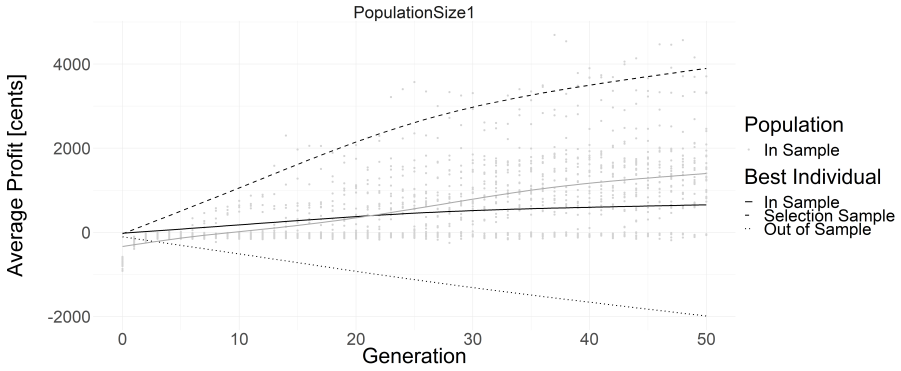


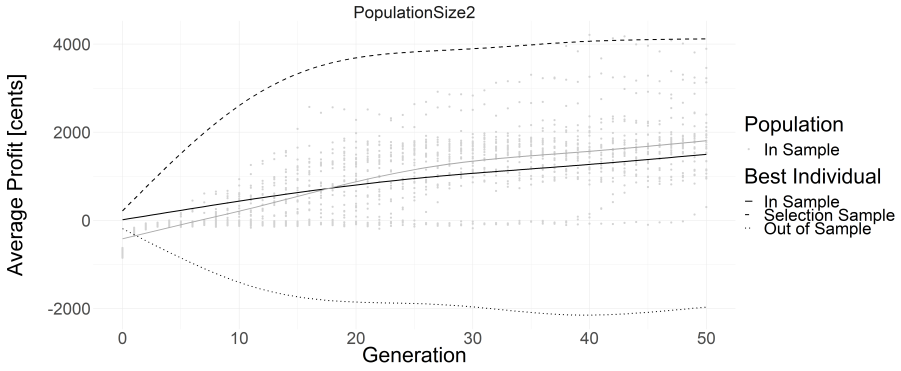

**Figure 5.69:** Critical difference plot of NED $Sample_{out}$ results using various population sizes.



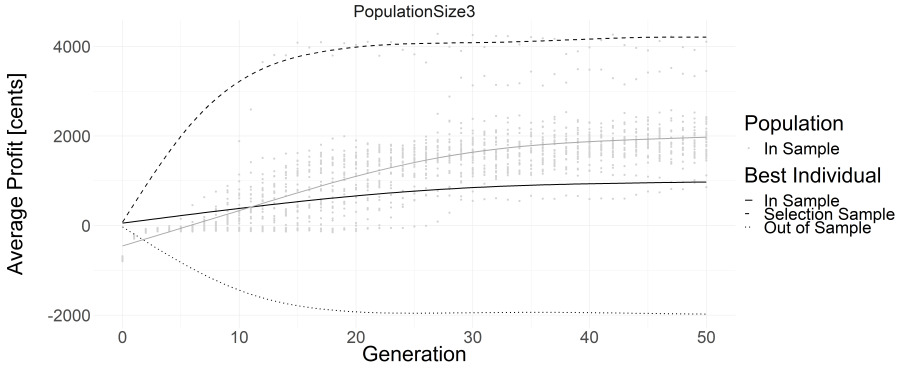

**(a)** Population size set at 50 individuals




**(b)** Population size set at 100 individuals

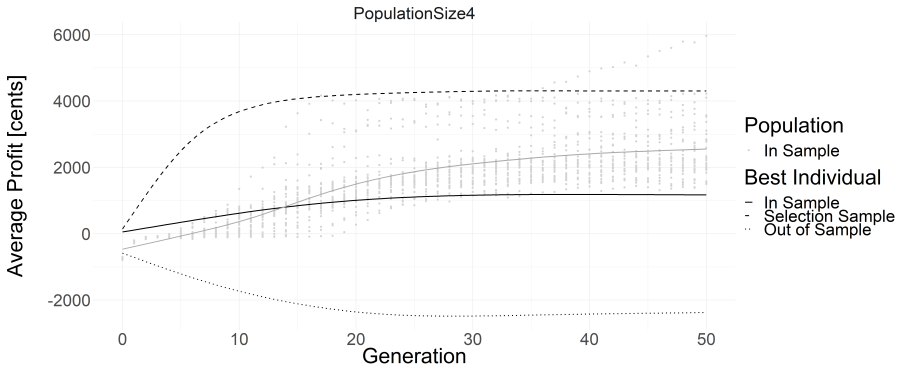



**(c)** Population size set at 200 individuals

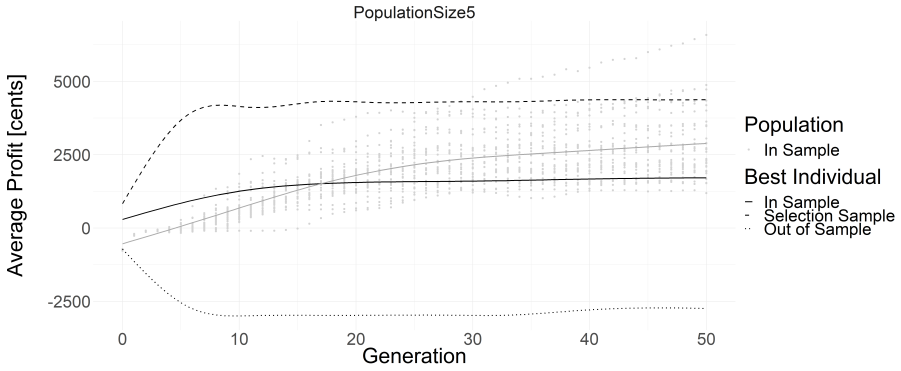



**(d)** Population size set at 400 individuals

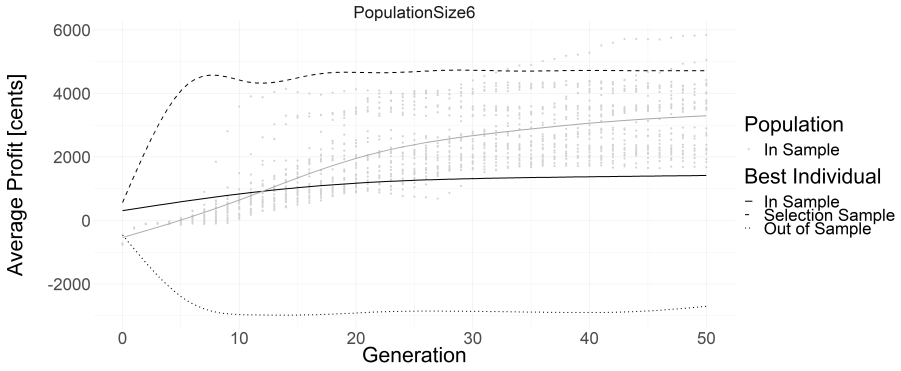
**Figure 5.70:** Scatter plot plotting returned profit using NED dataset and various population sizes.



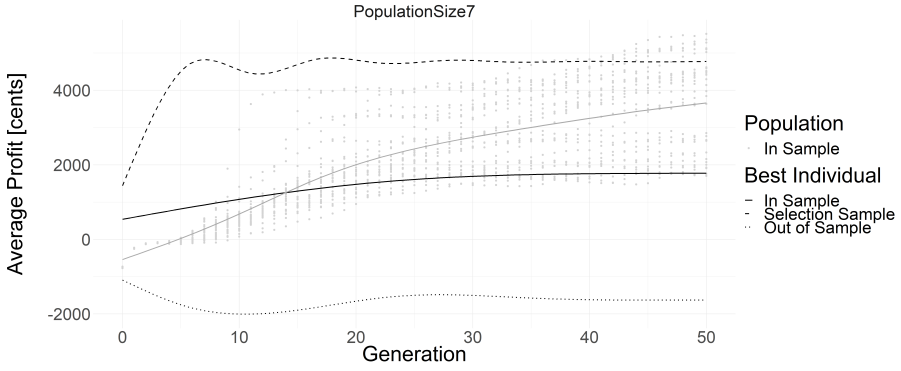

**(e)** Population size set at 800 individuals




**(f)** Population size set at 1600 individuals




**(g)** Population size set at 3200 individuals




**(h)** Population size set at 6400 individuals

**Figure 5.70:** Scatter plot plotting returned profit using NED dataset and various population sizes (Cont.).

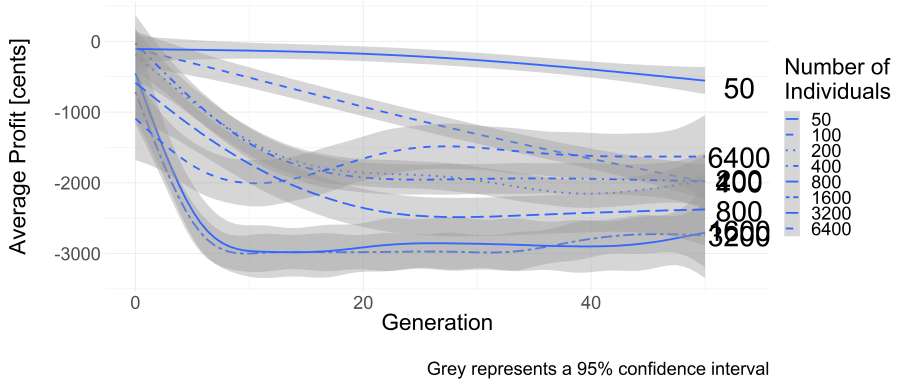



**Figure 5.71:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the NED dataset and various population sizes.

The $Sample_{out}$ profit returned by the trading rules evolved for the SAB dataset is presented in Figure 5.74 and shows that the trading rules evolved with a population size of 3200 and 6400 returned the largest $Sample_{out}$ profit. The graph also shows that the trading rules evolved using the SAB dataset and a population size of 1600 resulted in the lowest $Sample_{out}$ profit. The difference in average returns equates to less than 1.8% of the share price. The critical difference plot presented in Figure 5.72 shows a critical difference between the results returned by the trading rules evolved using a population size of 3200 and 6400 and the results returned by the trading rules evolved using a population size of 1600. The $Sample_{in}$ results presented in Figure 5.73 shows that the trading rules evolved using a population size of 3200 and 6400 returned the largest $Sample_{in}$ profit. Based on these observations, a population size of 3200, or 6400 was preferred when evolving trading rules for the SAB dataset.

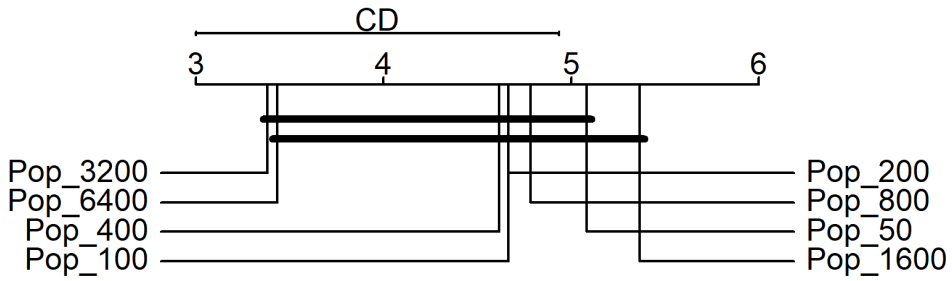
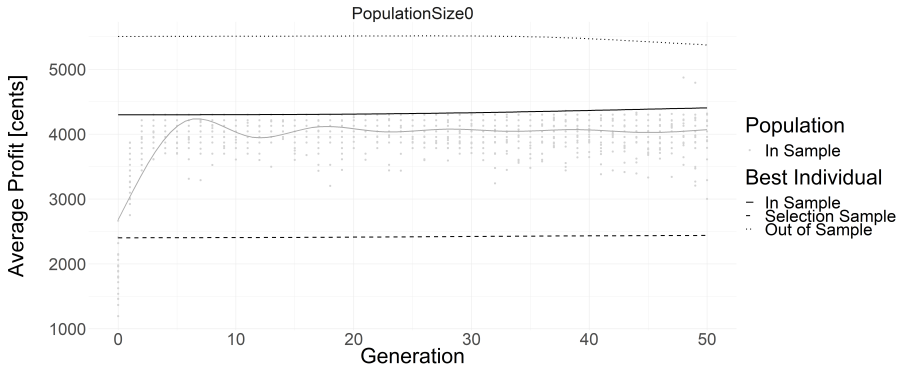
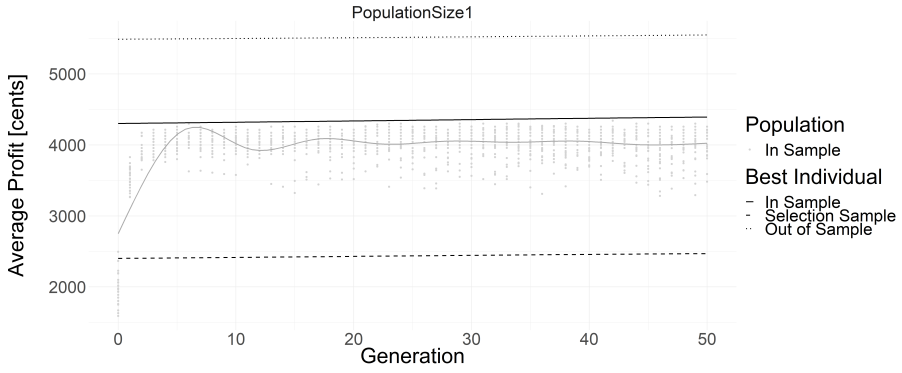


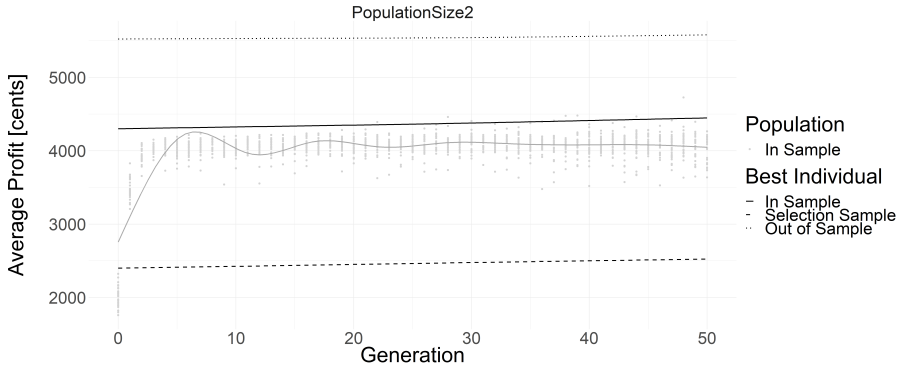

**Figure 5.72:** Critical difference plot of SAB $Sample_{out}$ results using various population sizes.



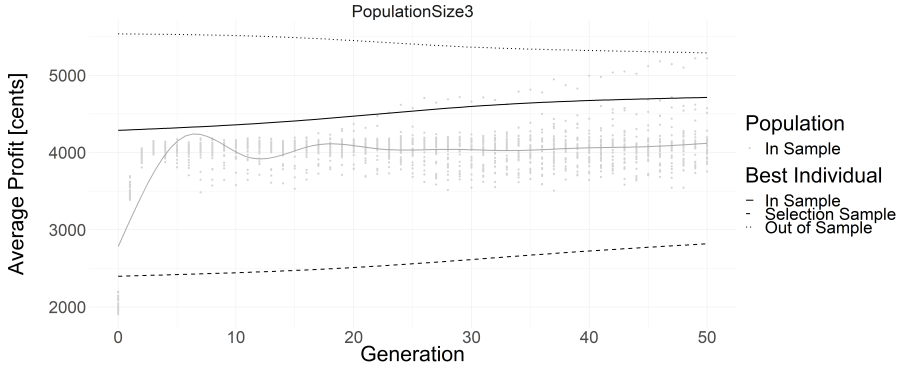

**(a)** Population size set at 50 individuals




**(b)** Population size set at 100 individuals

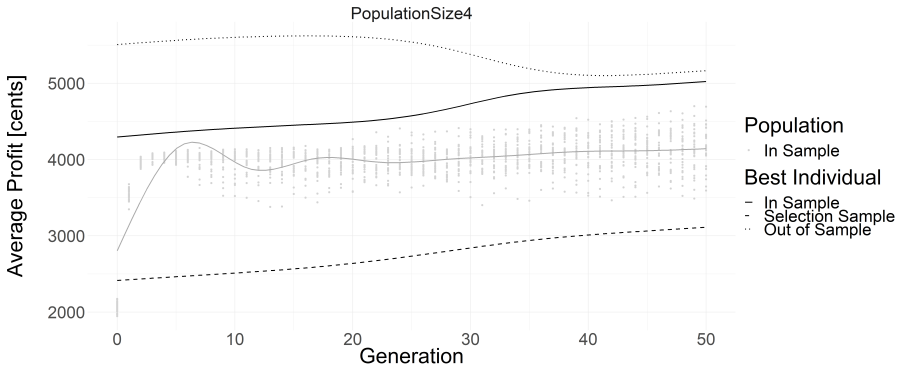



**(c)** Population size set at 200 individuals

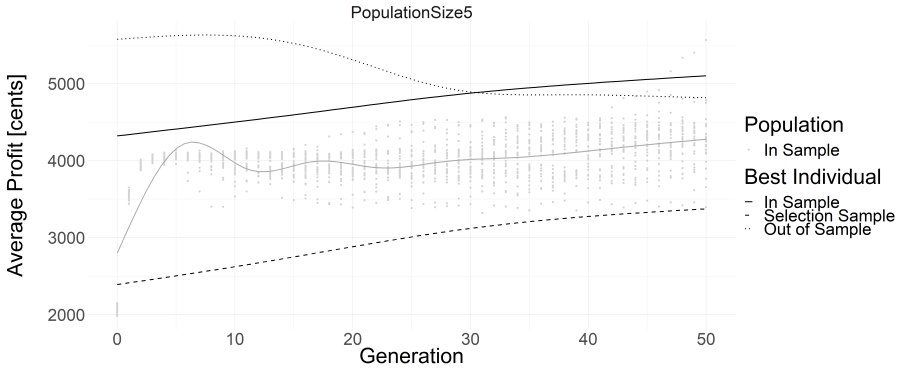



**(d)** Population size set at 400 individuals

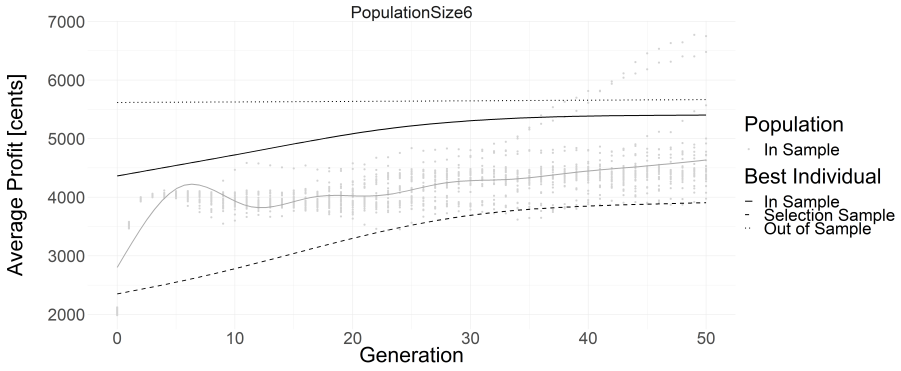
**Figure 5.73:** Scatter plot plotting returned profit using SAB dataset and various population sizes.



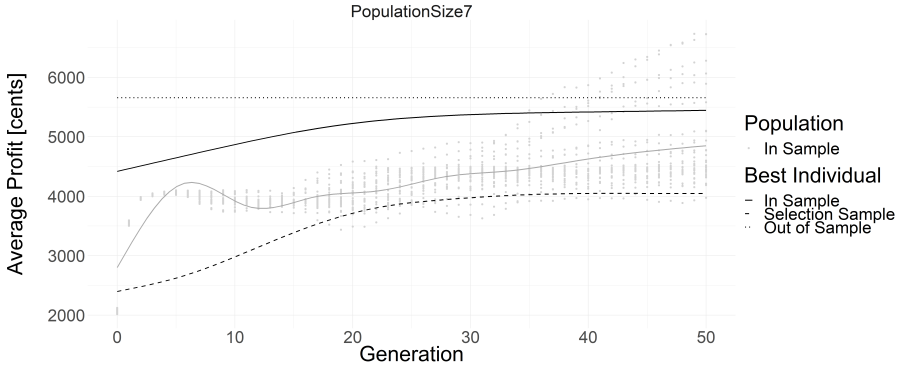

**(e)** Population size set at 800 individuals




**(f)** Population size set at 1600 individuals




**(g)** Population size set at 3200 individuals




**(h)** Population size set at 6400 individuals

**Figure 5.73:** Scatter plot plotting returned profit using SAB dataset and various population sizes (Cont.).

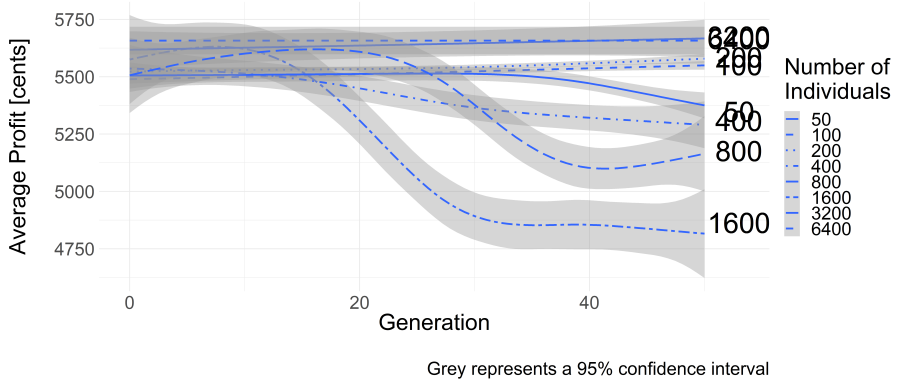



**Figure 5.74:** Average $Sample_{out}$ profit returned by the best individual within a simulation evolved using the SAB dataset and various population sizes.

The results returned by the trading rules evolved using various populations sizes show that, generally larger population sizes of 1600, 3200, or 6400 evolve the best performing trading rules. The exception to this observation was the results returned by the trading rules evolved for the NED dataset, which shows that a small population size of 50 is preferred. Except for the trading rules evolved for the SAB dataset, no significant difference was found between the results returned by the trading rules evolved using population sizes of 1600, 3200, and 6400. Trading rules evolved for the SAB dataset show no significant difference between the results returned by the trading rules evolved using a population size of 3200 or 6400. A significant difference between the results returned by the trading rules evolved using a population size of 1600 and 3200 was found. However, the average returned profit was less than 1.8%.

The population size has a computational expense. Selecting the lowest population size is preferred. Therefore, based on the observations in this section, a population size of 1600 was used for the remainder of this study.

### 5.5.3 Maximum Generation Sensitivity Analysis

Each generation allows the evolutionary process to optimise the genetic material within the population. Allen and Karjalainen [43] terminated evolution when the number of generations reached 50 or if there was no improvement within 25 generations. The purpose of this section is to determine the maximum number of generations that must pass before the trading rules stop improving significantly.

A simulation was run for each share presented in Table 5.2. The simulation used the configuration defined in Table 5.3. Rank selection was used as the selection operator, and the maximum number of

generations was set to 1500. A maximum of 1500 generations was selected as 1500 generations took approximately a week to compute per share code.

At set intervals the best individual for a given generation was run using the $Sample_{out}$ dataset. The result was recorded as a sample profit. The first sample profit was taken at 20 generations and then next sample at 40. Each subsequent profit sample interval was doubled, until the interval was 1280. A total of seven profit samples were taken for each share. The samples were compared using the Iman and Davenport test [142] and the test results are presented in Table 5.23.

The p-values in Table 5.23 show that a significant difference between the profit samples drawn from the trading rules evolved using the inverted shares INVNED, INVREM, and INVSBK exists. A post-hoc analysis was done on the sample results returned by the trading rules evolved for the INVNED, INVREM, and INVSBK datasets. The post-hoc results are presented in Figure 5.75. Post-hoc analysis shows that a significant difference in the profit samples returned at various intervals exists.

Figure 5.75a shows three groups of INVNED $Sample_{out}$ profit results. Each group is shown to be significantly different from any other group. The first group represents the profit results recorded at interval 20. The second group represents the $Sample_{out}$ profit results recorded intervals 40, 80, and 160. The third group represents the $Sample_{out}$ profit results recorded at intervals 320, 640, and 1280. The three groups are better represented in the critical difference plot presented in Figure 5.76a.

The scatter plot in Figure 5.77a plots the average INVNED $Sample_{in}$ profit returned by all the trading rules per simulation run per generation as dots. The lines represent the average profit returned by the best trading rules for each of the $Sample_{in}$, $Sample_{sel}$, and $Sample_{out}$ datasets. A comparison of the interval groups to the $Sample_{out}$ profit recorded in Figure 5.77a shows that the first group performed worse than the second and third group. The graph shows that the third group is the most stable, but at around generation 125, the $Sample_{out}$ profit plotted is greater than the profit plotted for the generations 80 and 160. The $Sample_{out}$ profit plotted at generation 125 is in-line with the profit plotted for the third interval group at generations 320, 640, and 1280.

Overall the scatter plot shows a rapid increase in profit across all the INVNED sample datasets and at around generation 150 the recorded profit begins to drop. The downward profit trend continues until generation 300 where it increases and stabilises at around generation 500. The $Sample_{in}$ and $Sample_{out}$ profit peaks at generation 125 and again at generation 500. Therefore, based on these observations, the maximum number of generations set by Allen and Karjalainen [43] is too low when trading INVNED. The number of generations should either be capped at 125 generations or be allowed to continue past 500 generations. An increase in the number of generations increases the computation time. A smaller number of generations is preferred. Therefore, based on the observations, a maximum number of 125

generations is preferred.

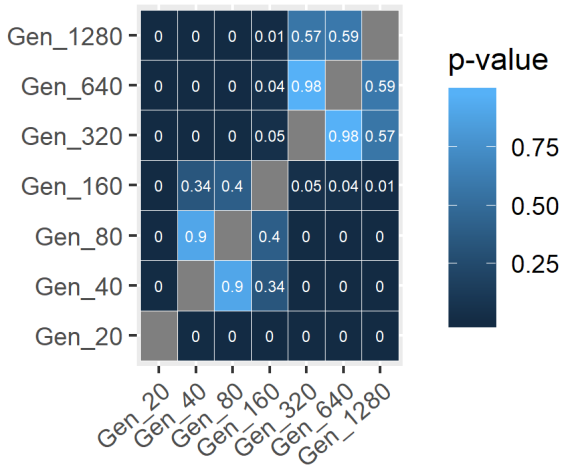
A pairwise comparison of the $Sample_{out}$ profits returned by the trading rules evolved for the INVREM profits recorded at each interval is presented in Figure 5.75b. The p-values show two significantly different groups. The first group contains the $Sample_{out}$ profits recorded at intervals 20, 40, 80, and 160. The second group contains the $Sample_{out}$ profits recorded at intervals 320, 640, and 1280. The critical difference plot presented in Figure 5.76b confirms that the result groupings are critically different. The scatter plot in Figure 5.77b is similar to that of Figure 5.77a. Profit increases rapidly until generation 125, then declines until generation 310, increasing again until generation 375, and stabilising at around 500 generations.

The results recorded for the trading rules evolved for the INVSBK dataset are similar to the results recoded for the trading rules evolved for the INVNED and the INVREM datasets. Therefore, based on the observations, a maximum number of generations is set at 125 for the remainder of this study.

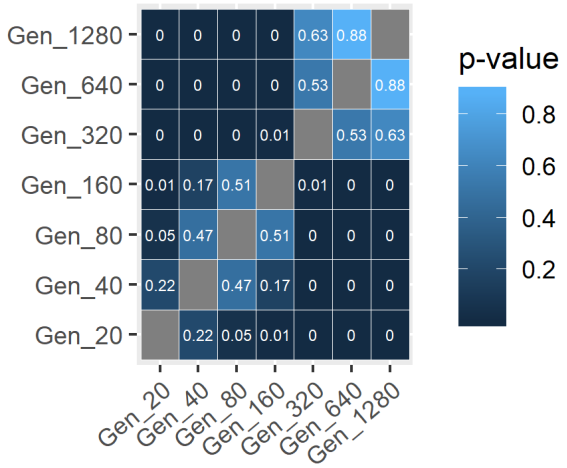
**Table 5.23:** p-value results using Iman and Davenport test with Finner's correction.

| | $Sample_{in}$ | | $Sample_{sel}$ | | $Sample_{out}$ | |
|---|---|---|---|---|---|---|
| **Share Code** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** |
| **agl** | 0.253419 | 0.306021 | 0.253419 | 0.328690 | 0.549709 | 0.908698 |
| **alsi** | **0.029435** | **0.043826** | 0.119918 | 0.174371 | 0.999776 | 0.999877 |
| **bil** | **0.005263** | **0.008756** | **0.007969** | **0.013246** | 0.331166 | 0.778719 |
| **gfi** | 0.855114 | 0.873789 | 0.908160 | 0.922560 | 0.966144 | 0.990116 |
| **imp** | 0.999938 | 0.999938 | 0.999938 | 0.999938 | 0.999938 | 0.999938 |
| **inv-ned** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **inv-rem** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **inv-sbk** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **lon** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | 0.764884 | 0.955048 |
| **ned** | 0.221408 | 0.289136 | 0.391329 | 0.460252 | 0.992314 | 0.997724 |
| **rch** | 0.332543 | 0.372792 | 0.389405 | 0.460252 | 0.997573 | 0.999039 |
| **rem** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | 0.934164 | 0.983107 |
| **sab** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | 0.628701 | 0.915994 |
| **sbk** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | 0.881647 | 0.981710 |
| **sol** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | 0.905040 | 0.981710 |

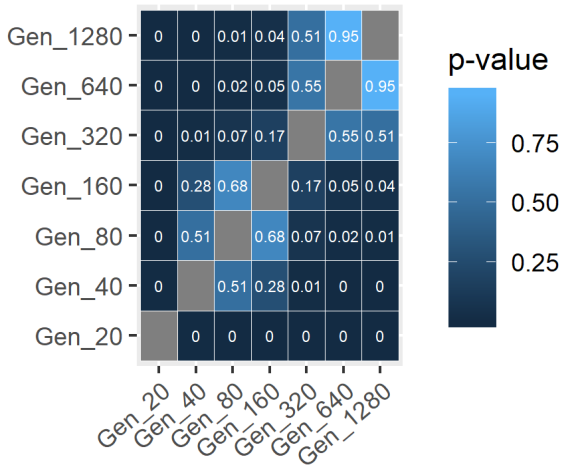
Note: **bold** values represent a significant difference.




**(a)** $Sample_{out}$ INV-NED

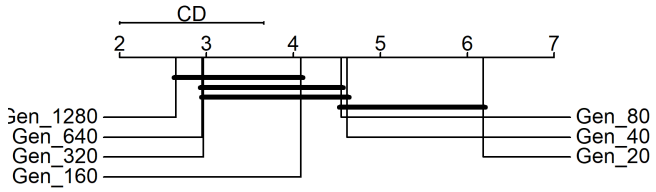



**(b)** $Sample_{out}$ INV-REM

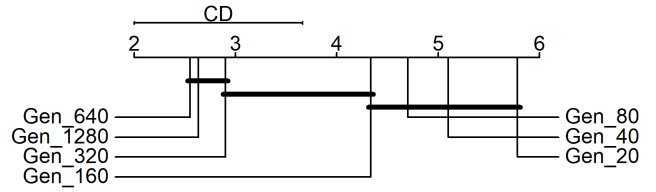



**(c)** $Sample_{out}$ INV-SBK

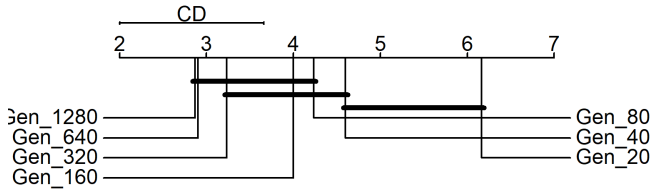
**Figure 5.75:** Friedman pairwise comparison of p-values for results (Cont.).




**(a)** $Sample_{out}$ INV-NED

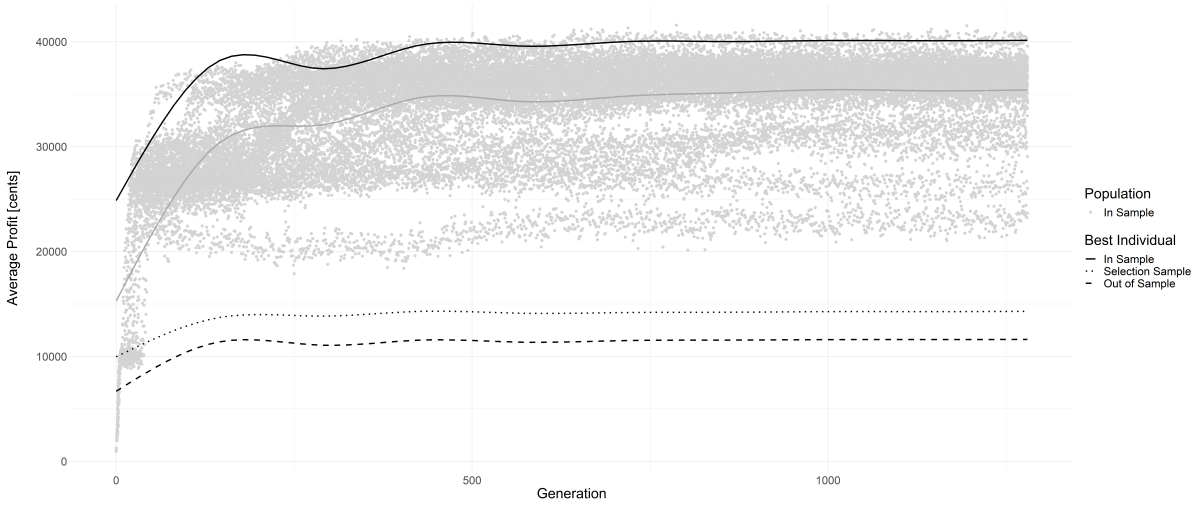



**(b)** $Sample_{out}$ INV-REM



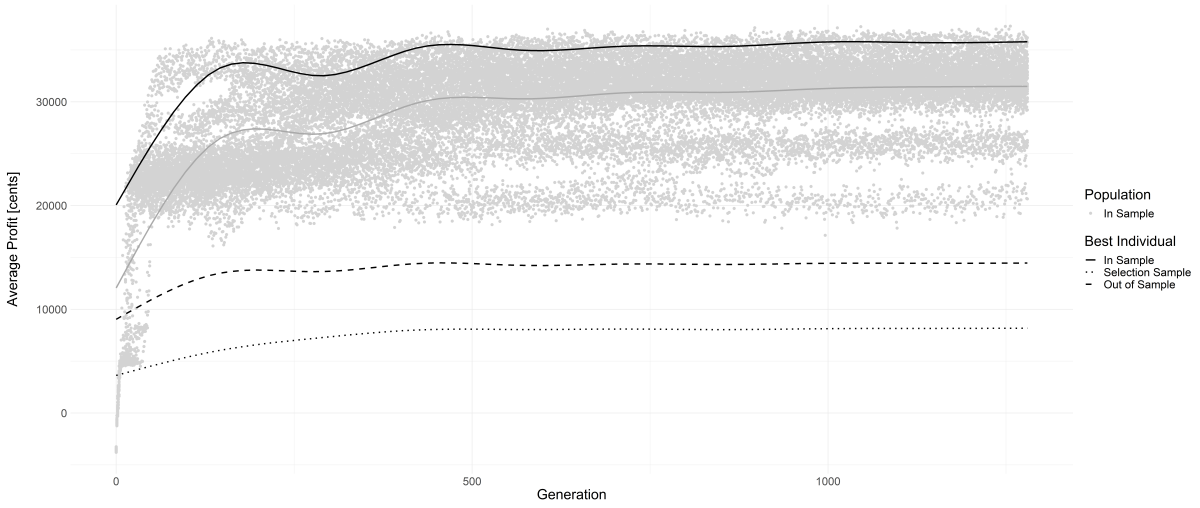

**(c)** $Sample_{out}$ INV-SBK

**Figure 5.76:** Critical difference plots for results (Cont.).

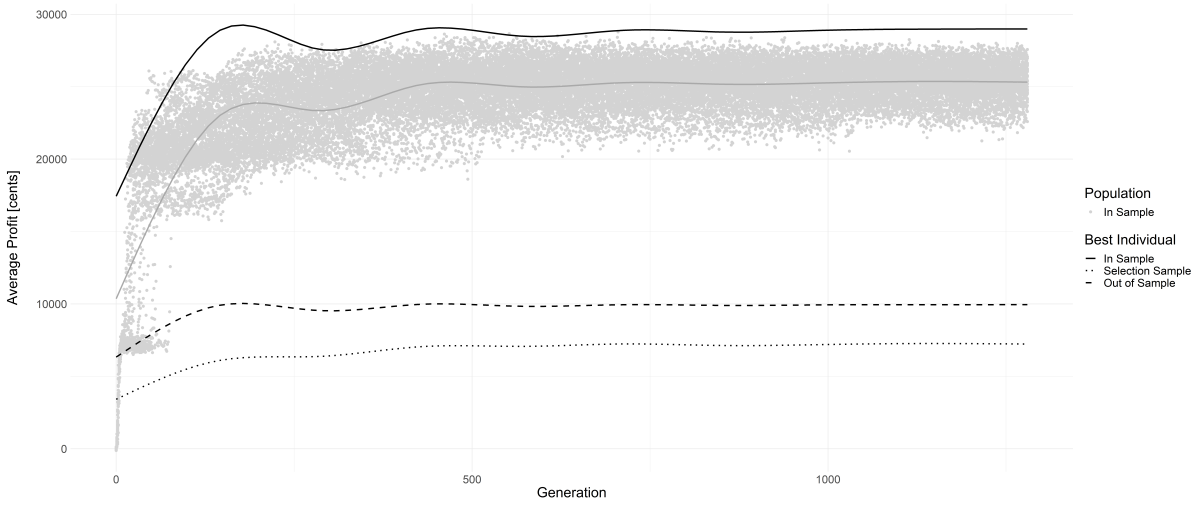



**(a)** INV-NED

**Figure 5.77:** Scatter plots of average profit. The dots represent the average $Sample_{in}$ profit for each simulation run. The $Sample_{in}$ plot shows that the level of exploration. The lines represent the mean best profit across $Sample_{in}$, $Sample_{sel}$, and $Sample_{out}$.




**(b)** INV-REM




**(c)** INV-SBK

**Figure 5.77:** Scatter plots of average profit. The dots represent the average $Sample_{in}$ profit for each simulation run. The $Sample_{in}$ plot shows that the level of exploration. The lines represent the mean best profit across $Sample_{in}$, $Sample_{sel}$, and $Sample_{out}$ (Cont.).

## 5.6 Summary

This chapter defined a single population GP by presenting the basic algorithm and grammar to evolve trading rules for stock market trading. The share datasets used in this thesis were discussed. The empirical process used in this thesis to evaluate the results was presented.

This chapter evaluated various fitness functions and concluded that the compounded excess return fitness function implemented by Allen and Karjalainen [43] is the preferred fitness function for this thesis. This chapter studied the effects that selection strategies have on the performance of evolved trading rules and concluded that rank selection [103, 126] is the most appropriate selection strategy for this thesis. The chapter ended with a parameter sensitivity analysis to conclude that the parameters defined by Allen and Karjalainen [43] will be used in the chapters that follow, except for the following parameter values determined by the parameter sensitivity analysis:

- Mutation probability of 35%;
- Population size of 1ǵ00 individuals;
- A maximum of 125 generations.

# Chapter 6

# Co-evolved Genetic Program For Trading Rules

*It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change.*

*- Charles Darwin (English Naturalist)*

The purpose of this chapter is to describe the co-evolved GP implementations used in the empirical study that follows. The chapter begins with a short description of co-evolution in Section 6.1. Section 6.1 comprises of two sub sections which describe in detail co-operative co-evolution and competitive co-evolution. This chapter concludes with a summary in Section 6.2.

## 6.1 Co-evolved Genetic Program for Stock Market Trading

Chapter 5 focused on optimising the GP implemented by Allen and Karjalainen [43]. This chapter introduces two different multi-population versions of the single population GP presented previously.

Recall from Section 3.4, that a co-evolved GP is a multi-population GP where either populations work together to evolve a solution, or compete to evolve a solution. Populations that work together are referred to as co-operative, while populations that work against one another are called competitive. This thesis compares trading rules evolved using both competitive co-evolution and co-operative co-evolution to trading rules evolved using the single population GP presented in the previous chapter.

The standard GP presented in Chapter 5 evolved a trading rule to return a boolean value of `true` or `false` based on the current market conditions. The boolean result was transcoded to buy, sell or hold depending on whether the trader had shares or not. Separating the trading rule into two questions, i.e. "If I have no shares, should I buy some?" and "If I have shares, should I sell them?", creates two distinct trading rules. These two rules, a buy rule and sell rule could be evolved independently. Co-operative co-evolution was used to evolve two different populations. Each population represented either the buy rule or sell rule. Together the populations form a trading rule. The implemented co-operative co-evolved GP is described in more detail in sub section 6.1.1.

The competitive co-evolved trader evolves two populations that compete against each other to become the best individual. The implemented competitive co-evolved GP is described in more detail later in sub section 6.1.2.

### 6.1.1 Co-operative Co-evolved Genetic Program

Co-operative co-evolution requires two or more populations that evolve together, each population benefits from the other through information sharing and reward sharing. Each population undergoes evolution independently. The single population evolutionary process defined in Algorithm 3 in Chapter 5 was divided into two parts to support co-operative co-evolution. The first part, presented in Algorithm 4, initialises multiple populations, iterates through the generations and returns the global best individuals from each population. The second part, illustrated in Algorithm 5 evolves each population indepen-

dently.

The key differences between the co-operative co-evolution algorithm presented here and the single population algorithm is that more than one population is evolved, a reward sharing is used to determine the best individuals, and multiple individuals are returned. The dual population approach of Seshadri [193] was applied to the optimised GP of Chapter 5, to implement a co-operative co-evolved GP.

Seshadri [193] used two populations to evolve a trading rule. One population evolved a buy rule and the other a sell rule. Two individuals were required to form a trading rule. One individual from the buy population, and one individual from the sell population. Using the sell population, Seshadri [193] sampled the best five individuals from the previous generation and five random individuals from the current population two form a sell reciprocal trading rule set. Each individual from the buy population was evaluated with each individual within the sell reciprocal trading rule set. A buy rule and a reciprocal sell rule combination were evaluated using the compounded excess return [43, 193] fitness function. Seshadri adjusted the fitness value using a penalty function proportional to the number of nodes within the individual. The fitness value of an individual from the buy population was set as the total fitness obtained across all evaluations. The process was repeated for the individuals within the sell population. Seshadri [193] used the S&P500 index to evolve the trading rules.

The co-operative co-evolution GP implemented in this thesis differs from the algorithm presented by Seshadri [193], in that a reward sharing system was used across all individuals. The reward sharing was based on the "bucket brigade" [144] credit scoring system. Each individual from the buy population was evaluated with each individual from the sell population, an approach followed by De Jong and Potter [79, 193], and Axelrod [51, 193]. The buy and sell rule combination was evaluated using a fitness function, to return a fitness value. The total reward of both the individuals used in the evaluation were incremented by the fitness value.

The process of buying and selling shares is illustrated in Algorithm 7. *EvaluateCombination* receives three parameters. The first parameter is an individual that represents a buying rule, the second is an individual that represents a selling rule. A buy or sell individual is defined by the same grammar and syntax as the optimised GP presented in Chapter 5 . The third parameter represents the dataset used for evaluation.

A trader has no shares on the first day of trading. The trader must execute the trading rule represented by an individual from the buy population. If the buy rule returns `true`, the trader must buy shares, otherwise no shares are bought. If the trader has no shares on day two the trader executes the buy trading rule again. If the trader has shares, the trader executes the sell trading rule. If the sell trading rule returns `true`, the trader must sell the shares. If the sell trading rule returns `false` the trader

does nothing. The process continues for each trading day. On the last day of trade any shares in the possession of the trader are sold. Each trading action is used to determine a compound excess return for the given buy, sell trading rules. An empirical study presented in Chapter 5 showed that the compound excess return (AK fitness function) was the preferred fitness function. The AK fitness function is therefore used to calculate the fitness value of the combination of buy and sell trading rules. The `EvaluateCombination` function returns the fitness value.

The optimised GP presented in Algorithm 3 in Chapter 5 maintained a global best individual. The global best individual of a population was only updated if the iteration best individual performed better than the global best on both the in sample dataset ($Sample_{in}$) and validation sample dataset ($Sample_{sel}$). The co-operative co-evolution GP implemented in this thesis maintained two global best individuals, one from each population. Because the reward value of an individual is the sum of the fitness values returned by all the evaluations an individual took part in, the global best individual is included in the reward sharing process.

The complete reward sharing process is illustrated in Algorithm 6. The function *RewardSharing* receives five parameters. The first two parameters represent the individuals from the buy population as the array *cBuy* and the global best individual from the buy population. The second two parameters represent the sell population, and the final parameter contains the dataset. The reward process evaluates every combination of individuals from the buy population and sell population using the function presented in Algorithm 7. The fitness result is added to the accumulated individual's reward. The reward process is repeated by evaluating the combination of each individual in the buy population and the global best individual from the sell population, and again combining each individual within the sell population and the global best individual from the buy population. The accumulated reward of each individual is returned.

**Algorithm 4:** Co-operative co-evolved GP algorithm.

$g = 0$; ▷ Generation counter
Set the maximum number of generations $G$;
Set the total number of individuals $\mu$;
Initialise the in sample dataset $Sample_{in}$ ▷ Defined in Section 5.2
Initialise the validation sample dataset $Sample_{sel}$
Initialise buy population $cBuy_g$; ▷ Randomly generate a population of trading rules
Initialise sell population $cSell_g$; ▷ Randomly generate a population of trading rules
Calculate the shared reward of each individual:
    $rBuy_g$, $rSell_g$ = REWARDSHARING($cBuy_g$, $null$, $cSell_g$, $null$, $Sample_{in}$)
Set the best buy individual $bBuy = cBuy_{g,n}, n \in \{rBuy_{g,0}, \ldots, rBuy_{g,\frac{\mu}{2}}\}$, where
    $rBuy_{g,n}$ is the maximum shared reward;
Set the best sell individual $bSell = cSell_{g,n}, n \in \{rSell_{g,0}, \ldots, rSell_{g,\frac{\mu}{2}}\}$, where
    $rSell_{g,n}$ is the maximum shared reward;
**while** $g < G$ **do**
    $g = g + 1$;
    EVOLVE($cBuy_g$, $bBuy$, $cSell_g$, $bSell$, $Sample_{in}$, 'buy');
    EVOLVE($cBuy_g$, $bBuy$, $cSell_g$, $bSell$, $Sample_{in}$, 'sell');
    Calculate the shared reward of each individual using $Sample_{in}$:
        $rbBuy$, $rbSell$, $rBuy_g$, $rSell_g$ = REWARDSHARING($cBuy_g$, $bBuy$, $cSell_g$, $bSell$, $Sample_{in}$)
    Re-calculate the temporary shared reward of each individual using $Sample_{sel}$:
        $rbBuyT$, $rbSellT$, $rBuyT_g$, $rSellT_g$ = REWARDSHARING($cBuy_g$, $bBuy$, $cSell_g$, $bSell$, $Sample_{sel}$)
    Set the best buy individual index $x = rBuy_{g,n}, n \in \{rBuy_{g,0}, \ldots, rBuy_{g,\frac{\mu}{2}}\}$, where
        $rBuy_{g,n}$ is the maximum shared reward;
    **if** $rbBuy > rBuy_{g,x}$ **and** $rbBuyT > rBuyT_x$ **then**
        $bBuy = cBuy_{g,x}$; ▷ New global best buy rule
    **end if**
    Set the best sell individual index $y = rSell_{g,n}, n \in \{rSell_{g,0}, \ldots, rSell_{g,\frac{\mu}{2}}\}$, where
        $rSell_{g,n}$ is the maximum shared reward;
    **if** $rbSell > rSell_{g,y}$ **and** $rbSellT > rSellT_{g,y}$ **then**
        $bSell = cSell_{g,y}$; ▷ New global best sell rule
    **end if**
**end while**
**return** $bBuy, bSell$ as the solution; ▷ Returns both parts of the trading rule

**Algorithm 5:** Co-operative co-evolved GP evolutionary process.

**procedure** EVOLVE($cBuy_g$, $bBuy$, $cSell_g$, $bSell$, $Sample_{in}$, `rule`)
  Set number of offspring $\lambda = 0$;
  **while** $\lambda < (\mu \div 2)$ **do** ▷ $\mu \div 2$ because there are 2 populations
    Calculate the shared reward of each individual using $Sample_{in}$:
      $rbBuy$, $rbSell$, $rBuy_g$, $rSell_g$ = REWARDSHARING($cBuy_g$, $bBuy$, $cSell_g$, $bSell$, $Sample_{in}$)
    Set evolution population $C_g$ and reward share $R_g$:
    **if** `rule` = 'buy' **then**
      $C_g = cBuy_g$;
      $R_g = rBuy_g$;
    **else**
      $C_g = rSell_g$;
      $R_g = rSell_g$;
    **end if**
    Set random number $r \sim U(0,1)$;
    **if** $r < P_m$ **then** ▷ $P_m$ is the mutation probability
      ▷ Sampling is performed using rank selection
      Sample parent $P_1$ from $C_{g-1}$ with bias to the worst $R_{g-1}$;
      Mutate $P_2$ to form $O_1$;
      Sample $D_1$ from $C_g$ with bias to the worst shared reward $R_g$;
      Remove $D_1$ from $C_g$;
      Add $O_1$ to $C_g$;
      $\lambda = \lambda + 1$;
      } Mutation
    **else**
      ▷ Sampling is performed using rank selection
      Sample parents $P_1$ and $P_2$ from $C_{g-1}$ with bias to the worst $R_{g-1}$;
      Offspring $O_1$ = Re-combination of $P_1$ and $P_2$;
      Offspring $O_2$ = Re-combination of $P_2$ and $P_1$;
      Add $O_1$ and $O_2$ to $C_g$;
      Sample $D_1$ and $D_2$ from $C_g$ with bias to the worst $R_g$;
      Remove $D_1$ and $D_2$ from $C_g$;
      $\lambda = \lambda + 2$;
      } Crossover
    **end if**
  **end while**
**end procedure**

**Algorithm 6:** Co-operative co-evolution reward sharing algorithm.

**function** REWARDSHARING(*cBuy*, *bBuy*, *cSell*, *bSell*, *Sample*)
  Initialise the total reward *rBuy* of each buy individual in the population *cBuy* to 0;
  Initialise the total reward *rSell* of each sell individual in the population *cSell* to 0;
  **for** $x = 0; x < (\mu \div 2); x = x + 1$; **do** ▷ 2, Because there are 2 populations
    **for** $y = 0; y < (\mu \div 2); y = y + 1$; **do**
      Calculate the fitness $f$ = EVALUATECOMBINATION($cBuy_x$, $cSell_y$, *Sample*);
      Update the total reward of the individuals used in the evaluation:
        $rBuy_x = rBuy_x + f$;
        $rSell_y = rSell_y + f$;
    **end for**
  **end for**
  **if** $bBuy \neq null$ **and** $bSell \neq null$ **then**
    Initialise total reward *rbBuy* of the best buy individual *bBuy* to 0;
    Initialise total reward *rbSell* of the best sell individual *bSell* to 0;
    **for** $y = 0; y < (\mu \div 2); y = y + 1$; **do**
      Calculate the fitness $f$ = EVALUATECOMBINATION(*bBuy*, $cSell_y$, *Sample*);
      Update the total reward of the individuals used in the evaluation:
        $rbBuy = rbBuy + f$;
        $rSell_y = rSell_y + f$;
    **end for**
    **for** $x = 0; x < (\mu \div 2); x = x + 1$; **do**
      Calculate the fitness $f$ = EVALUATECOMBINATION(*bSell*, $cBuy_x$, *Sample*);
      Update the total reward of the individuals used in the evaluation:
        $rbSell = rbSell + f$;
        $rBuy_x = rBuy_x + f$;
    **end for**
  **end if**
  **return** *rbBuy*, *rbSell*, *rBuy*, *rSell*; ▷ Return the shared rewards
**end function**

**Algorithm 7:** Co-operative co-evolved trading rule evaluation.

```
function EVALUATECOMBINATION(BuyRule, SellRule, Sample)
    Set boolean s = false;                          ▷ The trader has no shares on the first day
    Set fitness f = 0;
    for i = 0; i < LENGTH(Sample); i = i + 1; do
        if s == false then
            Set buy action a = BuyRule(Sample_i);   ▷ Determine the action using the BuyRule
            if a == true then
                Buy shares;
                s = true;
            end if
            Update fitness value f using fitness function and action;
        else
            Set sell action a = SellRule(Sample_i); ▷ Determine the action using the SellRule
            if a == true then
                Sell shares;
                s = false;
            end if
            Update fitness value f using fitness function and action;
        end if
    end for
    if s == true then
        Sell shares;
        Update fitness value f using fitness function and action;
    end if
    return f;                  ▷ Return the fitness value of the BuyRule, SellRule combination
end function
```

### 6.1.2 Competitive Co-evolved Genetic Program

Competitive co-evolution requires two or more populations that evolve in competition, each population reciprocally driving one another to increasing levels of performance and complexity [79, 185]. The performance of one population is inversely proportional to the performance of another, implying the fitness of one individual is dependent on the fitness of another. Angeline and Pollack [47] defined a relative fitness function as any fitness calculation that is dependent on other individuals to calculate the fitness measure [47].

The single population evolutionary process defined in Algorithm 3 in Chapter 5 was modified to support a relative fitness function, and more than one population. The modified evolutionary process is presented in Algorithm 8. Instead of a single population, the competitively co-evolved algorithm begins by initialising $P$ populations of equal size. Individuals within a population are initialised using the same approach as the single population GP presented in Chapter 5. The size of each population is determined by the configured total number of individuals $\mu$ divided by the number of populations $P$.

Just like the single population GP, each individual within a population represents a complete trading rule. The trading rule returns a boolean value of `true` or `false`, which is transcoded to buy, sell or hold depending on whether the trader has shares or not. The evaluation of a competitively co-evolved individual is determined by a relative fitness value as opposed to the absolute fitness value used in a single population GP.

The relative fitness calculation is presented in Algorithm 10 as two functions: `CalcAllRelative Fitness` and `CalcRelativeFitness`. The function `CalcAllRelativeFitness` requires true parameters. The first parameter $C_g$ contains all the iteration individuals from every population. The second parameter denotes the current population $p$. `CalcAllRelativeFitness` iterates through each individual in population $p$ calling function `CalcRelativeFitness`.

`CalcRelativeFitness` compares an individual from population $p$ to all individuals in all other populations. Comparison is done by calculating the absolute fitness of each individual to determine which individual has the largest absolute fitness value. The individual from population $p$ receives a point each time its absolute fitness value is greater than the absolute fitness value of the individual it is compared too [79, 132, 185]. `CalcRelativeFitness` returns the total awarded points as the relative fitness value of the individual.

The optimised GP presented in Algorithm 3 in Chapter 5 maintains a global best individual. The competitive co-evolution GP implemented in this thesis maintains a global best individual for each population. At the end of each generation the relative fitness value of the global best individuals are re-calculated by comparing the global best individual to all the individuals in all the other populations. The best individual from the current population is the individual that has the largest relative fitness value across all the individuals within the current population. The best individual from the current population is then compared to the global best individual from the current population. The iteration best individual from the population becomes the new global best individual, if its $Sample_{in}$, and $Sample_{sel}$ relative fitness values are greater than the relative fitness values of the global best individual of the population. At the end of evolution, the competitive co-evolved GP returns the overall global best individual as the solution.

**Algorithm 8:** Competitive co-evolved GP algorithm.

$g = 0$; ▷ Generation counter
Set the maximum number of generations $G$;
Set the maximum number of populations $P$;
Set the total number of individuals $\mu$;
Initialise the in sample dataset $Sample_{in}$ ▷ Defined in Section 5.2
Initialise the validation sample dataset $Sample_{sel}$
**for** $p = 0; p < P; p = p + 1$; **do**
    Initialise initial generation's population $C_{p,g}$; ▷ Randomly generate a population of trading rules
**end for**
**for** $p = 0; p < P; p = p + 1$; **do**
    Calculate relative fitness $rf$ for each individual in $C_{p,g}$
        $rf_{p,g}$ = CALCALLRELATIVEFITNESS($C_g$, $p$);
    Set the population's global best individual for this population $pB_p = C_{p,g,n}, n \in C_{p,g}$, where
        $pB_p$ produces the maximum relative fitness $rf_p$;
**end for**
**while** $g < G$ **do**
    $g = g + 1$;
    **for** $p = 0; p < P; p = p + 1$; **do**
        EVOLVE($C$, $p$, $g$);
        Calculate relative fitness $rf$ for each individual in $C_{p,g}$
            $rf_{p,g}$ = CALCALLRELATIVEFITNESS($C_g$, $p$);
        Set the best individual index $x$, for this population and generation:
            $x = rf_{p,g,n}, n \in \{rf_{p,g,0}, \ldots, rf_{p,g,\frac{\mu}{P}}\}$, where
            $rf_{p,g,n}$ is the maximum relative fitness value;
        Re-calculate the population's global best individual's relative fitness using $Sample_{in}$:
            $rfB$ = CALCRELATIVEFITNESS($pB_p$, $Sample_{in}$, $C_g$, $p$);
        **if** $rfB > rf_{p,g,x}$ **then**
            Calculate temporary relative fitness using the validation sample $Sample_{sel}$:
                $rfI$ = CALCRELATIVEFITNESS($C_{p,g,x}$, $Sample_{sel}$, $C_g$, $p$);
                $rfB$ = CALCRELATIVEFITNESS($pB_p$, $Sample_{sel}$, $C_g$, $p$);
            **if** $rfI > rfB$ **then**
                Set $pB_p = C_{p,g,x}$; ▷ New global population best
            **end if**
        **end if**
    **end for**
**end while**
**return** $B_x, x \in \{pB_0, \ldots, pB_P\}$, where ▷ Return the best individual across all the populations
    $B_x$ produces the maximum relative fitness for both $Sample_{in}$ and $Sample_{sel}$;

**Algorithm 9:** Competitive co-evolved GP evolutionary process.

▷ Evolve receives all populations as $C$

**procedure** EVOLVE($C$, $p$, $g$)

Set number of offspring $\lambda = 0$;

**while** $\lambda < (\mu \div P)$ **do** ▷ $\mu \div P$ because there are P populations

Set random number $r \sim U(0,1)$;

**if** $r < P_m$ **then** ▷ $P_m$ is the mutation probability

▷ Sampling is performed using rank selection

Sample parent $P_1$ from $C_{p,g-1}$ with bias to worst $rf_{p,g-1}$;

Mutate $P_2$ to form $O_1$;

Sample $D_1$ from $C_{p,g}$ with bias to worst $rf_{p,g}$;

Remove $D_1$ from $C_{p,g}$;

Add $O_1$ to $C_{p,g}$ as $C_{p,g,n+1}$, where $n$ is the size of $C_{p,g}$;

Set relative fitness:

$rf_{p,g,n+1}$ = CALCRELATIVEFITNESS($C_{p,g,n+1}$, $Sample_{in}$, $C_g$, $p$);

$\lambda = \lambda + 1$;

Mutation

**else**

▷ Sampling is performed using rank selection

Sample parents $P_1$ and $P_2$ from $C_{p,g-1}$, with bias to best $rf_{p,g-1}$;

Offspring $O_1$ = Re-combination of $P_1$ and $P_2$;

Offspring $O_2$ = Re-combination of $P_2$ and $P_1$;

Sample $D_1$ and $D_2$ from $C_{p,g}$ with bias to worst $rf_{p,g}$;

Remove $D_1$ and $D_2$ from $C_{p,g}$;

Add $O_1$ and $O_2$ to $C_{p,g}$ as $C_{p,g,n+1}$ and $C_{p,g,n+2}$, where $n$ is the size of $C_{p,g}$;

Set relative fitness:

$rf_{p,g,n+1}$ = CALCRELATIVEFITNESS($C_{p,g,n+1}$, $Sample_{in}$, $C_g$, $p$);

Set relative fitness:

$rf_{p,g,n+2}$ = CALCRELATIVEFITNESS($C_{p,g,n+2}$, $Sample_{in}$, $C_g$, $p$);

$\lambda = \lambda + 2$;

Crossover

**end if**

**end while**

**end procedure**

**Algorithm 10:** Competitive co-evolved GP relative fitness algorithms.

▷ $C$ denotes all the populations, and generations
**function** CALCALLRELATIVEFITNESS($C_g$, $p$)
  **for** $n = 0; n < (\mu \div P); n = n + 1$; **do** ▷ For each individual within the population
    Set the relative fitness $rf$ of the individual $n$ in population $p$, and generation $g$:
      $rf_{p,n}$ = CALCRELATIVEFITNESS($C_{p,g,n}$, $Sample_{in}$, $p$);
  **end for**
  **return** $rf_{p,g}$ ▷ Return the fitness values for all individuals within the population and generation
**end function**
**function** CALCRELATIVEFITNESS(*Individual*, *Sample*, $C_g$, $p$)
  Set relative fitness $rf = 0$;
  Calculate the *Individual*'s fitness value $f$ using a fitness function and the *Sample* dataset;
  **for** $i = 0; i < P; i = i + 1$; **do**
    **if** $i \neq p$ **then** ▷ Compare the *Individual* against individuals from other populations
      **for** $n = 0; n < (\mu \div P); n = n + 1$; **do** ▷ For each individual within the population
        Calculate the temporary *Sample* fitness value $t$ of $C_{p,g,n}$,
          using a fitness function and the *Sample* dataset;
        *Individual* gets a point each time its fitness $f$
          is greater than an individual in another population
        **if** $f > t$ **then**
          $rf = rf + 1$;
        **end if**
      **end for**
    **end if**
  **end for**
  **return** $fR$; ▷ Return the individual's relative fitness for the sample provided
**end function**

## 6.2 Summary

This chapter extended the single population GP presented in Chapter 5 to form two different multi-population co-evolution approaches. The first approach a co-operative co-evolved GP and the second a competitive co-evolved GP. The co-evolved GPs and the single population GPs are compared in the next chapter to determine if evolved and co-evolved trading rules can outperform a buy-and-hold strategy, and if a co-evolved GP performs better than a single population GP.

# Chapter 7

# Empirical Study

*The true method of knowledge is experiment.*

*-William Blake (English painter, poet and printmaker)*

This chapter compares an optimised GP to two co-evolved GPs to show that trading rules evolved by a co-operative co-evolved GP does not perform as well as rules evolved by either a standard GP or a competitively co-evolved GP. This chapter begins with Section 7.1, describing the study and the empirical process implemented. Section 7.2 compares the evolved trading rules applied to a real world market without the complexities of transaction costs. The ability of the evolved traders to trade using real world fees is discussed in Section 7.3. Section 7.4 compares the evolved traders to four random trading strategies. The chapter concludes with a summary in Section 7.5.

## 7.1 Empirical Study

This thesis examines the profitability of evolved stock market trading rules on the JSE. A trading rule is a result of an evolutionary process. The basic evolutionary process described in Chapter 5 was extended to co-evolution, a multi-population evolutionary process. Two variations of co-evolution, namely co-operative and competitive co-evolution, were defined in Chapter 6. The three GPs were used to evolve trading rules over a set of shares listed on the JSE. A complete list of the shares used was presented in Table 5.2.

Following the same empirical process as presented in Chapter 5, an independent simulation was run for each of the three GPs. Each simulation was repeated for each of the shares. To ensure that the simulations were evaluated fairly, a fixed set of random seeds were used. Each simulation run was assigned one of the 30 random seeds. This ensured that each simulation was repeatable, and differences in the performance were not due to random chance, but due to differences in the algorithm's implementation.

Each GP used a fixed number of individuals determined by parameter sensitivity analysis outlined in Chapter 5. To ensure that all three GPs were initialised in the same way, 30 random seeds were used to generate 30 sets of $\mu$ individuals. These individuals were used as the initial individuals for each of the three GPs. This ensured that differences in performance of the GPs were due to differences in the algorithm's implementation and not the initial populations.

Each simulation was configured using the same evolutionary operators and parameters. The parameter sensitivity analysis presented in Chapter 5 determined that a mutation probability of 35%, a population size ($\mu$) of 1600, and a maximum number of 125 generations were the preferred GP configuration parameters for this thesis. An empirical analysis in Chapter 5 determined that rank selection was the preferred selection operator. Rank selection was used to determine which individuals underwent mutation, crossover, and which individuals survived to the next generation. An empirical analysis

in Chapter 5 determined that the compounded excess returns fitness function (AK) defined by Allen and Karjalainen [43] was the preferred fitness function for this thesis.

Allen and Karjalainen noted that lowering the trading costs increases the returns of the GP [43]. Neely *et al.* [170], Telbany [102], Mahfoud and Mani [160, 161], Li and Tsang [204, 205], and Potvina *et al.* [177] reimplemented the work of Allen and Karjalainen using various fee structures with varying degrees of success. Because of this, the simulations were repeated twice, once without fees and once with fees. The results without fees are discussed in Section 7.2, and the analysis of the results with fees are discussed in Section 7.3. Section 7.4 compares the trading results presented in Section 7.3 to four random trading strategies.

## 7.2 Excluding fees

This section compares the profitability of trading rules evolved using three different GP approaches, namely a single population GP, competitively co-evolved GP, and a co-operative co-evolved GP. The results discussed in this section excluded trading fees. The profit at the end of trading was recorded for each of the trading rules. The Iman and Davenport extension [142] of the Friedman statistical test was run on profit returned by the best individuals from each simulation, dataset, and share. The results of the Iman and Davenport test are presented in Table 7.1.

The Iman and Davenport test results show that a significant difference exists between the results returned by the evolved trading rules. A Friedman pairwise test was performed on the significantly different $Sample_{out}$ results to determine which of the GP's results were significantly different. The results are presented in Figure 7.1. The Friedman test showed that the results returned by the evolved trading rules using a co-operative co-evolved GP were significantly different from the other results.

The critical difference plots presented in Figure 7.1 confirm the p-value results and show that the results of the co-operative co-evolved GP are significantly worse than the results returned by the trading rules evolved using single population evolution and competitive co-evolution.

The mean profit obtained by the best individuals for each simulation is summarised in Table 7.2. The mean profit results show that the co-operative co-evolved GP performed significantly worse than the other two approaches for 10 of the 11 significantly different $Sample_{out}$ share datasets. The box plots presented in Figure 7.3 depict how poorly the co-operative co-evolved trading rules performed. Except for NED, the results returned by the evolved trading rules using the standard single population GP and the competitively co-evolved GP have smaller boxes denoting a smaller deviation in the results, and a higher mean than the co-operative co-evolved GP across all the significantly different shares. The mean

deviation measures the average difference that an individual simulation run result is from the overall simulation average. This implies that the smaller the deviation, the less chance the results are skewed by outliers and therefore more reliable.

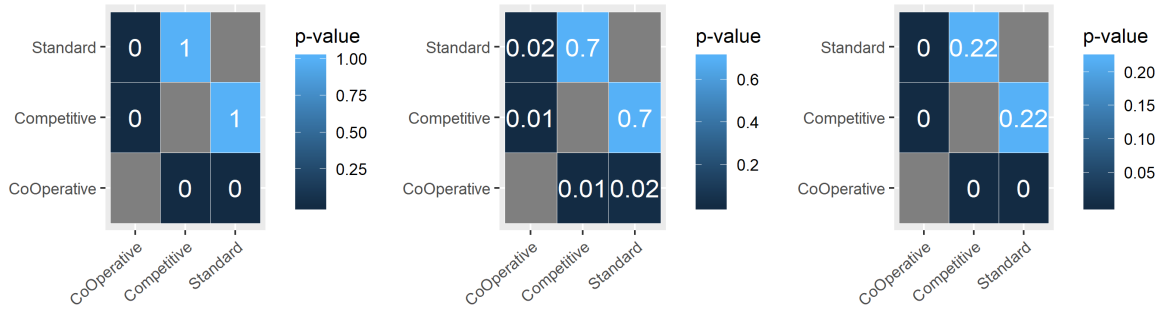
The box plots in Figure 7.3 in conjunction with the mean results in Table 7.2 show that the competitively co-evolved trading rules performed better than the trading rules evolved using the single population GP when trading ALSI40, BIL, NED, REM, SBK, SAB, and SOL. The trading rules evolved using the single population GP performed better than the competitively co-evolved trading rules when trading INVNED, INVREM, INVSBK, and RCH. However, except for the INVREM results, the Friedman test failed to find a significant difference between the results returned by the trading rules evolved using competitive co-evolution and the results returned by the trading rules evolved using single population evolution.

The results show that the trading rules evolved using competitive co-evolution and single population evolution performed equally well across all shares, except for NED and INVREM. When trading NED, a cooperative co-evolved GP was preferred. When trading INVREM, a single population GP was preferred.

**Table 7.1:** p-value results without fees using Iman and Davenport test with Finner's correction.

| | $Sample_{in}$ | | $Sample_{sel}$ | | $Sample_{out}$ | |
|---|---|---|---|---|---|---|
| **Share Code** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** |
| **agl** | **0.005204** | **0.005204** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **alsi** | **0.000000** | **0.000000** | **0.001532** | **0.001767** | 0.130636 | 0.160536 |
| **bil** | **0.000001** | **0.000001** | 0.335652 | 0.335652 | **0.000000** | **0.000000** |
| **gfi** | **0.000002** | **0.000002** | **0.000000** | **0.000000** | 0.677553 | 0.707859 |
| **imp** | **0.000000** | **0.000000** | **0.024323** | **0.026038** | 0.991976 | 0.991976 |
| **inv-ned** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **inv-rem** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **inv-sbk** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **lon** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | 0.655771 | 0.707859 |
| **ned** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000007** | **0.000010** |
| **rch** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **rem** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **sab** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000003** | **0.000004** |
| **sbk** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000002** | **0.000003** |
| **sol** | **0.000000** | **0.000000** | **0.000238** | **0.000298** | **0.000000** | **0.000000** |

Note: **bold** values represent a significant difference.



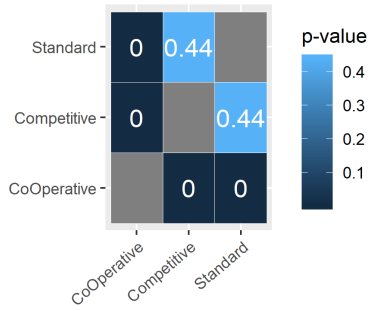

**(a)** $Sample_{out}$ AGL



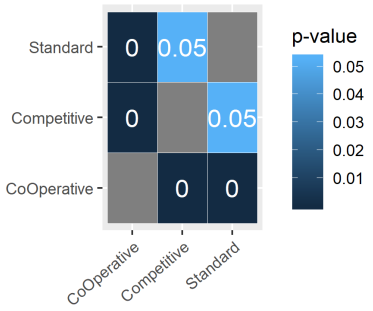

**(b)** $Sample_{out}$ ALSI



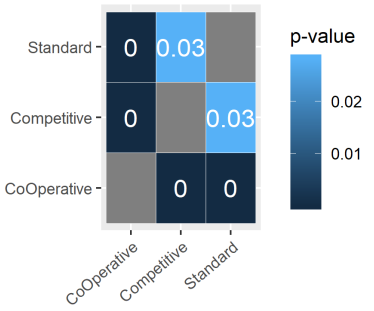

**(c)** $Sample_{out}$ BIL



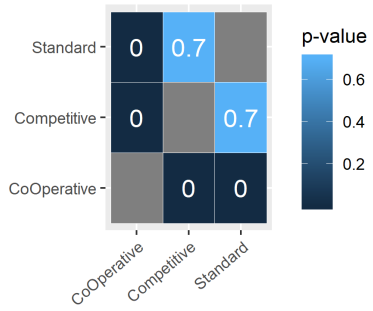

**(d)** $Sample_{out}$ INV-NED



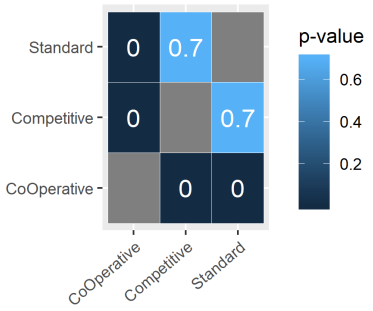

**(e)** $Sample_{out}$ INV-REM



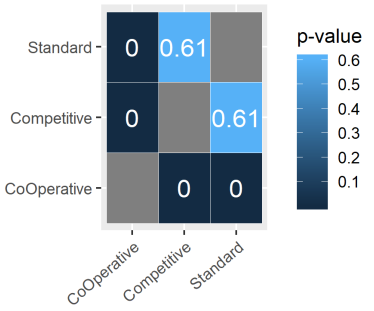

**(f)** $Sample_{out}$ INV-SBK



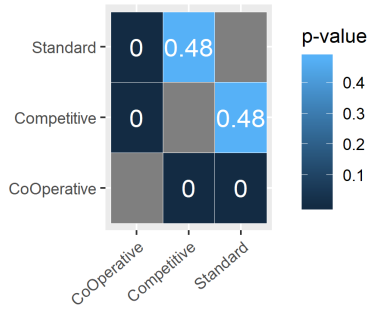

**(g)** $Sample_{out}$ NED



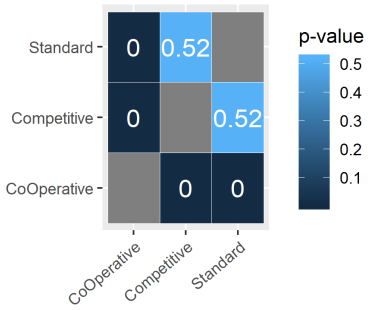

**(h)** $Sample_{out}$ RCH



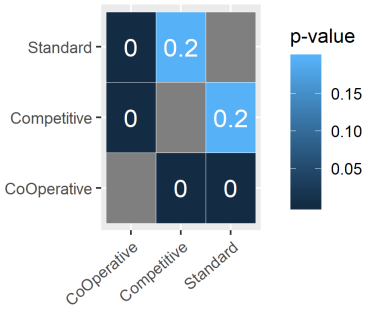

**(i)** $Sample_{out}$ REM




**(j)** $Sample_{out}$ SAB

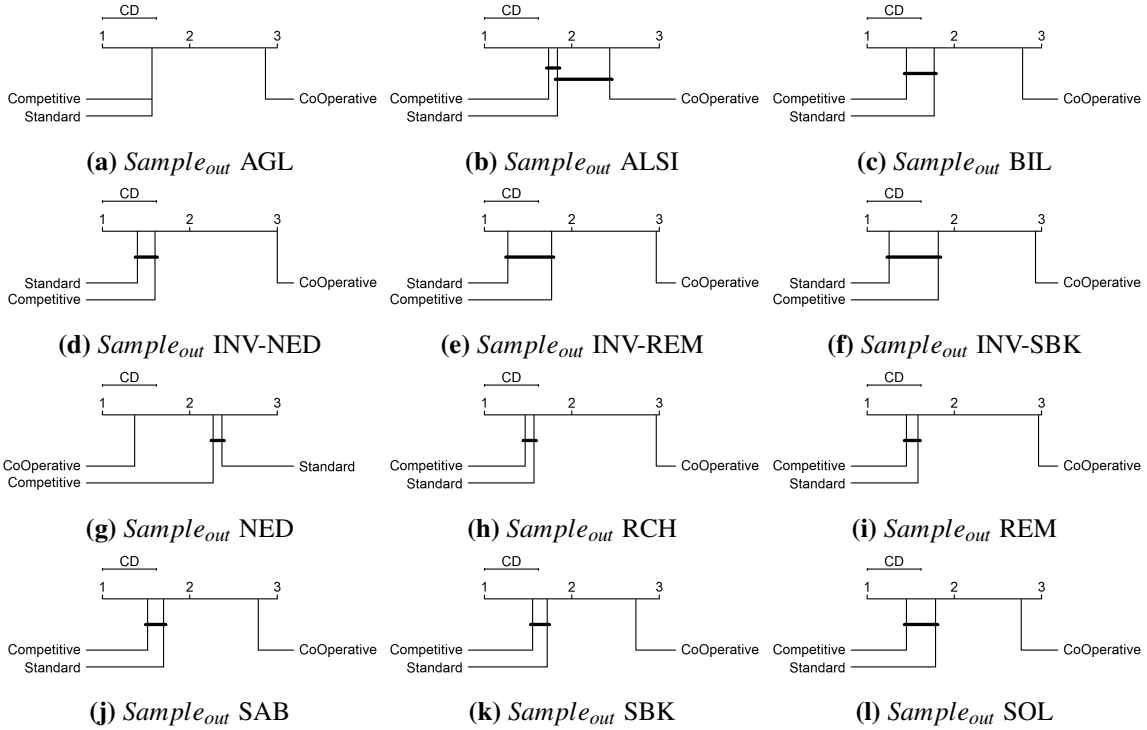



**(k)** $Sample_{out}$ SBK




**(l)** $Sample_{out}$ SOL

**Figure 7.1:** Friedman pairwise comparison of p-values for results without fees.




**(a)** $Sample_{out}$ AGL **(b)** $Sample_{out}$ ALSI **(c)** $Sample_{out}$ BIL

**(d)** $Sample_{out}$ INV-NED **(e)** $Sample_{out}$ INV-REM **(f)** $Sample_{out}$ INV-SBK

**(g)** $Sample_{out}$ NED **(h)** $Sample_{out}$ RCH **(i)** $Sample_{out}$ REM

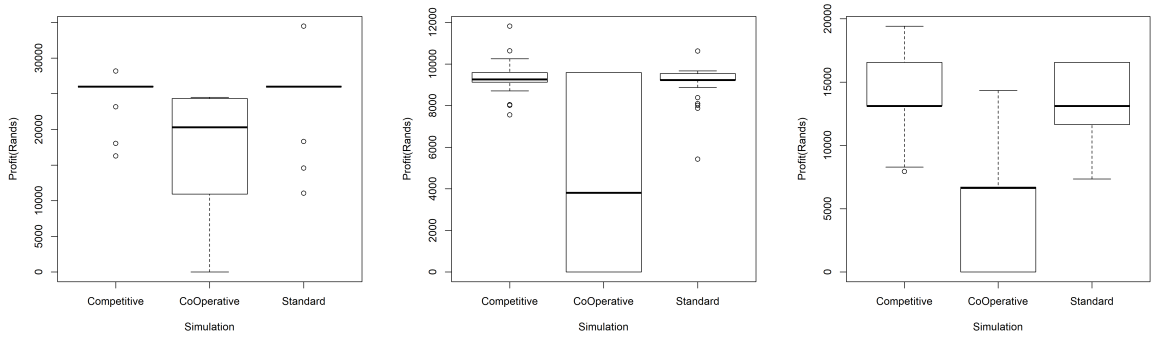
**(j)** $Sample_{out}$ SAB **(k)** $Sample_{out}$ SBK **(l)** $Sample_{out}$ SOL

**Figure 7.2:** Critical difference plots for results without fees.

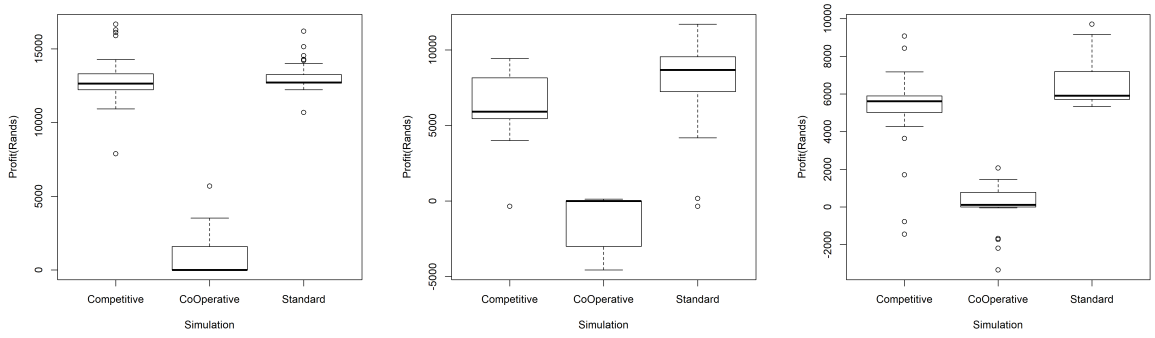
**Table 7.2:** Mean $Sample_{out}$ profit per share without fees.

| Share Code | Co-Operative | Competitive | Standard |
|---|---|---|---|
| **agl** | *17518.23* | **25381.37** | 25139.13 |
| **alsi** | *4783.57* | **9316.80** | 9065.63 |
| **bil** | *5870.03* | **14118.47** | 13075.57 |
| **gfi** | -419.67 | **131.97** | *-1280.03* |
| **imp** | *13243.33* | **16291.73** | 15982.73 |
| **inv-ned** | *929.77* | 12892.07 | **13091.80** |
| **inv-rem** | *-1251.40* | 6323.30 | **7977.53** |
| **inv-sbk** | *130.83* | 5037.13 | **6693.43** |
| **lon** | *13807.13* | **17522.17** | 15756.33 |
| **ned** | **-219.60** | -2279.83 | *-2848.47* |
| **rch** | *641.73* | 2184.63 | **2187.70** |
| **rem** | *5108.03* | **9835.50** | 9762.33 |
| **sab** | *1868.63* | **5453.03** | 4969.83 |
| **sbk** | *249.20* | **1692.23** | 1555.63 |
| **sol** | *5010.50* | **16213.47** | 13053.43 |

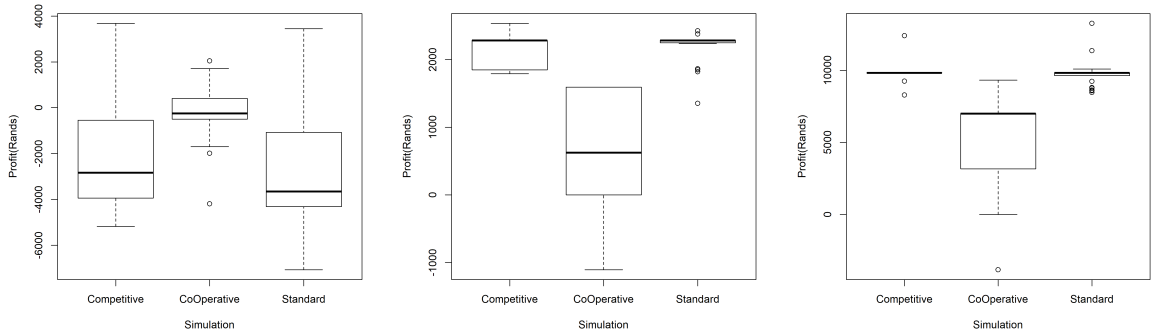
Note: **bold** represents the highest value, and *italic* the lowest value.



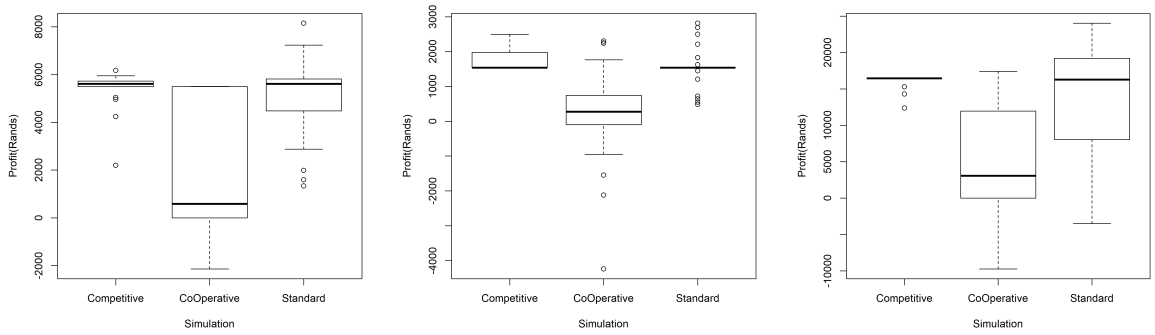

**(a)** $Sample_{out}$ AGL **(b)** $Sample_{out}$ ALSI **(c)** $Sample_{out}$ BIL




**(d)** $Sample_{out}$ INV-NED **(e)** $Sample_{out}$ INV-REM **(f)** $Sample_{out}$ INV-SBK




**(g)** $Sample_{out}$ NED **(h)** $Sample_{out}$ RCH **(i)** $Sample_{out}$ REM




**(j)** $Sample_{out}$ SAB **(k)** $Sample_{out}$ SBK **(l)** $Sample_{out}$ SOL

**Figure 7.3:** Box-plots of $Sample_{out}$ profit results without fees.

## 7.3 Including fees

This section repeats the simulations of Section 7.2 to determine the effect that fees has on the performance of the evolved trading rules. The fee structure defined in Section 2.4 was used. Each trade was made with 1000 shares. The mean profit returned by the evolved trading rules are presented in Table 7.3. The results show that the competitive co-evolved trading rules returned the highest mean profit in 11 of the 15 shares and the co-operative co-evolved trading rules produced the lowest mean profit across 14 of the 15 shares.

The mean profit deviation results presented in Table 7.3 show that the competitively co-evolved trading rules were the most consistent resulting in the lowest deviation across 11 of the 15 shares. The Iman and Davenport [142] statistical test with Finner's correction was done on the mean profit results to determine if the simulation results are significantly different. The p-values of the statistical test presented in Table 7.4 show that a significant difference exists across all the share simulations.

The results of a post-hoc analysis of the profit results are presented in Figure 7.4. The p-values show that the profit results returned by the co-operatively co-evolved trading rules were significantly different from the results of the other two algorithms. The critical difference plots presented in Figure 7.5, in conjunction with the mean profit results of the best individuals, confirm that the trading rules evolved using the co-operative co-evolved GP performed significantly worse than the other trading rules.

The p-values obtained from the post-hoc analysis of the INVNED and SOL profit results show that the results obtained from the trading rules evolved using a competitive co-evolved GP are significantly different from that of the single population GP results. The results show that the trading rules evolved using the competitive co-evolved GP performed significantly better than the trading rules evolved using the standard GP when trading SOL. However, trading rules evolved using the standard GP performed significantly better than the trading rules evolved using the competitive co-evolved GP when trading INVNED. The trading rules evolved using a competitive co-evolved GP performed better than the evolved trading rules using the single single population GP for 10 of the 15 shares.

**Table 7.3:** Mean $Sample_{out}$ profit, and standard deviation per share with fees.

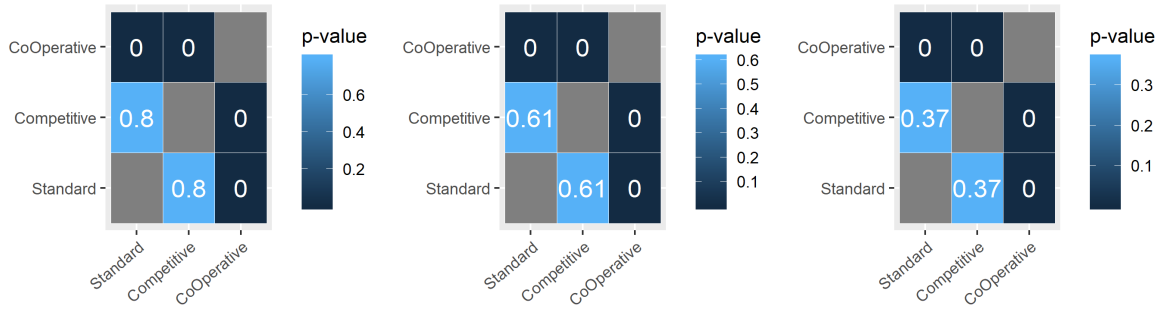
| | Standard | | Competitive | | Co-Operative | |
|---|---|---|---|---|---|---|
| **Share Code** | **Mean Profit** | $\sigma$ | **Mean Profit** | $\sigma$ | **Mean Profit** | $\sigma$ |
| **agl** | 233796.07 | ±8624.00 | **238416.06** | ±**0.00** | *116075.26* | ±*116075.26* |
| **alsi** | 91128.61 | ±2851.79 | **92640.05** | ±**258.73** | *33711.21* | ±*43035.70* |
| **bil** | 123908.93 | ±7832.92 | **128804.51** | ±**0.00** | *44087.70* | ±*56477.87* |
| **gfi** | **-12889.49** | ±13785.28 | -13837.08 | ±*14357.16* | *-27415.77* | ±**11547.11** |
| **imp** | 169605.59 | ±5426.77 | **172412.54** | ±**0.00** | *122435.69* | ±*58605.91* |
| **inv-ned** | **54435.84** | ±6730.08 | 46679.36 | ±*9018.08* | *250.53* | ±**515.66** |
| **inv-rem** | 6769.87 | ±*10045.23* | **10404.96** | ±**4781.11** | *-2952.77* | ±5511.83 |
| **inv-sbk** | **20998.47** | ±**2718.97** | 14849.33 | ±8208.28 | *-5591.64* | ±*9319.41* |
| **lon** | 196280.74 | ±13085.38 | **197359.67** | ±**10999.45** | *98341.61* | ±*90639.88* |
| **ned** | -48931.84 | ±28239.59 | *-54749.04* | ±*32513.65* | **-1393.64** | ±**4543.67** |
| **rch** | **15436.26** | ±**0.00** | **15436.26** | ±**0.00** | *4286.16* | ±*6690.06* |
| **rem** | 67225.70 | ±960.79 | **67722.66** | ±**0.00** | *33692.59* | ±*31862.22* |
| **sab** | 49231.25 | 6338.79 | **53030.66** | ±**0.00** | *17676.89* | ±*23569.18* |
| **sbk** | 6152.10 | ±131.01 | **6219.87** | ±**0.00** | *1800.57* | *3240.82* |
| **sol** | 98461.25 | ±55094.09 | **142051.53** | ±**0.00** | *46388.96* | ±*61851.94* |

Note: the profit values in **bold** and *italic* represents the highest and lowest profit values respectively, and the $\sigma$ values in **Bold** and *italic*, represents the lowest value or more preferred $\sigma$ and the highest or least preferred $\sigma$ respectively.

**Table 7.4:** p-value results with fees using Iman and Davenport test with Finner's correction.

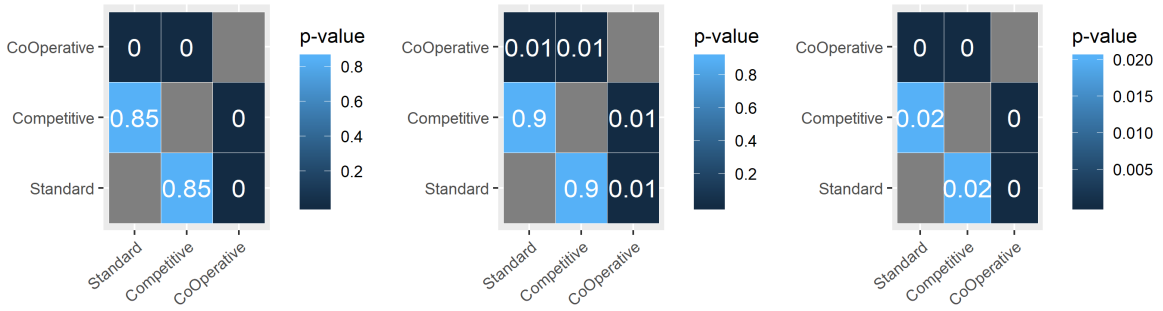
| | $Sample_{in}$ | | $Sample_{sel}$ | | $Sample_{out}$ | |
|---|---|---|---|---|---|---|
| **Share Code** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** | **P-Value** | **Corrected P-Value** |
| **agl** | **0.000000** | **0.000000** | 0.878819 | 0.895777 | **0.000912** | **0.001053** |
| **alsi** | **0.006682** | **0.006682** | **0.000000** | **0.000001** | **0.000026** | **0.000065** |
| **bil** | **0.000001** | **0.000001** | **0.000001** | **0.000002** | **0.000018** | **0.000053** |
| **gfi** | **0.000000** | **0.000000** | **0.005621** | **0.007658** | **0.000190** | **0.000237** |
| **imp** | **0.000000** | **0.000000** | 0.797369 | 0.841494 | **0.005621** | **0.005621** |
| **inv-ned** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **inv-rem** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000026** | **0.000065** |
| **inv-sbk** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000000** |
| **lon** | **0.000000** | **0.000000** | 0.475932 | 0.554103 | **0.000109** | **0.000148** |
| **ned** | **0.000000** | **0.000000** | **0.000190** | **0.000356** | **0.000000** | **0.000000** |
| **rch** | **0.000002** | **0.000002** | **0.000335** | **0.000558** | **0.000002** | **0.000006** |
| **rem** | **0.000000** | **0.000000** | 0.929981 | 0.929981 | **0.003668** | **0.003930** |
| **sab** | **0.000003** | **0.000003** | **0.000003** | **0.000005** | **0.000037** | **0.000065** |
| **sbk** | **0.000002** | **0.000002** | **0.002402** | **0.003601** | **0.000031** | **0.000065** |
| **sol** | **0.000000** | **0.000000** | **0.000000** | **0.000000** | **0.000029** | **0.000065** |

Note: **bold** values represent a significant difference.




**(a)** $Sample_{out}$ AGL **(b)** $Sample_{out}$ ALSI **(c)** $Sample_{out}$ BIL

**Figure 7.4:** Friedman pairwise comparison of p-values for results with fees.



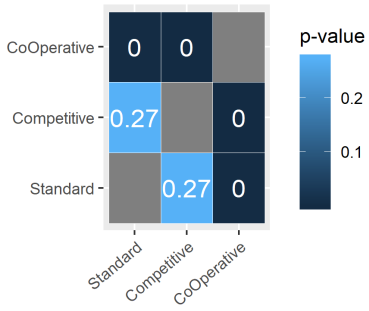

**(d)** $Sample_{out}$ GFI



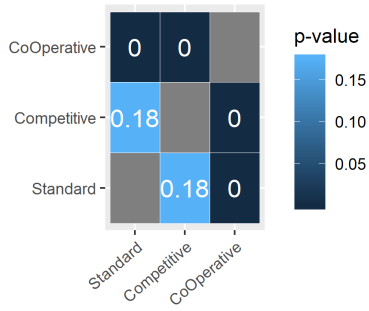

**(e)** $Sample_{out}$ IMP



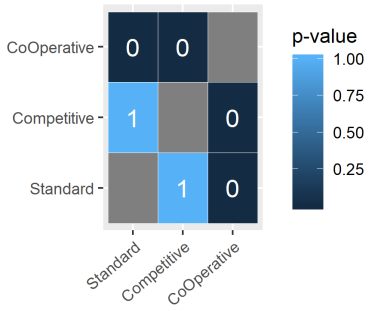

**(f)** $Sample_{out}$ INV-NED



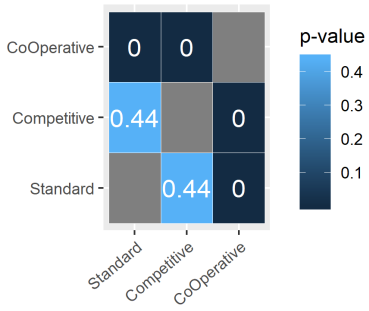

**(g)** $Sample_{out}$ INV-REM



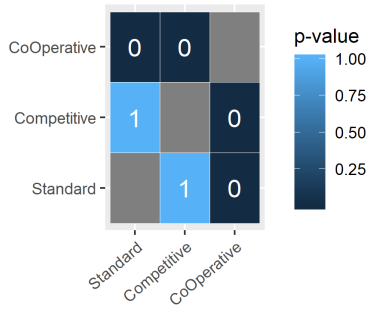

**(h)** $Sample_{out}$ INV-SBK



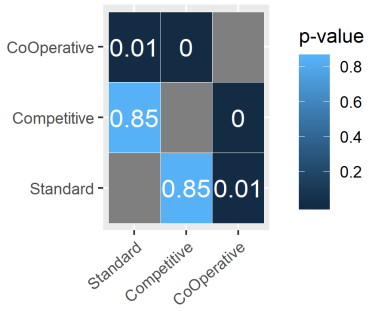

**(i)** $Sample_{out}$ LON



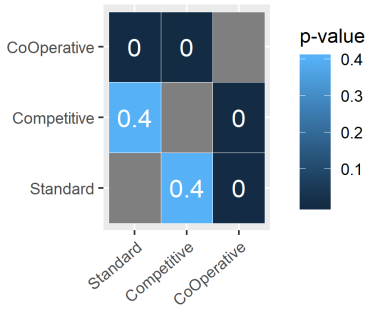

**(j)** $Sample_{out}$ NED



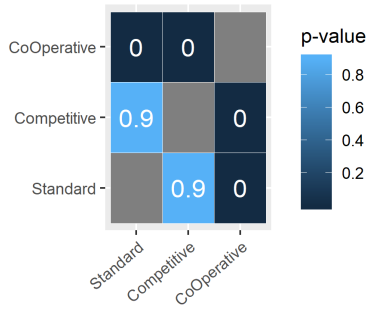

**(k)** $Sample_{out}$ RCH



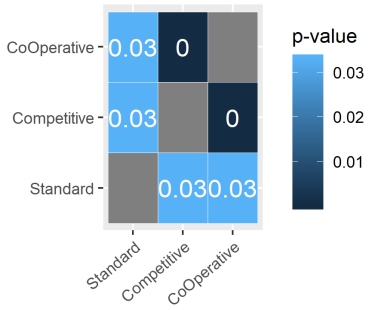

**(l)** $Sample_{out}$ REM




**(m)** $Sample_{out}$ SAB

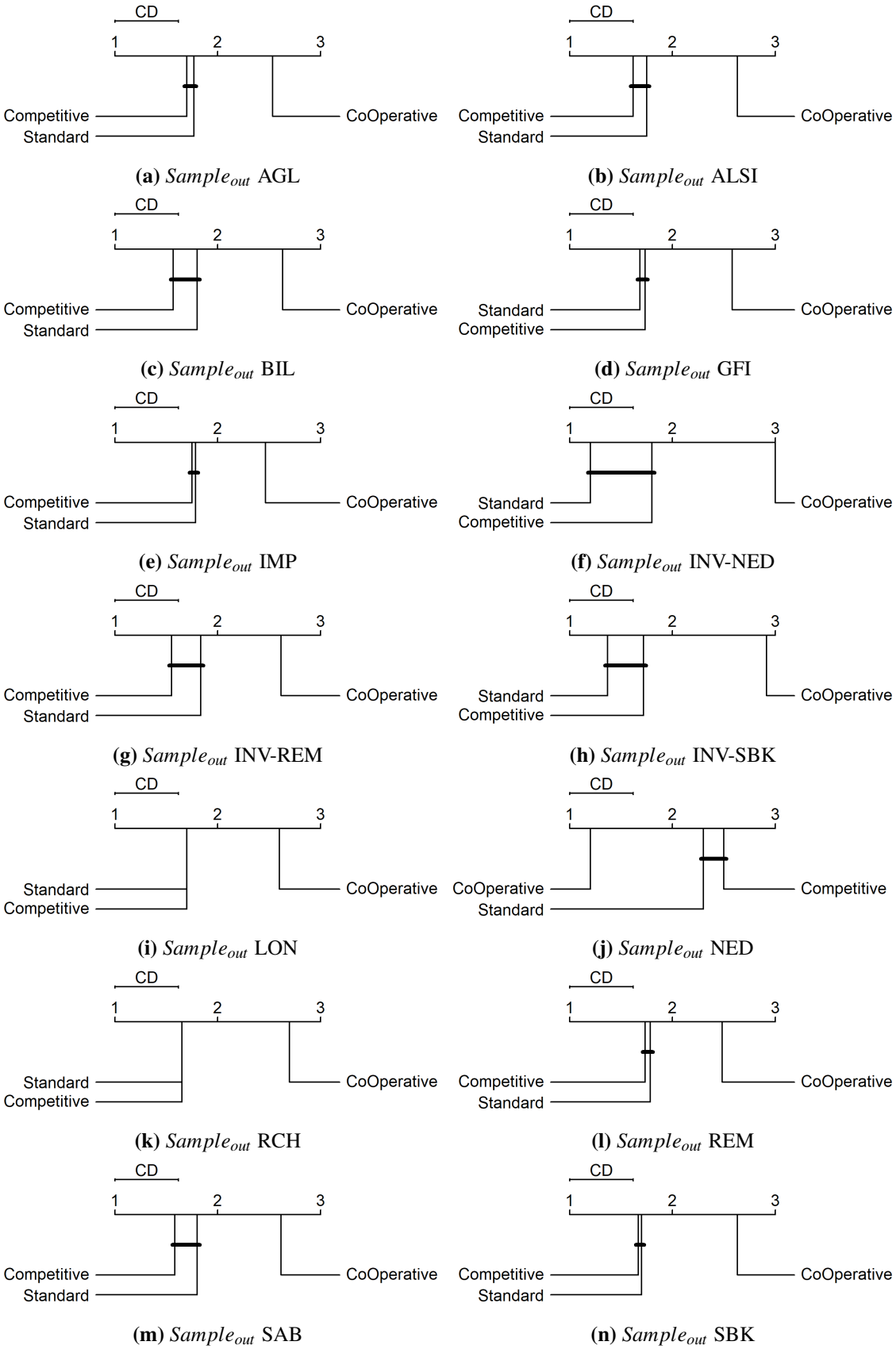



**(n)** $Sample_{out}$ SBK




**(o)** $Sample_{out}$ SOL

**Figure 7.4:** Friedman pairwise comparison of p-values for results with fees (Cont.).




**(a)** $Sample_{out}$ AGL

**(b)** $Sample_{out}$ ALSI

**(c)** $Sample_{out}$ BIL

**(d)** $Sample_{out}$ GFI

**(e)** $Sample_{out}$ IMP

**(f)** $Sample_{out}$ INV-NED

**(g)** $Sample_{out}$ INV-REM

**(h)** $Sample_{out}$ INV-SBK

**(i)** $Sample_{out}$ LON

**(j)** $Sample_{out}$ NED

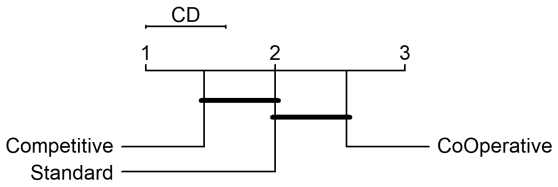
**(k)** $Sample_{out}$ RCH

**(l)** $Sample_{out}$ REM

**(m)** $Sample_{out}$ SAB

**(n)** $Sample_{out}$ SBK

**Figure 7.5:** Critical difference plots for results with fees.




**(o)** $Sample_{out}$ SOL

**Figure 7.5:** Critical difference plots for results with fees (Cont.).

## 7.4 Comparison to random strategies

Section 7.3 compared the results returned by the trading rules evolved using three different GP algorithms. This section compares the results returned by the evolved trading rules to a random trader in order to determine if the performance of evolved trading rules is due to chance. Three random trading rules were created.

The first random trading rule is referred to as the flip-flop trading strategy. On the first day of trade, the flip-flop strategy either buys a share or not. The decision to buy or not to buy is performed with equal probability. If the flip-flop strategy bought shares on the first day, it sells the shares on the second day. If the flip-flop strategy did not buy shares on the first day, it buys shares on the second day and sells them on the third day. The flip-flop trading rule results in a buy on one day, and a sell the day after, followed by a buy, and then a sell. The process continues until the last day of trade.

The second and third random trading rules are referred to as random walks. The random walk trading strategy uses a probability to determine if it should buy a share when it has none, or sell a share when it has shares. The decision to buy, sell, or hold shares is presented in Algorithm 11. The first random walk is aggressive, configured to use a probability $P_a$ of 50%, implying the aggressive random walk has a 50% chance of changing its position. The second random walk is conservative, configured with a probability of 30%. The conservative random walk has a 30% chance of buying shares when it has none, and 30% of selling the shares when it has shares.

**Algorithm 11:** Random walk algorithm.

```
Set boolean s = false;                          ▷ The trader has no shares on the first day
Set action probability P_a                      ▷ Configured probability
for d = startDay; d < endDay; d = d + 1; do
    Set random number r ~ U(0,1);
    if r < P_a then
        if s == false then
            Buy shares;
            s = true;
        else
            Sell shares;
            s = false;
        end if
    else
        Do nothing;
    end if
end for
Trading is over sell shares;
```

The EMH states that a market is efficient and therefore returning a profit greater than the market is difficult. The trading rules were compared to a buy-and-hold trading strategy to test the trading rules performance against the market's performance. A buy-and-hold strategy buys shares on the first day and sells them on the last day.

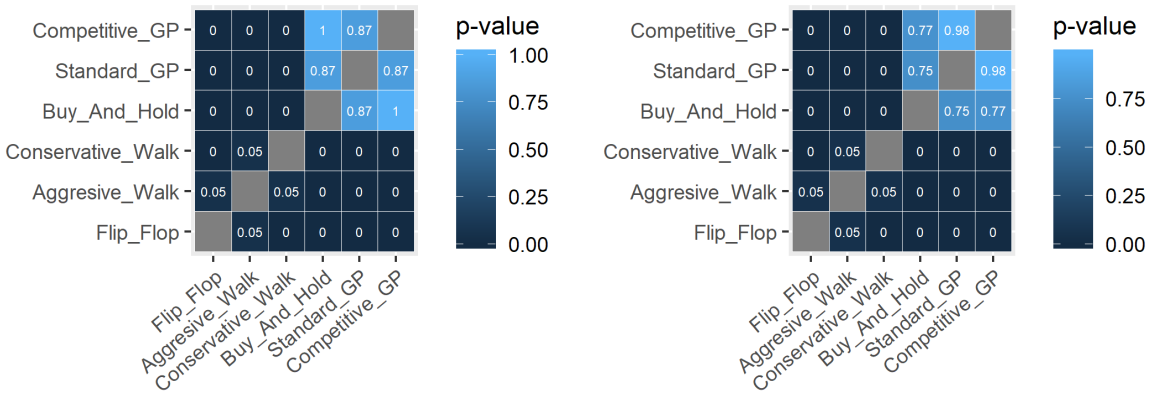
Four simulations were run, one for each of the random trading strategies. Each simulation was run 30 times independently with trading fees against each share. The mean profit returned by the random trading strategies are presented in Table 7.5. The results of random strategies were compared to the results returned by the trading rules evolved using the standard GP and the competitively co-evolved GP. Section 7.3 and Section 7.2 show that the results returned by the co-operative co-evolved trading rules performed significantly worse than the trading rules evolved using competitive co-evolution and single population evolution, and therefore the co-operative co-evolved GP results were excluded from this section. The co-evolved and evolved trading rule results are also presented in Table 7.5.

The Iman and Davenport [142] statistical test returned a p-value less than 0.00001 across all the result comparisons, showing that the results were significantly different and required post-hoc analysis. The results of the post-hoc analysis are presented in Figure 7.6. The post-hoc analysis, in conjunction

with the mean profit results, shows that the profit returned by the buy-and-hold strategy and the evolved trading rules were significantly greater than the profit returned by the random trading strategies. This shows that the performance of evolved trading rules is not due to random chance.

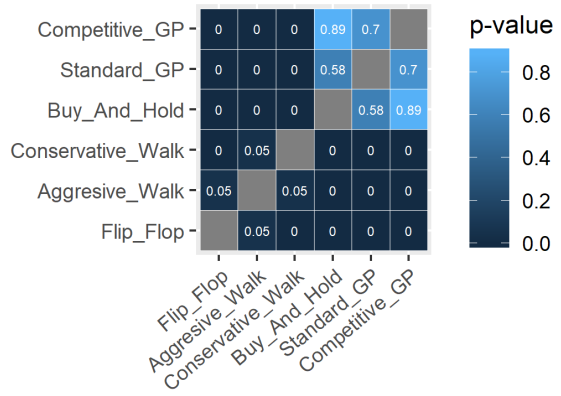
The GFI, INVNED, INVREM, and INVSBK profit results returned by the buy-and-hold strategy were significantly less than the results returned by the co-evolved and evolved trading rules. The NED profit results returned by the buy-and-hold strategy were significantly more than the results returned by the co-evolved and evolved trading rules. The evolved trading rules returned a positive trading balance at the end of trading INVNED, INVREM, and INVSBK, compared to the buy-and-hold strategy that returned a positive trading balance at the end of INVREM.

The results show that the trading rules evolved using either a competitive co-evolved GP or a standard GP can return a profit significantly greater than the buy-and-hold strategy if the shares are trending down, or are highly volatile. However, when the share price is trending upwards; the trading rules evolved using either a competitive co-evolved GP or a standard GP performed as well as the buy-and-hold strategy.

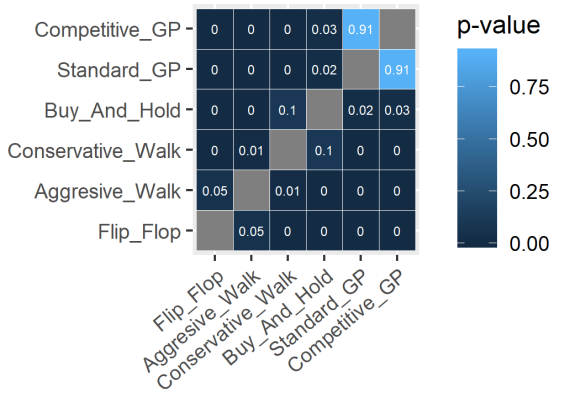



**(a)** $Sample_{out}$ AGL **(b)** $Sample_{out}$ ALSI

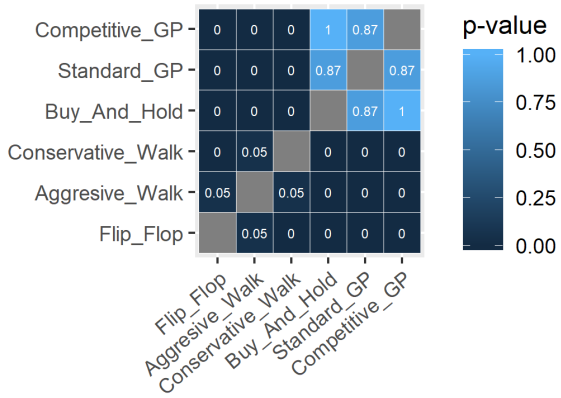
**Figure 7.6:** Friedman pairwise comparison of p-values for results with fees.



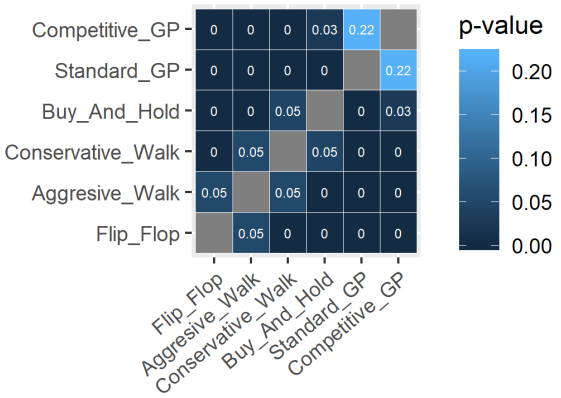

**(c)** $Sample_{out}$ BIL



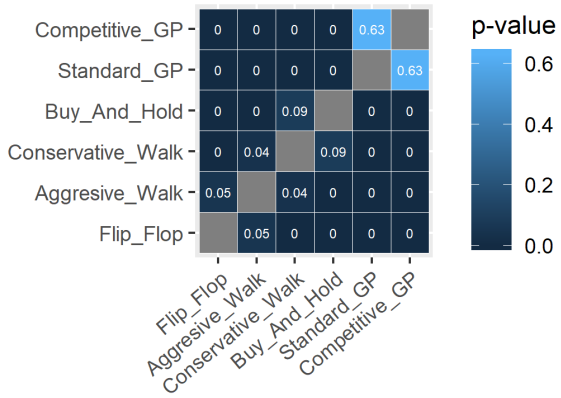

**(d)** $Sample_{out}$ GFI



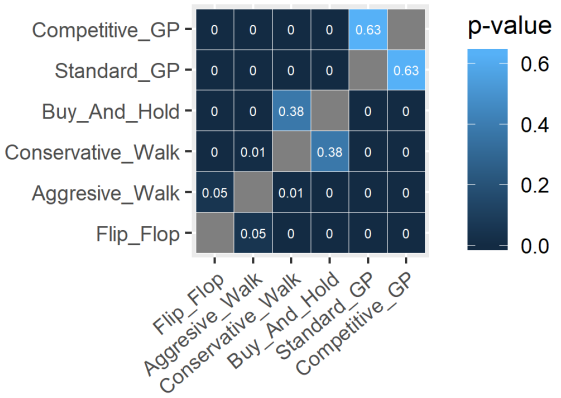

**(e)** $Sample_{out}$ IMP




**(f)** $Sample_{out}$ INV-NED

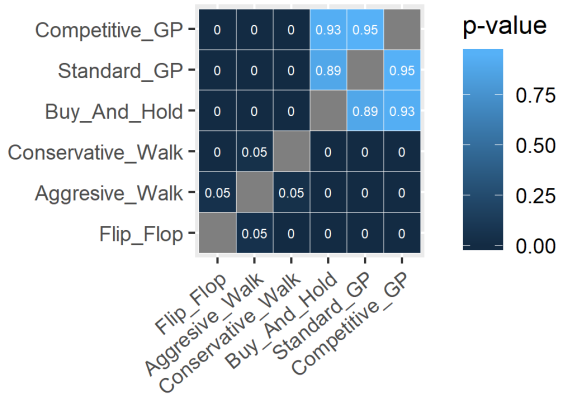



**(g)** $Sample_{out}$ INV-REM

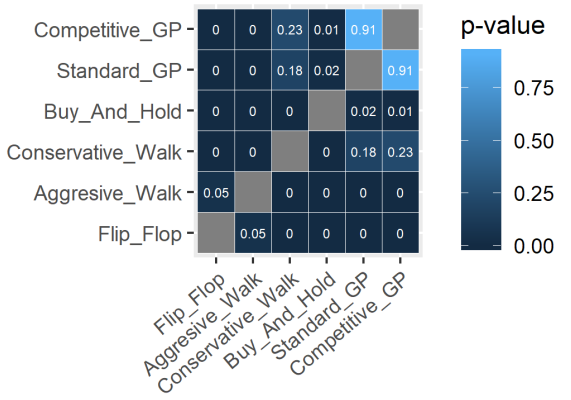



**(h)** $Sample_{out}$ INV-SBK

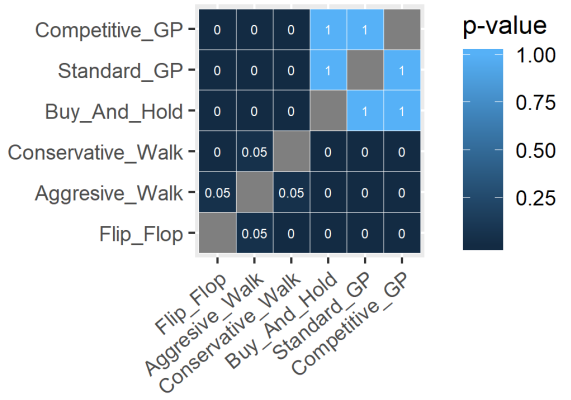
**Figure 7.6:** Friedman pairwise comparison of p-values for results with fees (Cont.).



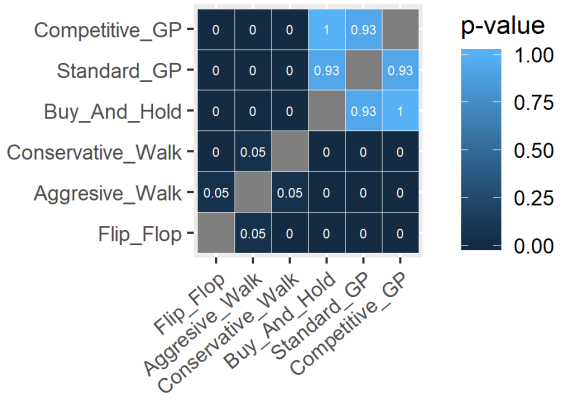

**(i)** $Sample_{out}$ LON



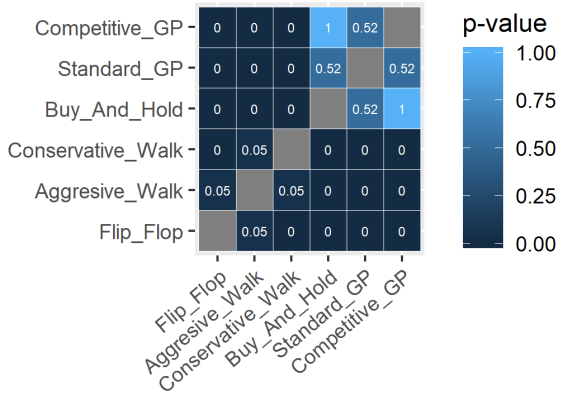

**(j)** $Sample_{out}$ NED



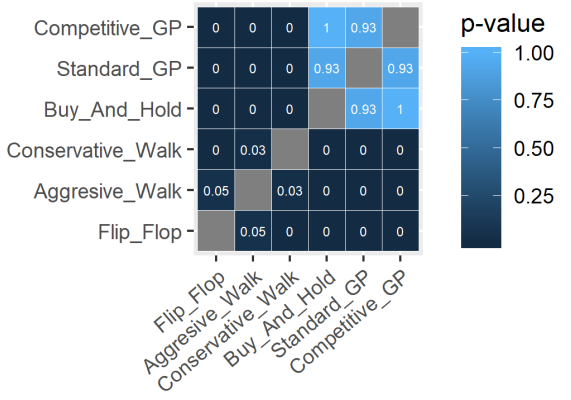

**(k)** $Sample_{out}$ RCH




**(l)** $Sample_{out}$ REM

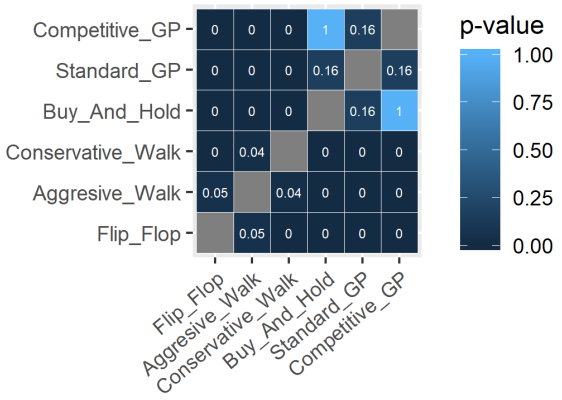



**(m)** $Sample_{out}$ SAB




**(n)** $Sample_{out}$ SBK

**Figure 7.6:** Friedman pairwise comparison of p-values for results with fees (Cont.).




**(o)** $Sample_{out}$ SOL

**Figure 7.6:** Friedman pairwise comparison of p-values for results with fees (Cont.).

**Table 7.5:** Mean $Sample_{out}$ profit per share with fees analysis.

| Share Code | Flip-Flop | Aggresive-Walk | Conservative-Walk | Buy-And-Hold | Standard-GP | Competitive-GP |
|---|---|---|---|---|---|---|
| **agl** | -1181250.1 | -545804.39 | -73380.09 | 238416.064 | 233796.066 | 238416.064 |
| **alsi** | -798813.6 | -368276.96 | -70352.65 | 92773.873 | 91128.609 | 91619.901 |
| **bil** | -530573.3 | -239004.29 | -17670.24 | 128804.507 | 123908.929 | 126006.385 |
| **gfi** | -449855.0 | -227971.68 | -71642.77 | -31688.221 | -12986.580 | -13982.185 |
| **imp** | -609825.0 | -281131.76 | -17836.25 | 172412.542 | 167401.685 | 172412.542 |
| **inv-ned** | -247838.6 | -117399.32 | -30503.31 | 13036.492 | 54847.899 | 47000.941 |
| **inv-rem** | -300960.2 | -159758.40 | -61094.00 | -45929.835 | 5271.031 | 10195.870 |
| **inv-sbk** | -198314.5 | -99682.58 | -40331.80 | -34220.852 | 20835.322 | 18103.539 |
| **lon** | -1475372.0 | -669354.48 | -110665.79 | 203049.046 | 196280.745 | 197359.673 |
| **ned** | -493363.5 | -229928.73 | -63381.78 | -6736.434 | -48931.842 | -53313.152 |
| **rch** | -140858.0 | -60216.61 | -11959.24 | 15436.255 | 15436.255 | 15436.255 |
| **rem** | -619371.6 | -264102.02 | -51334.02 | 67722.663 | 67225.703 | 67722.663 |
| **sab** | -508261.6 | -252189.20 | -53341.61 | 53030.657 | 48300.081 | 53030.657 |
| **sbk** | -334221.9 | -160191.85 | -45572.17 | 6219.866 | 6152.103 | 6219.866 |
| **sol** | -973212.5 | -411406.68 | -59980.31 | 142051.531 | 93724.734 | 142051.531 |

## 7.5 Summary

Three different GPs were compared with and without trading fees. It was found that a co-operative co-evolved GP produced trading rules that performed significantly worse than trading rules evolved using either a standard GP or a competitive co-evolved GP. This chapter compared the profitability of trading rules evolved using a standard GP and trading rules evolved using a competitively co-evolved GP to four random trading strategies on shares traded on the JSE. The next chapter discusses the findings of this thesis, and presents ideas for future work.

# Chapter 8

# Conclusions

*A conclusion is the place where you got tired of thinking*

*-Arthur Bloch (American Writer)*

This chapter briefly highlights the findings and contributions of this thesis and discusses directions for future research.

## 8.1 Summary of Conclusions

This thesis presented an empirical process to optimise and evaluate the single population genetic program (GP) algorithm developed by Allen and Karjalainen [43]. Parameter sensitivity experiments were run to determine the preferred parameters, while an empirical process showed that: rank selection was the preferred selection operator, and that a compounded excess return over the buy-and-hold strategy was the preferred fitness function.

Two co-evolution variations of the single population GP were proposed. The first variation was that of a co-operative co-evolved GP. The second variation was a competitive co-evolved GP. The three different GPs were run against 15 different share datasets. Each dataset was sub-divided into three different samples. An in-sample dataset used for evolving a generation. A selection or verification sample dataset used to validate the best individual, and an out-of-sample dataset to test the performance of the evolved rules on unseen data.

The GPs were run with and without transaction fees. The results of the different runs were compared using an empirical process. Post-hoc analysis found a significant difference between the results of the trading rules generated by the co-operative co-evolved GP and the other two GPs. The co-operative co-evolved trading rules performed significantly worse across all the shares, with and without fees than the other trading rules.

The results including transaction fees showed that the trading rules evolved by the competitively co-evolved GPs performed better, but not significantly better than the rules evolved by the standard single population GP, for 10 of the 15 shares. The results also showed that the competitively co-evolved GP evolved trading rules that performed significantly better than the trading rules evolved by the standard GP when trading SOL. However, trading rules evolved using the standard GP performed significantly better than trading rules evolved by the competitively co-evolved GP when trading INVNED.

The trading rules evolved by the competitive co-evolved GP and the standard GP were compared to three random trading rules. The results found that the evolved trading rules performed significantly better than randomly buying and selling shares. The trading rules were compared to the passive trading strategy, buy-and-hold. No significant difference was found between the results returned by the trading rules and the results returned by the buy-and-hold strategy for 10 of the 15 shares. The shares that showed significant difference were shares trending downwards. In these cases the evolved traders per-

formed significantly better than the buy-and-hold strategy. The trading rules significantly out-performed the buy-and-hold for 4 of the 15 different shares.

This thesis presented evidence showing that trading rules evolved using either a competitive co-evolved GP or a standard GP can return a profit significantly greater than the buy-and-hold strategy if the shares are trending down, or are highly volatile. However, when the share price is trending upwards, trading rules evolved using either a competitive co-evolved GP or a standard GP performed as well as the buy-and-hold strategy.

The thesis found that the competitive co-evolved GP produced more reliable trading rules and performed generally better than trading rules evolved using a single population GP.

## 8.2 Future Work

Several areas are identified for future research to improve the competitive co-evolved GP in stock market trading rule generation.

#### Improve the competitive co-evolved algorithm

This thesis implemented a simple competitive co-evolved GP. The algorithm used two populations and a simple bipartite relative fitness as defined by Hillis [79, 132, 185]. Future work could include the work of Rosin and Belew [79, 185] by studying the effect that competitive fitness sharing, shared sampling, and the hall-of-fame has on the performance of the evolved trading rules.

#### Classify training data

This thesis shows that the market conditions used during evolution affect the performance of a trading rule when exposed to the out-of-sample data. To evolve the trading rules, this thesis divided datasets into three distinct continuous samples of fixed sizes. A suggestion is to automatically classify continuous samples within the dataset, determining the trend of the sample. Then to combine continuous samples of varying trends into an in sample dataset and selection dataset. The intent is to expose the evolutionary process to share trends that evolve the best out-of-sample trading rules.

#### Include more datasets

Each trading rule uses the share data of the share it is trading. No additional market information is provided. Perhaps more information could be made available to the trading rules. For example, currency price, interest rate, weather data, season, day of the week, and, other share data from similar or other market sectors.

### Improve the mutation operator

The mutation operator used in this thesis implemented both a grow mutation operator and a prune mutation operator with equal probability. Future work could study the effect the probability has on the performance of evolved trading rules. Furthermore, the probability could change overtime. For example, while the population has a small average tree size, the mutation operator may favour growing the chromosome. When the population has a large average tree size the mutation operator may favour pruning the chromosomes.

### Implementation of a dynamic mutation probability

This thesis used a static mutation probability of 35%. The mutation probability was determined by an empirical analysis. The empirical analysis also showed that a mutation probability of 70% was preferred when evolving trading rules over the LON dataset. Future work could explore the impact that a dynamic mutation probability has on the profitability of evolved trading rules. The dynamic mutation probability could start high allowing the GP to explore the search space, and overtime the mutation probability could decrease, resulting in less exploration and more optimisation.

# Bibliography


[1] businesstech.co.za: South Africa repo vs prime rate: 2005-2015. https://businesstech.co.za/news/columns/110913/sa-repo-rate-vs-prime-rate-2005-2015/. (Accessed on 01/31/2017).

[2] history.com: War in Iraq begins. http://www.history.com/this-day-in-history/war-in-iraq-begins, Mar. 2003. (Accessed on 01/31/2017).

[3] cnn.com: South Africa gets 2010 World Cup. http://edition.cnn.com/2004/SPORT/football/05/15/worldcup.2010/, May 2004. (Accessed on 01/31/2017).

[4] bbc.co.uk: Madrid train attacks. http://news.bbc.co.uk/2/shared/spl/hi/guides/457000/457031/html/, June 2007. (Accessed on 01/31/2017).

[5] jse.co.za: South African stock exchange - history. http://www.jse.co.za, 2008. (Accessed on 06/12/2008).

[6] bbc.com: Ukraine's two different revolutions. http://www.bbc.com/news/world-europe-25210230, Dec. 2013. (Accessed on 01/31/2017).

[7] cnn.com: September 11th fast facts. http://edition.cnn.com/2013/07/27/us/september-11-anniversary-fast-facts/, July 2013. (Accessed on 01/31/2017).

[8] bbc.com: Lebanon profile. http://www.bbc.com/news/world-middle-east-14649284, Apr. 2016. (Accessed on 01/31/2017).

[9] angloamerican.com: Anglo American history. http://www.angloamerican.com/about-us/history, 2017. (Accessed on 01/31/2017).

[10] bhpbilliton.com: BHP Billiton - a leading global resources company. http://www.bhpbilliton.com/, 2017. (Accessed on 01/31/2017).

[11] cnn.com: Spain train bombings fast facts. http://edition.cnn.com/2013/11/04/world/europe/spain-train-bombings-fast-facts/, 2017. (Accessed on 01/31/2017).

[12] dictionary.com: Co-adaptation. http://www.dictionary.com/browse/coadaptation, 2017. (Accessed on 03/02/2018).

[13] European parliament: Exchange rates 2001-2005: Euro vs US Dollar. http://www.europarl.europa.eu/RegData/etudes/note/join/2007/379231/IPOL-TRAN_NT(2007)379231_EN.pdf, 2017. (Accessed on 01/31/2017).

[14] fin24.com: A history of great value for Remgro. http://www.fin24.com/Finweek/Investment/a-history-of-great-value-for-remgro-20160217, 2017. (Accessed on 01/31/2017).

[15] globalsecurity.org: Liberia - second civil war – 1997–2003. http://www.globalsecurity.org/military/world/war/liberia-1997.htm, 2017. (Accessed on 01/31/2017).

[16] golfields.co.za: Gold Fields - about Gold Fields. https://www.goldfields.co.za/au_main.php, 2017. (Accessed on 01/31/2017).

[17] implats.co.za: Company profile. http://www.implats.co.za/implats/Company-profile.asp, 2017. (Accessed on 01/31/2017).

[18] inflation.eu: Inflation South Africa 2008 - CPI inflation South Africa 2008. http://www.inflation.eu/inflation-rates/south-africa/historic-inflation/cpi-inflation-south-africa-2008.aspx, 2017. (Accessed on 01/31/2017).

[19] justice.gov.za: Truth and Reconciliation Commission. http://www.justice.gov.za/Trc/, 2017. (Accessed on 01/31/2017).

[20] lonmin.com: Lonmin home page. https://www.lonmin.com/, 2017. (Accessed on 01/31/2017).

[21] mg.co.za: Marikana: One year after the masacre. https://marikana.mg.co.za/, Oct. 2017.

[22] mining.com: Lonmin shares collapse as Glencore completes stake divestment. http://www.mining.com/lonmin-shares-collapse-as-glencore-completes-stake-divestment/, 2017. (Accessed on 01/31/2017).

[23] nedbank.co.za: About us. https://www.nedbank.co.za/content/nedbank/desktop/gt/en/aboutus.html, 2017. (Accessed on 01/31/2017).

[24] nedbank.co.za: Our history. https://www.nedbank.co.za/content/nedbank/desktop/gt/en/aboutus/about-nedbank-group/who-we-are/Our-history.html, 2017. (Accessed on 01/31/2017).

[25] onlygold.com: Historical gold prices. http://onlygold.com/Info/Historical-Gold-Prices.asp, 2017. (Accessed on 01/31/2017).

[26] oxforddictionary.com: Overfitting. https://en.oxforddictionaries.com/definition/overfitting, 2017. (Accessed on 03/02/2018).

[27] remgro.com: Company history. http://www.remgro.com/about-remgro/history/, 2017. (Accessed on 01/31/2017).

[28] richemont.com: Richemont home page. https://www.richemont.com/, 2017. (Accessed on 01/31/2017).

[29] sahistory.org.za: Thabo Mbeki (1942 - ) timeline. http://www.sahistory.org.za/article/thabo-mbeki-1942-timeline, 2017. (Accessed on 01/31/2017).

[30] sasol.com: Sasol home page. http://www.sasol.com/, 2017. (Accessed on 01/31/2017).

[31] standardbank.co.za: AutoShare investment account. http://www.standardbank.co.za/standardbank/Personal/Banking/Savings-and-investments/Auto-Share-Invest, 2017. (Accessed on 02/07/2017).

[32] standardbank.co.za: Standard Bank trading fees. https://securities.standardbank.co.za/ost/nsp/BrochureWarepublic/Ost/fees.html, 2017. (Accessed on 01/31/2017).

[33] taxonomy.zoology.gla.ac.uk: Cospeciation. http://taxonomy.zoology.gla.ac.uk/ToL/Phthiraptera/links/cospeciation.html, 2017. (Accessed on 03/02/2018).

[34] wikipedia.org: 2000's energy crisis. https://en.wikipedia.org/w/index.php?title=2000s_energy_crisis&oldid=760226749, 2017. (Accessed on 01/31/2017).

[35] wikipedia.org: Markov chain. https://en.wikipedia.org/w/index.php?title=Markov_chain&oldid=832049521, 2017. [Online; accessed 26-March-2017].

[36] wikipedia.org: Non-deterministic Turing machine. https://en.wikipedia.org/wiki/Non-deterministic_Turing_machine, 2017. (Accessed on 02/11/2017).

[37] wikipedia.org: SABMiller. https://en.wikipedia.org/wiki/SABMiller, 2017. (Accessed on 01/31/2017).

[38] wikipedia.org: Sasol. https://en.wikipedia.org/wiki/Sasol, 2017. (Accessed on 01/31/2017).

[39] wikipedia.org: Standard Bank. https://en.wikipedia.org/wiki/Standard_Bank, 2017. (Accessed on 01/31/2017).

[40] wikipedia.org: Central limit theorem. https://en.wikipedia.org/w/index.php?title=Central_limit_theorem&oldid=859263874, 2018. (Accessed on 03/02/2018).

[41] wikipedia.org: Empedocles. https://en.wikipedia.org/w/index.php?title=Empedocles&oldid=841259049, 2018. (Accessed on 03/02/2018).

[42] S. B. Achelis. *Technical analysis from A to Z*. McGraw-Hill Education, second edition, Dec. 2013.

[43] F. Allen and R. Karjalainen. Using genetic algorithms to find technical trading rules. *Rodney L. White Center for Financial Research*, pages 20–95, Oct. 1995.

[44] F. Allen and R. Karjalainen. Using genetic algorithms to find technical trading rules1. *Journal of financial Economics*, 51(2):245–271, Feb. 1999.

[45] B. Andersen, J. McDonnell, and W. Page. Configuration optimisation of mobile manipulators with equality constraints using evolutionary programming. In *First Annual Conference on Evolutionary Programming*, pages 71–91, Feb. 1993.

[46] P. J. Angeline. Subtree crossover: Building block engine or macromutation. In *Genetic programming 1997: Proceedings of the second annual conference*, volume 97, pages 9–17. Morgan Kaufmann, July 1997.

[47] P. J. Angeline and J. B. Pollack. Competitive environments evolve better solutions for complex tasks. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann, 1993.

[48] G. Arnold. *The Financial Times guide to investing*. Prentice Hall, 2004.

[49] J. W. Atmar III. *Speculation on the evolution of intelligence and its possible realisation in machine form*. PhD thesis, New Mexico State University, 1976.

[50] R. Axelrod and W. D. Hamilton. The evolution of co-operation. *Science journal*, 211(4489):1390–1396, 1981.

[51] R. M. Axelrod. *The evolution of co-operation*. Basic books, member of the Perseus Books Group, revised edition, 2006.

[52] T. Bäck. *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Jan. 1996.

[53] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation*. IOP Publishing Ltd, first edition, 1997.

[54] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE transactions on evolutionary computation*, 1(1):3–17, Apr. 1997.

[55] T. Bäck, F. Hoffmeister, and H.-P. Schwefel. A survey of evolution strategies. In *Proceedings of the fourth international conference on genetic algorithms*, volume 2, pages 2–9. Morgan Kaufmann, June 1991.

[56] T. Bäck, G. Rudolph, and H.-P. Schwefel. Evolutionary programming and evolution strategies: Similarities and differences. In *In Proceedings of the second annual conference on evolutionary programming*, pages 11–22, 1993.

[57] T. Bäck and M. Schütz. Evolution strategies for mixed-integer optimisation of optical multilayer systems. In *Proceedings of the 4th Annual conference of Evolutionary Programming*, volume EP-95, pages 33–51. MIT Press, Oct. 1995.

[58] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimisation. *Evolutionary computation*, 1(1):1–23, Mar. 1993.

[59] T. Bäck and H.-P. Schwefel. Evolutionary computation: An overview. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 20–29. IEEE, May 1996.

[60] J. D. Bagley. The behaviour of adaptive systems which employ genetic and correlation algorithms. Technical report, University of Michigan, College of Literature, Science and the Arts, Department of Computer Science, 1967.

[61] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the second international conference on genetic algorithms*, volume 206, pages 14–21, 1987.

[62] N. A. Barricelli. Numerical testing of evolution theories. *Acta Biotheoretica*, 16(3-4):99–126, Mar. 1963.

[63] R. J. Bauer. *Genetic algorithms and investment strategies*, volume 19. John Wiley & Sons, 1994.

[64] R. J. Bauer. Genetic algorithms and the management of exchange rate risk, 1995.

[65] R. J. Bauer and J. R. Dahlquist. *Technical markets indicators: Analysis & performance*, volume 64. John Wiley & Sons, 1999.

[66] R. J. Bauer and G. E. Liepins. *Genetic algorithms and computerized trading strategies*, volume 5. Research and Publications, School of Business Administration, University of Western Ontario, 1992.

[67] R. J. Bauer Jr. An introduction to genetic algorithms: A mutual fund screening example. *Neurovest journal*, 2(4):16–19, 1994.

[68] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural computing*, 1(1):3–52, 2002.

[69] G. W. J. Bishop. Charles H. Dow and the Dow Theory. *The economic history review*, 13(3):520–521, 1961.

[70] A. G. Bluman. *Elementary statistics*. Brown Melbourne, 1995.

[71] J. Bollinger. *Bollinger on Bollinger bands*. McGraw Hill Professional, 2001.

[72] M. D. Bordo. An historical perspective on the crisis of 2007-2008. Technical report, National Bureau of Economic Research, 2008.

[73] G. E. P. Box and K. B. Wilson. On the experimental attainment of optimum conditions. In *Breakthroughs in statistics*, pages 270–310. Springer, 1992.

[74] H. J. Bremermann. Optimisation through evolution and re-combination. *Self-organizing systems*, 93:106, 1962.

[75] M. K. Brunnermeier. Deciphering the liquidity and credit crunch 2007–2008. *The journal of economic perspectives*, 23(1):77–100, 2009.

[76] G. L. Buffon. *Histoire naturelle générale et particulière*, volume 18. de l'Imprimerie de F. Dufart, 1785.

[77] G. Burgin. On playing two-person zero-sum games against non-minimax players. *IEEE transactions on systems science and cybernetics*, 5(4):369–370, 1969.

[78] B. Calvo and R. G. Santafé. SCMAMP: Statistical comparison of multiple algorithms in multiple problems. *The R Journal*, 8/1(RJ-2016-017):248–256, Aug. 2016.

[79] N. Casas. A review of landmark articles in the field of co-evolutionary computing. *arXiv*, pages 1–5, June 2015.

[80] D. J. Cavicchio. Adaptive search using simulated evolution. Technical report, University of Michigan, College of Literature, Science and the Arts, Department of Computer Science, 1970.

[81] R. E. Charles. Anaximander, earliest precursor of Darwin. *Popular Science Monthly*, 67(1):701–706, Dec. 1905.

[82] S.-H. Chen and C.-H. Yeh. Toward a computable approach to the efficient market hypothesis: An application of genetic programming. *Journal of economic dynamics and control*, 21(6):1043–1063, 1997.

[83] D. Cliff and G. F. Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In *European Conference on Artificial Life*, pages 200–218. Springer, 1995.

[84] R. Cluver. *Investment without tears*. O. Burgess, 1988.

[85] A. Cowles. Can stock market forecasters forecast. *Econometrica*, 1(3):309–324, 1933.

[86] N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 183–187, 1985.

[87] G. B. Cuvier. *Le règne animal distribué d'après son organisation: Pour servir de base à l'histoire naturelle des animaux et d'introduction à l'anatomie comparée*, volume 1. Hauman et compe., 1836.

[88] K. W. Dam. The sub-prime crisis and financial regulation: International and comparative perspectives. *Chicago journal of international law*, 10(2):1, Mar. 2010.

[89] C. Darwin. *On the origin of the species by means of natural selection: Or, the preservation of favoured races in the struggle for life*. John Murray, 1869.

[90] E. Darwin. *Zoonomia; or, The laws of organic life...*, volume 2. by D. Carlisle, for Thomas and Andrews, 1803.

[91] K. A. De Jong. *Analysis of the behaviour of a class of genetic adaptive systems*. phdthesis, Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975.

[92] M. de la Maza. A seagul visits the race track. In *Proceedings of the third international conference on Genetic algorithms*, pages 208–212. Morgan Kaufmann Publishers Inc., 1989.

[93] E. B. De Lima, G. L. Pappa, J. M. de Almeida, M. A. Gonçalves, and W. Meira. Tuning genetic programming parameters with factorial designs. In *Congress on Evolutionary Computation*, pages 1–8. IEEE, July 2010.

[94] J. B. P. A. de Monet et al. *Philosophie zoologique: Ou exposition des considérations relatives a l'histoire naturelle des animaux...*, volume 1. F. Savy, 1873.

[95] D. W. Dearholt. Some experiments on generalisation using evolving automata. In *Proceedings of the 9th international conference. on system sciences*, pages 131–13, 1976.

[96] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of machine learning research*, 7:1–30, Jan. 2006.

[97] M. Dianati, I. Song, and M. Treiber. An introduction to genetic algorithms and evolution strategies. Technical report, University of Waterloo, Ontario, N2L 3G1, Canada, 2002.

[98] R. Dreżewski, J. Sepielak, and L. Siwik. Generating robust investment strategies with agent-based co-evolutionary system. In *International conference on computational science*, pages 664–673. Springer, 2008.

[99] O. J. Dunn. Multiple comparisons among means. *Journal of the American statistical association*, 56(293):52–64, 1961.

[100] C. R. Eastman. *Anaximander, earliest precursor of Darwin*. Harvard University, 1905.

[101] R. D. Edwards, J. Magee, and W. H. C. Bassetti. *Technical analysis of stock trends*. Taylor & Francis, ninth edition, 2007.

[102] M. E. El-Telbany. The Egyptian stock market return prediction: A genetic programming approach. In *Electrical, electronic and computer engineering, 2004. ICEEC '04. 2004 international conference on*, pages 161–164, Sept. 2004.

[103] A. P. Engelbrecht. *Computational intelligence: An introduction*. Wiley, 2007.

[104] S. S. Epp. *Discrete mathematics with applications*. Cengage Learning, forth international edition, 2010.

[105] B. Erten and J. A. Ocampo. Super cycles of commodity prices since the mid-nineteenth century. *United Nations Department of Economic and Social Affairs*, 44(110):14–30, Feb. 2012.

[106] E. F. Fama. The behaviour of stock market prices. *The journal of business*, 38(1):34–105, 1965.

[107] E. F. Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.

[108] H. Finner. On a monotonicity problem in step-down multiple test procedures. *Journal of the American statistical association*, 88(423):920–923, 1993.

[109] D. B. Fogel. Applying evolutionary programming to selected travelling salesman problems. *Cybernetics and systems*, 24(1):27–36, 1993.

[110] D. B. Fogel. In memoriam Alex S. Fraser [1923-2002]. *IEEE Transactions on Evolutionary Computation*, 6(5):429–430, 2002.

[111] D. B. Fogel. *Evolutionary computation: Toward a new philosophy of machine intelligence*. IEEE Press Series on Computational Intelligence. Wiley, 2006.

[112] D. B. Fogel. Foundations of evolutionary computing. In *Modeling and Simulation for Military Applications*, pages 1–13, May 2006.

[113] D. B. Fogel. Nils Barricelli-artificial life, co-evolution, self-adaptation. *IEEE computational intelligence magazine*, 1(1):41–45, 2006.

[114] L. J. Fogel. On the organisation of intellect, 1964.

[115] L. J. Fogel, P. J. Angeline, and D. B. Fogel. Approach to self-adaptation on finite state machines. In *Evolutionary programming IV: Proceedings of the fourth annual conference on evolutionary programming*, volume 355. MIT Press, 1995.

[116] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial intelligence through simulated evolution*. John Wiley, 1966.

[117] R. Forsyth. BEAGLE - A Darwinian approach to pattern recognition. *Kybernetes*, 10(3):159–166, 1981.

[118] D. R. Frantz. Non-linearities in genetic adaptive search. Technical report, Department of Health, Education, and Welfare, 1972.

[119] A. Fraser and D. Burnell. *Computer models in genetics*. Mcgraw-Hill Book Co., New York., 1970.

[120] A. S. Fraser. Simulation of genetic systems by automatic digital computers I. Introduction. *Australian Journal of Biological Sciences*, 10(4):484–491, 1957.

[121] G. J. Friedman. *Selective feedback computers for engineering synthesis and nervous system analogy*. Master's thesis, UCLA, 1956.

[122] S. García, A. Fernández, J. Luengo, and F. Herrera. A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability. *Computational Intelligence and Complexity*, 13(10):959, Dec. 2008.

[123] S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information sciences*, 180(10):2044–2064, May 2010.

[124] S. Garcia and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of machine learning research*, 9(Dec):2677–2694, 2008.

[125] A. Ghandar, Z. Michalewicz, M. Schmidt, T.-D. Tô, and R. Zurbrugg. Computational intelligence for evolving trading rules. *IEEE transactions on evolutionary computation*, 13(1):71–86, 2009.

[126] D. E. Golberg. *Genetic algorithms in search, optimisation, and machine learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[127] C. Gordon. Empedocles. https://www.iep.utm.edu/empedocl, 2017. (Accessed on 03/02/2018).

[128] M. Graczyk, T. Lasota, Z. Telec, and B. Trawiński. Non-parametric statistical analysis of machine learning algorithms for regression problems. In *International conference on knowledge-based and intelligent information and engineering systems*, volume 6276 of *Lecture Notes in Computer Science*, pages 111–120. Springer, Berlin, Heidelberg, 2010.

[129] J. Grefenstette and R. Daley. Methods for competitive and co-operative co-evolution. In *Adaptation, co-evolution and learning in multiagent systems: Papers from the 1996 AAAI spring symposium*, pages 45–50, 1996.

[130] T. Hellstrom and K. Holmstrom. *Predicting the stock market*. PhD thesis, Malardalen University, department of mathematics and physics, Vasteras, Sweden, 1998.

[131] P. M. Hildebrand. The sub-prime crisis: A central banker's perspective. *Journal of Financial Stability*, 4(4):313–320, 2008.

[132] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimisation procedure. *Physica D: Nonlinear Phenomena*, 42(1-3):228–234, 1990.

[133] Y. Hochberg. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802, 1988.

[134] B. S. Holland and M. D. Copenhaver. An improved sequentially rejective Bonferroni test procedure. *Biometrics*, pages 417–423, 1987.

[135] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[136] J. H. Holland. *Hidden order: How adaptation builds complexity*. Basic Books, 1995.

[137] J. H. Holland. *Emergence: From chaos to order*. OUP Oxford, 2000.

[138] J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. *SIGART Bull*, 1(63):49–49, June 1977.

[139] R. B. Hollstein. *Artificial genetic adaptation in computer control systems*. PhD thesis, University of Michigan, 1971.

[140] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[141] H. I. and T. S. Using genetic programming to predict financial data. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the congress of evolutionary computation*, volume 1, pages 244–251, Mayflower Hotel, Washington D.C., USA, July 1999. IEEE Press.

[142] R. L. Iman and J. M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in statistics: Theory and methods*, 9(6):571–595, 1980.

[143] F. E. James. Monthly moving averages an effective investment tool. *Journal of financial and quantitative analysis*, 3(3):315–326, 1968.

[144] H. John. Holland. Properties of the bucket brigade. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 1–7, 1985.

[145] T. Jones et al. Crossover, macromutation, and population-based search. In *Proceedings of the sixth international conference on genetic algorithms*, pages 73–80, 1995.

[146] M. A. Kaboudan. Genetic programming prediction of stock prices. *Computational economics*, 16(3):207–236, 2000.

[147] W. I. King. Technical methods of forecasting stock prices. *Journal of the American statistical association*, 29(187):323–325, 1934.

[148] J. R. Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*, volume 34. Stanford University, department of computer science Stanford, CA, 1990.

[149] J. R. Koza. *Genetic programming II, automatic discovery of reusable subprograms*. MIT Press, Cambridge, MA, 1992.

[150] J. R. Koza. *Genetic programming: On the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[151] J. R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

[152] J. R. Koza, F. H. Bennett, and O. Stiffelman. Genetic programming as a Darwinian invention machine. In *European conference on genetic programming*, pages 93–108. Springer, 1999.

[153] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic programming IV: Routine human-competitive machine intelligence*, volume 5. Springer science & business media, 2006.

[154] S. S. Lam. A genetic fuzzy expert system for stock market timing. In *Evolutionary computation, 2001. Proceedings of the 2001 congress on*, volume 1, pages 410–417. IEEE, 2001.

[155] W. B. Langdon. *Genetic programming and data structures*. PhD thesis, University College London, 1996.

[156] R. W. Larsen and J. S. Herman. A comparison of evolutionary programming to neural networks and an application of evolutionary programming to a navy mission planning problem. In *Proceedings of the 1st annual conference on evolutionary programming (La Jolla, CA, 1992)*, pages 127–133, 1992.

[157] K. Lesetja. Lesetja Kganyago: Speech. https://www.resbank.co.za/lists/speeches/attachments/337/speech_lesetja%20kganyago.pdf, Mar. 2012. (Accessed on 01/31/2017).

[158] A. R. Luca. Post apartheid South Africa: the first ten years; Chapter 12: Bringing inflation under control. https://www.imf.org/external/pubs/nft/2006/soafrica/eng/pasoafr/sach12.pdf, Jan. 2006. (Accessed on 01/31/2017).

[159] B. E. Lutter and R. C. Huntsinger. Engineering applications of finite automata. *Transactions of the society for computer simulation*, 13(1):5–11, 1969.

[160] S. Mahfoud and G. Mani. Financial forecasting using genetic algorithms. *Applied artificial intelligence*, 10(6):543–566, Dec. 1996.

[161] G. Mani, K.-K. Quah, S. Mahfoud, and D. Barr. An analysis of neural-network forecasts from a large-scale, real-world stock selection system. In *Proceedings of 1995 Conference on Computational Intelligence for Financial Engineering (CIFEr)*, pages 72–78. IEEE, Apr. 1995.

[162] B. R. Marshall, M. R. Young, and L. C. Rose. Candlestick technical trading strategies: Can they create value for investors? *Journal of banking & finance*, 30(8):2303–2323, 2006.

[163] J. McCarthy. History of LISP. In *History of programming languages I*, pages 173–185. ACM, 1978.

[164] J. R. McDonnell, B. L. Andersen, W. C. Page, and F. G. Pin. Mobile manipulator configuration optimisation using evolutionary programming. Technical report, Oak Ridge National Lab., TN (United States), 1992.

[165] R. McDonnell, J and D. Waagen. Determining neural network connectivity using evolutionary programming. *Conference Record of the Twenty-Sixth Asilomar Conference on Signals, Systems & Computers*, 2(26):786–790, Oct. 1992.

[166] S. W. McNatton. HOBBES: A predicting expert system for thoroughbred horse racing, 1994.

[167] G. Mendel. Experiments in plant hybridisation (1865). *Verhandlungen des naturforschenden Vereins Brünn.*, Mar. 1996.

[168] H. Morgan, T, C. B. Bridges, and A. H. Sturtevant. The genetics of Drosophila Melanogaster. *Biblphia genet*, 2:1–262, 1925.

[169] P. B. Myszkowski and Ł. Rachwalski. Trading rule discovery on Warsaw Stock Exchange using revolutionary algorithms. In *Computer science and information technology, 2009. IMCSIT'09. International multi-conference on*, pages 81–88. IEEE, 2009.

[170] C. Neely, P. Weller, and R. Dittmar. Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *Journal of financial and Quantitative Analysis*, 32(4):405–426, 1997.

[171] J. F. Nicholls, K. M. Malan, and A. P. Engelbrecht. Evaluation of fitness functions for evolved stock market forecasting. In *The 7th international conference on computational intelligence In economics and finance (CIEF), 2008*. AI-ECON, 2008.

[172] W. C. Page, J. R. McDonnell, and B. Anderson. An evolutionary programming approach to multi-dimensional path planning. In *Proceedings of the first annual conference on evolutionary programming*, pages 63–70. Evolutionary programming society, La Jolla, CA, 1992.

[173] N. Peterson. *Wall street lingo: Thousands of investment terms explained simply*. Atlantic Publishing Group, 2007.

[174] S. Phelps, S. Parsons, P. McBurney, and E. Sklar. Co-evolution of auction mechanisms and trading strategies: Towards a novel approach to microeconomic design. In *In GECCO-02 Workshop on Evolutionary Computation in Multi-Agent Systems*, 2002.

[175] J. G. Pigeon. Statistics for experimenters: Design, innovation and discovery, 2006.

[176] V. W. Porto and D. B. Fogel. Alternative methods for training neural networks. In *Proceedings of the 1st annual conference on evolutionary programming (La Jolla, CA, 1992)*, pages 100–10, 1992.

[177] J. Potvina, P. Sorianoa, and M. Vall. Generating trading rules on the stock markets with genetic programming. *Computers & Operations Research*, 31(7):1033–1047, 2004.

[178] K. Prabhakaran and S. Nagarajan. Effectiveness of technical indicators: A study on CNX IT Indices. *Journal of Management and Science*, 2(2):11, 2012.

[179] T. C. Price. Using co-evolutionary programming to simulate strategic behaviour in markets. *Journal of Evolutionary Economics*, 7(3):219–254, 1997.

[180] M. Pring. *Candlesticks explained*. Martin J. Pring on technical analysis series. McGraw-Hill Companies,Incorporated, June 2002.

[181] N. Prize. *The Nobel prize in physiology or medicine*. nobelprize.org, 1933.

[182] R. Rhea. *The Dow Theory: An explanation of its development and an attempt to define its usefulness as an aid in speculation*. Fraser Publishing Company, 1993.

[183] D. M. Rom. A sequentially rejective test procedure based on a modified Bonferroni inequality. *Biometrika*, 77(3):663–665, Sept. 1990.

[184] R. S. Rosenberg et al. Simulation of genetic populations with biochemical properties: I. The model. *Mathematical biosciences*, 7:223–257, 1970.

[185] C. D. Rosin and R. K. Belew. New methods for competitive co-evolution. *Evolutionary computation*, 5(1):1–29, 1997.

[186] S. L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data mining and knowledge discovery*, 1(3):317–328, 1997.

[187] E. G. Schaefer. *How I helped more than 10,000 investors to profit in stocks*. Prentice-Hall, 1960.

[188] C. Schoreels and J. M. Garibaldi. A comparison of adaptive and static agents in equity market trading. *IEEE/WIC/ACM international conference on intelligent agent technology*, pages 393–399, Sept. 2005.

[189] C. Schoreels and J. M. Garibaldi. The effect of varying parameters on performance for adaptive agents in technical equity market trading. *IEEE 3rd international conference on computational cybernetics*, pages 243–248, Apr. 2005.

[190] C. Schoreels, B. Logan, and J. M. Garibaldi. Agent based genetic algorithm employing financial technical analysis for making trading decisions using historical equity market data. *Proceedings. IEEE/WIC/ACM international conference on intelligent agent technology, 2004. (IAT 2004)*, pages 421–424, Sept. 2004.

[191] H.-P. P. Schwefel. *Evolution and optimum seeking*, volume 4 of *Sixth Generation Computer Technologies*. John Wiley & Sons, Inc., first edition, Jan. 1993.

[192] A. V. Sebald and D. B. Fogel. Design of fault tolerant neural networks for pattern classification. *Fogel and Atmar*, 684:90–99, 1992.

[193] M. Seshadri. *Comprehensibility, overfitting and co-evolution in genetic programming for technical trading rules*. PhD thesis, Worcester Polytechnic Institute, 2003.

[194] M. Shahsavar, A. A. Najafi, and S. T. A. Niaki. Statistical design of genetic algorithms for combinatorial optimization problems. *Mathematical Problems in Engineering*, 2011(872415):17, July 2011.

[195] D. J. Sheskin. *Handbook of parametric and non-parametric statistical procedures*. crc Press, 2003.

[196] A. Smith and A. E. Smith. *Penalty functions*. Handbook of Evolutionary Computation. Oxford University Press and Institute of Physics Publishing, 1997. Section C 5.2.

[197] J. D. Smith. The many talents of the trading channel index. *AIQ, Opening Bell*, 5(9):2–5, 1996.

[198] J. M. Smith. The theory of games and the evolution of animal conflicts. *Journal of theoretical biology*, 47(1):209–221, 1974.

[199] J. M. Smith. Evolution and the theory of games. In *Did Darwin get it right?*, pages 202–215. Springer, 1988.

[200] J. M. Smith and G. R. Price. The logic of animal conflict. *Nature*, 246(5427):15, Nov. 1973.

[201] F. Streichert. Introduction to evolutionary algorithms. In *Frankfurt MathFinance Workshop*, pages 1–21, Apr. 2002.

[202] A. Takeuchi. Evolutionary automata-comparison of automaton behaviour and Restle's learning model. *Information sciences*, 20(2):91–99, 1980.

[203] G. Tilkin. MACD divergences. *Active trader*, pages 2–4, June 2001.

[204] E. Tsang. Forecasting where computational intelligence meets the stock market. *Frontiers of computer science in China*, 3(1):53–63, 2009.

[205] E. P. K. Tsang, J. Li, and J. M. Butler. EDDIE beats the bookies. *Softw., Pract. Exper.*, 28(10):1033–1043, 1998.

[206] W. W. H. Tsang, T. T. L. Chong, et al. Profitability of the on-balance volume indicator. *Economics bulletin*, 29(3):2424–2431, 2009.

[207] D. Van Reybrouck. *From primitives to primates: A history of ethnographic and primatological analogies in the study of prehistory*. Sidestone Press Dissertations, 2012.

[208] A. R. Wallace. *The Malay Archipelago: The land of the orang-utan and the bird of paradise; a narrative of travel, with studies of man and nature*. Courier Corporation, 1869.

[209] A. Weismann. *The germ-plasm: A theory of heredity*. Scribner's, 1893.

[210] G. Wilson and W. Banzhaf. Fast and effective predictability filters for stock price series using linear genetic programming. In *Congress on Evolutionary Computation*, pages 1–8. IEEE, July 2010.

[211] B. Wolfgang, N. Peter, K. Robert, and F. Frank. *Genetic programming: An introduction: On the automatic evolution of computer programs and its applications*. Morgan Kaufmann, 1998.

[212] E. O. Wright and J. Rogers. *American society: How it really works*. WW Norton & Company, second edition, Feb. 2015.

# Appendix A

# Appendix 1 - Stock Data


## Abstract

This appendix is supplementary to the data introduction presented in Section 5.2. A description of each share is presented in Section A.1.


## A.1 Stock Data

Data for the experiments was sourced from Sharenet. Sharenet is a market statistics company based in Cape Town, South Africa. The data used covers the period April 2003 to June 2008. This section begins with an overview of the financial market during the period 2003 to 2008. The sub sections that follow cover each of the shares used in this study, ending with a sub section on inverted shares.

### A.1.1 Overview of the financial market 2003 to 2008

The selection of shares not only covers different sectors, but different market conditions: In 2003, the then president, Mr. Thabo Mbeki [29], was running for his second term, the Truth and Reconciliation Commission [19] released its final report, and South Africans in general were looking forward to economic prosperity. The year 2003 saw the second invasion of Iraq [2]; this saw investors running to safe havens such as gold, putting pressure on the American dollar [13]. In 2000, gold traded at $272.65 and by 2004 it was at $435.60 [13, 25].

The period 2004 to 2006 saw an increase in world attacks with the Madrid [11] and UK train bombings [4], revolutions throughout eastern Europe [6], political change in Liberia [15] and Lebanon

[8], and more bombings in Mumbai. The world had changed from a positive outlook to a negative one, marked by the fall of the twin towers in New York in September of 2001 [7].

While the rest of the world experienced the pressure of war, the rising cost of living and oil [34], South Africa felt immune lagging behind the rest of the world. On 15 May 2004 South Africa was selected as the host of the football World Cup [3], promising infrastructure upgrades, and greater investment. By 2006, South Africa was experiencing their lowest interest rates in years [1]. All the upbeat sentiment, low interest rates, and government spending on infrastructure saw a rise in consumer spending.

The pressure of resources, the diminishing Rand value, and the appetite of South African consumers, all drove consumer inflation to new heights. June 2008 saw the consumer price inflation index (CPIX) at a high of 11.1% [18]. As prices increased, so did consumer's demand for credit [157]. With South Africa's neighbour, Zimbabwe, experiencing runaway inflation levels never seen before in any country, the South African Government needed to control inflation in South Africa. Monetary policy wanted inflation contained within 3-6% [157]. The South African Reserve Bank Governor, Tito Mboweni, started raising the interest rates in 2006 to curb consumer spending. Interest rate hikes put consumers into more debt, causing an increase in bank repossessions — ultimately a loss to the bank [157].

Towards the end of 2007 the western world, including South Africa, experienced the same increase in property prices, and an increase in demand for credit. Consumerism was at an all-time high [212]. Financial markets were pushed on the back of borrowed money and borrowed time. Ultimately, in 2008 the deck of cards came crashing down. The world market experienced a bull run on resources, known as the commodity super cycle [105]. The American economy suffered at the hands of the sub-prime crisis [131], causing the American housing bubble to burst. Barely hanging on, the Federal Reserve Chairman, Ben Bernanke, cut the American interest rates [75, 131], which in turn caused the American dollar to fall. Investors moved their money out of the American dollar and into gold. The sub-prime crisis caused worried investors to retreat from financial shares [75]. The ripple effects were felt throughout the world. Emerging markets, such as China and India, required more resources for their manufacturing industries. Investors that used to play in the financial market pushed money into resources stock, pushing resources higher. The sub-prime forced banks to cut-off credit. No credit meant no spending [72, 88].

Two sectors of the South African economy experienced two different cycles. The resources sector driven by emerging markets experienced a bull run, or as some economists were calling it, a resource super-cycle. The financial sector driven by high consumer debt experienced a bear run not seen since

the late 1990's. Other industries experienced mixed cycles as a result of consumer spending, rise in resource cost, and decline in the accessibility of credit.

### A.1.2 All Share Index 40

The ALSI40 is not a share price, but rather an index of shares. The ALSI is short for the all share index, and refers to the JSE all share index. The ALSI40 is a sub-set of the ALSI, made up of the forty largest companies by market capitalisation across all sectors listed on the JSE. The ALSI40 share price history is outlined in Figure A.1. Between 2003 and 2008, South Africa was a commodity based country, with a large portion of the JSE consisting of mining stock. It is for this reason that the ALSI40 reflects the commodity boom during the period 2003 to 2008. The financial sector is a large component of the JSE, and this too is reflected in the ALSI40, specifically the drop in 2007, which reflects the start of the financial crises. ALSI40 closed on 2003-04-03 with a share price of R73.55 and closed on 2008-06-16 with a share price of R291.82.

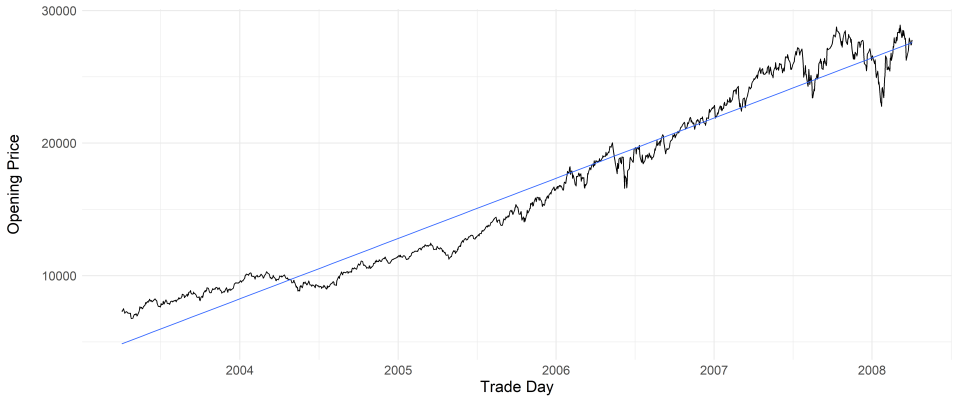



**Figure A.1:** ALSI40 share price from 2003–2008. The ALSI40 represented the top 40 shares on the JSE by market capitalisation.

### A.1.3 Anglo American Plc

Founded in 1917, AGL [9] is a multinational mining company based in Johannesburg, South Africa and London, United Kingdom. AGL produces 40% of the worlds platinum and is a major producer of diamonds, copper, nickel, iron ore, and metallurgical and thermal coal. AGL is listed on the LSE as AAL and the JSE as AGL. AGL operates in the metals and mining sector. The AGL share price history is outlined in Figure A.2. The graph shows how AGL benefited from the commodities super cycle.

AGL closed on 2003-04-03 with a share price of R121.00 and closed on 2008-06-16 with a share price of R532.50.

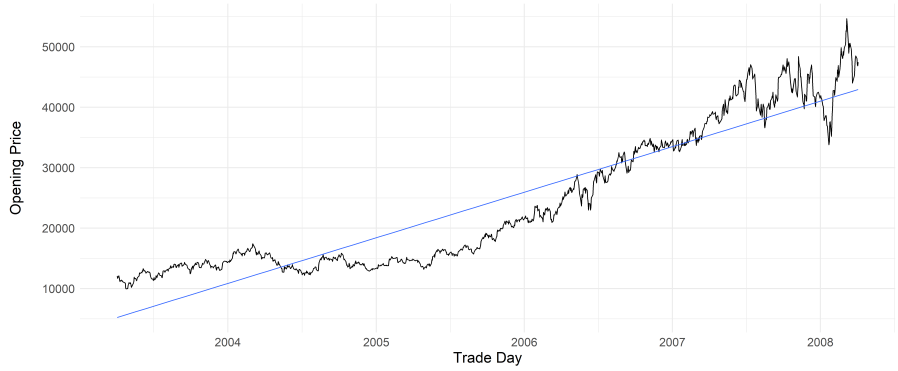



**Figure A.2:** Anglo American Plc share price from 2003–2008.

### A.1.4 BHP Billiton Plc

Created in 2001 [10], by the merger of an Australian company, Broken Hill Proprietary Company (BHP), founded in 1885 and a British company, Billiton, founded in 1860. BIL is a multinational mining company based Melbourne Australia. BIL produces iron ore, manganese, petroleum, aluminium, copper, lean, zinc, uranium, coal, nickel, cobalt, stainless steel, titanium, and diamonds. BIL is listed on the JSE, NYSE and LSE as BIL, BBL and BLT respectively. BIL operates in the metals and mining sector. The BIL share price history is outlined in Figure A.3. The graph shows how BIL benefited from the commodities super cycle. BIL closed on 2003-04-03 with a share price of R41.90 and closed on 2008-06-16 with a share price of R300.00.

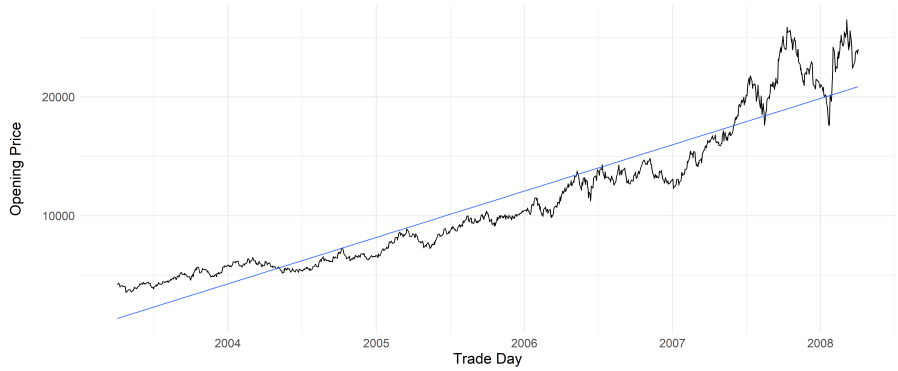



**Figure A.3:** BHP Billiton share price from 2003–2008.

### A.1.5 Gold Fields Ltd

Created in 1998 [16], with the amalgamation of Gold Fields of South Africa Ltd and Glencore Ltd, GFI is a multinational mining company based in Johannesburg, South Africa. GFI is a gold mining company. GFI is listed on the JSE, NYSE, Euronext NV, European stock exchange (ENX), SIX Swiss stock exchange (SIX), and the NASDAQ Dubai exchange as GFI, GFI, GFLB, GOLI and GFI respectively. GFI operates in the gold mining sector. The GFI share price history is outlined in Figure A.4. Gold is a currency metal, and is a haven in times of economic conflict. This is clearly seen in the share price. The data shows a rise in the share price. This was due to the Iraq war and the fall of the twin towers. As global conflict eased from 2003–2004, a drop is seen in the share price. The start of the European conflict, and the bombings in Madrid and London in 2004 saw a rise in the GFI share price. The share price gained momentum in 2006 and slowly declined during the sub-prime crises and continues to decline during the commodity super cycle of 2008 as money was moved to resources. GFI closed on 2003-04-03 with a share price of R81.01, it reached a high on 2006-07-12 with a price of R173.80, and closed on 2008-06-16 with a share price of R87.01.


**Figure A.4:** Gold Fields of South Africa Ltd share price from 2003–2008.

### A.1.6 Impala Platinum

Founded in 1917, IMP [17] is a multinational mining company based in Johannesburg, South Africa. IMP is in the business of mining, refining and marketing of platinum group metals (PGMs), as well as nickel, copper and cobalt. IMP is listed on the JSE, LSE, and OTC Markets Group as IMP, IPLA and IMPUY respectively. IMP operates in the platinum sector. The IMP share price history is outlined in Figure A.5. The graph shows how IMP benefited from the commodities super cycle. IMP closed on 2003-04-03 with a share price of R53.00 and closed on 2008-06-16 with a share price of R307.00.

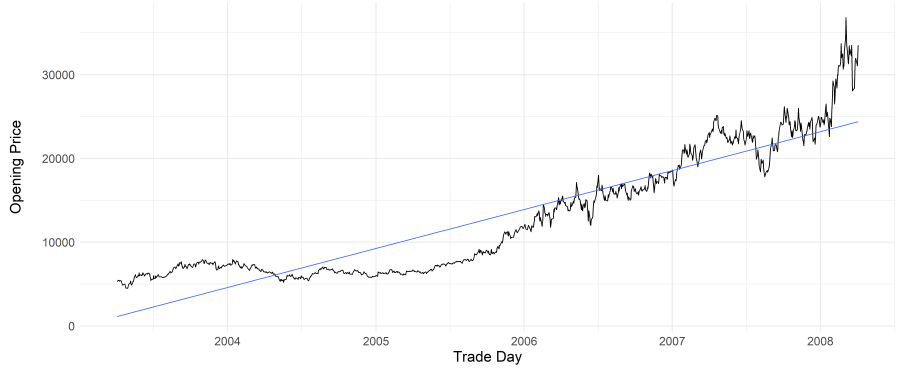



**Figure A.5:** Impala Platinum share price from 2003–2008.

### A.1.7 Lonmin

Founded in 1909, LON [20] is a British mining company, operating in South Africa. LON is based in London, England. LON is a producer of platinum group metals (PGMs). LON is listed on the JSE, and LSE as LON and LMI, respectively. LON operates in the platinum sector. The LON share history is outlined in Figure A.6. The graph shows how LON benefited from the commodities super cycle. Mid 2007 saw a decline in LON shares. This was due to a number of factors including, the global financial crisis and the decrease in accessible capital, a hostile take over bid from Xstrata that failed in 2008, the eventual sale of shares by Glencore in 2012, and labour unrest that resulted in the Marikana massacre in 2012 [21, 22]. LON closed on 2003-04-03 with a share price of R87.00 and closed on 2008-06-16 with a share price of R503.00.

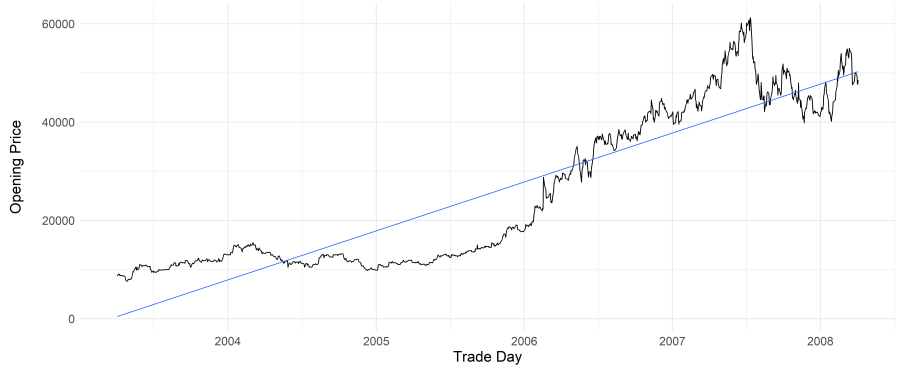



**Figure A.6:** Lonmin share price from 2003–2008.

### A.1.8 Nedcor Ltd

Founded in 1888, NED, commonly known as Nedbank [23], is one of the largest banks in South Africa. NED was founded in Amsterdam as the *Nederlandsche Bank en Credietvereeniging voor Zuid-Afrika* [24]. NED is based in Johannesburg, South Africa, and has interests in finance, mortgage, investments and banking. NED is listed on the JSE as NED. NED operates in the banking sector. The NED share price history is outlined in Figure A.7. The graph shows how the flow of money out of financial markets affected NED at the beginning of 2000 to 2004. South Africa was awarded the football World Cup in 2004 [3]. The promise of infrastructure upgrades, and greater investment saw the financial industry increase in value. This is reflected in the NED share price. The downturn in interest rates saw an increase in the financial sector. In 2006, South Africa was experiencing their lowest interest [1] rates in years, meaning more people began to borrow money, credit increased, and the financial industry benefited. The financial crisis started in 2008, and this saw the fall of the banking sector. This fall is reflected in the NED share price. NED closed on 2003-04-03 with a share price of R92.05 and closed on 2008-06-16 with a share price of R90.25.


**Figure A.7:** Nedcor Ltd share price from 2003–2008.

### A.1.9 Richmond Securities AG

*Compagnie Financière Richemont SA*, also known as RCH [28], is a Switzerland based luxury goods holding company founded in 1988. RCH is listed on the JSE and SIX as RCH and CFR, respectively. RCH operates in the management, logistics and luxury goods sectors. The RCH share price history is outlined in Figure A.8. RCH markets to the wealthy; its goods include expensive commodities such as gold, platinum, diamonds, and leather. RCH offers alternative investments to the super rich in the form of luxury goods such as watches or jewellery. The commodities boom saw an increase in the cost of

gold, diamonds, and platinum as reflected in the share price of RCH. As credit became available from 2006 onwards, more people could afford luxury items, which saw an increase in the RCH share price. RCH closed on 2003-04-03 with a share price of R10.30 and closed on 2008-06-16 with a share price of R46.75.

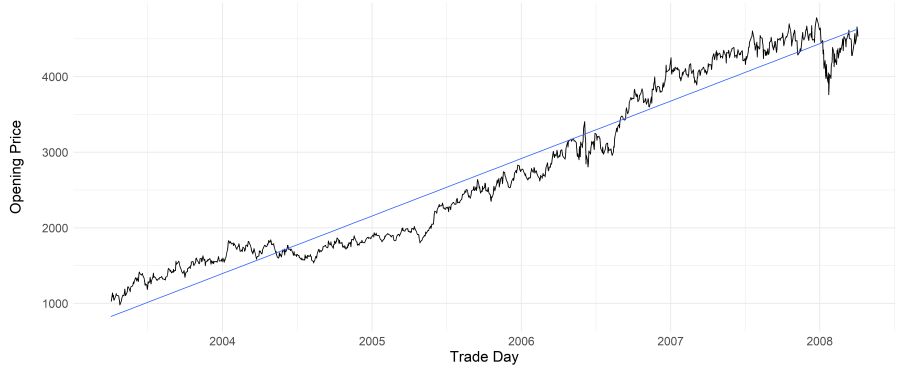



**Figure A.8:** Richmond Securities AG share price from 2003–2008.

### A.1.10 Remgro Ltd

Founded in 1968, REM [27] is a multinational investment holding company based in Stellenbosch, South Africa. REM consists of diversified stock; with shares in food, banking, healthcare, industrial, insurance, infrastructure, media, and sport. REM is listed on the JSE as REM. REM operates across many sectors and is referred to as a conglomerate. The REM share price history is outlined in Figure A.9. The inflation increase of 2000 saw an increase in goods and services. Inflation was due to a number of reasons. Two of the biggest reasons for inflation at that time was the opening of the South African market to the world post 1994 [158], and the inclusion of a market sector that, due to the laws of Apartheid, was not allowed to participate in. Both the ability to sell to the rest of the world, and to sell to a new local market increased the demand in goods, thereby increasing prices. The demand for goods increased further when interest rates were decreased and the availability of credit increased. REM co-founded South Africa's first cellular telecoms company, Vodacom, in 1993 [14] and unbundled British American Tobacco in 2008 [14]. All these factors contributed to an increase in revenue for REM, which is reflected in the share price. REM closed on 2003-04-03 with a share price of R52.51 and closed on 2008-06-16 with a share price of R205.00.

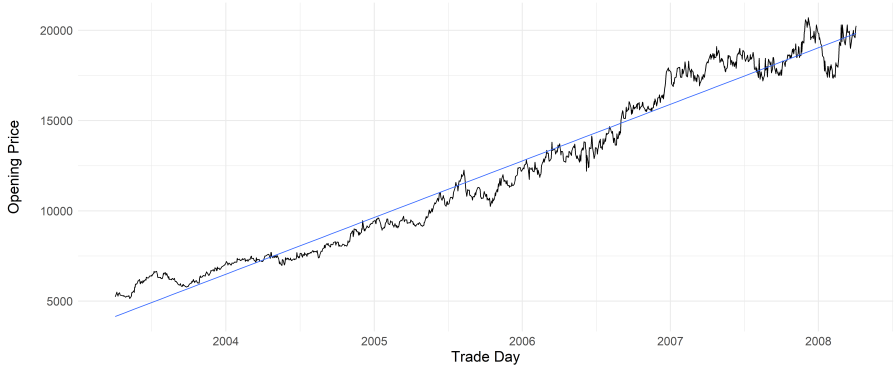



**Figure A.9:** Remgro Ltd share price from 2003–2008.

### A.1.11 SABMiller Plc

Founded in 1895, SAB [37] was a multinational brewing and beverage company head-quartered in London, England. SAB was the first industrial company to list on the JSE in 1897. From the early 1990's SAB expanded internationally through acquisitions until eventually it acquired Miller Brewing in 2002 to form SABMiller Plc and moved its primary listing from the JSE to the LSE. SAB traded as SAB on the JSE. SAB is primarily a beer brewing and distribution company. However, it also bottles and produces non-alcoholic and alcoholic beverages. SAB has operations throughout the world. The SAB share price history is outlined in Figure A.10. The graph shows the rapid growth of SAB as it acquired company after company. The financial crisis and sudden drop in access capital saw the share price fall in 2008, but it continued to climb as the markets stabilised. SAB closed on 2003-04-03 with a share price of R52.00 and closed on 2008-06-16 with a share price of R184.25.

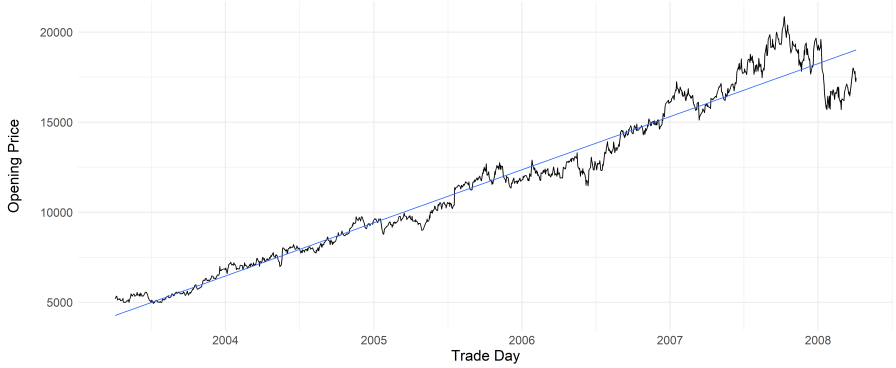



**Figure A.10:** SABMiller Plc share price from 2003–2008.

### A.1.12 Standard Bank Group Ltd

Founded in 1862, SBK [39] is a multinational financial services company based in Johannesburg, South Africa. SBK is in the business of commercial banking, foreign exchange, insurance, investment banking, investment management, private banking and wealth management. SBK is listed on the JSE as SBK and operates in the financial and banking sector. The SBK share price history is outlined in Figure A.11. The decrease in the interest rate saw an increase in credit demand, reflected in the SBK share price. The interest rate began to climb in 2006, which resulted in greater profit for the bank until the financial crisis of 2008, which resulted in an increase in SBK share price followed by a rapid fall. SBK closed on 2003-04-03 with a share price of R28.00 and closed on 2008-06-16 with a share price of R78.01.

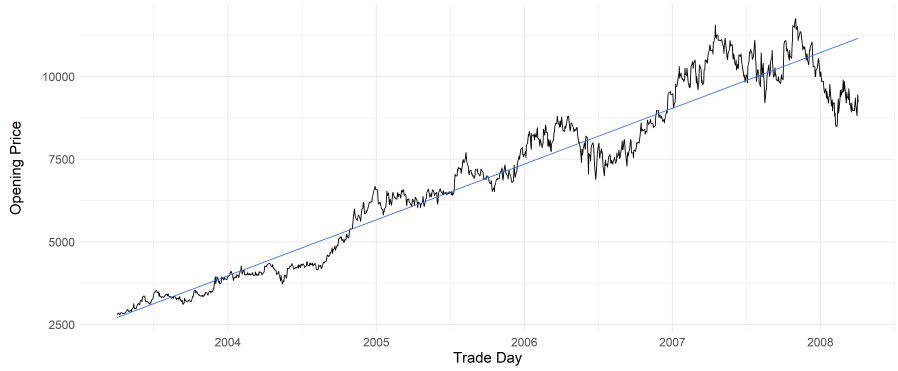



**Figure A.11:** Standard Bank share price from 2003–2008.

### A.1.13 Sasol Ltd

Founded in 1950, SOL [30, 38] is an energy and chemical company based in Johannesburg, South Africa. SOL develops synthetic fuels and produces different liquid fuels, and electricity. SOL is listed on the JSE, and NYSE as SOL and SSL, respectively. SOL operates in the oil and gas sector. The SOL share price history is outlined in Figure A.12. SOL benefited from the commodities super cycle, which is reflected in its share price. SBK closed on 2003-04-03 with a share price of R90.10 and closed on 2008-06-16 with a share price of R464.01.

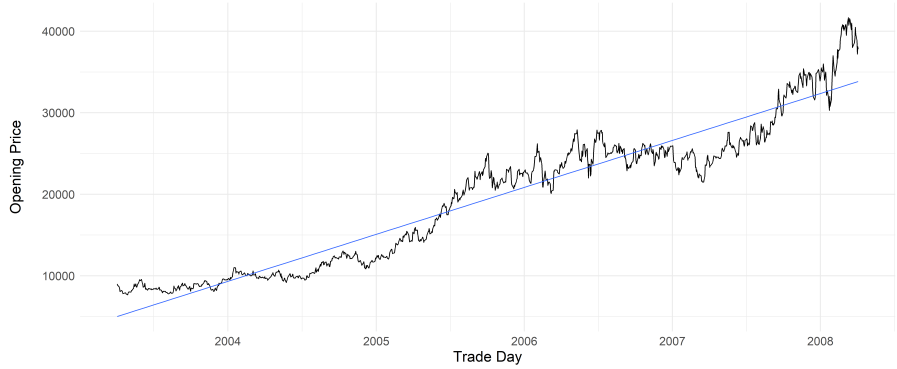



**Figure A.12:** Sasol Ltd share price from 2003–2008.

### A.1.14 Inverted shares

South Africa has in the past been a commodity driven economy. This is reflected in the history of the ALSI40 index, and the number of commodity companies that contributed to the index. It is for this reason that South Africa benefited from the commodities super cycle that occurred between 2003 and 2008. Almost all sectors of the South African economy benefited. While the financial sector saw a sharp decline towards the end of 2008, it was not enough to end lower than the start of 2003. It is for this reason that three simulated share prices were introduced.

Consider the possibility that the share data reflects all possible share information, and that the trend can be observed using technical analysis. More importantly, consider the possibility that technical analysis can predict the trend change and capitalise on it. It is then logical that the market is not random because it can be forecast. From this assumption, it is logical to say that the factors that predict the forward movement of the share should work in reverse. Based on this, three shares were selected, namely NED, REM, and SBK and their share data was reversed so that the price on the last day 2008-06-16 became the price on the first day 2003-04-03.

The inverted shares are referred to as INVNED, INVREM, and INVSBK, and depicted in Figures A.13, A.14, and A.15, respectively. This creates a new trading dataset which is different from the datasets already discussed. INVNED begins bullish increasing until mid 2004 when it drops suddenly, increases again in 2005 until July and then falls rapidly. INVREM is a typical bear chart that begins high and continues to fall. INVSBK provides a bull run from the start of trade until 2004 and then it falls with a small upswing in 2005.


**Figure A.13:** Inverted Nedbank Ltd share price from 2003–2008.

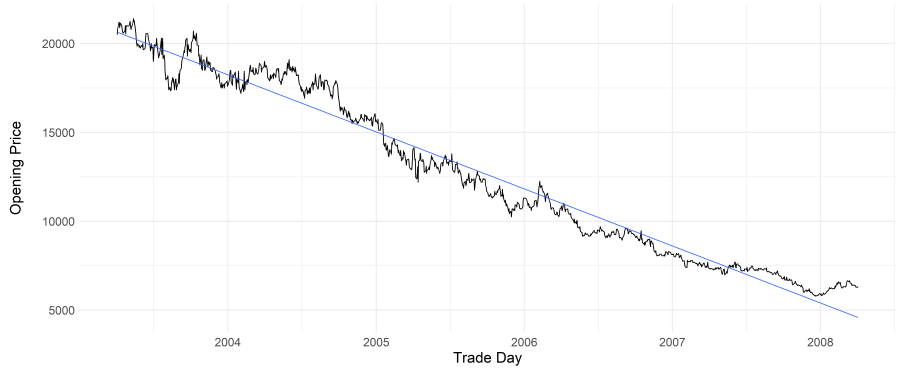



**Figure A.14:** Inverted Remgro share price from 2003–2008.

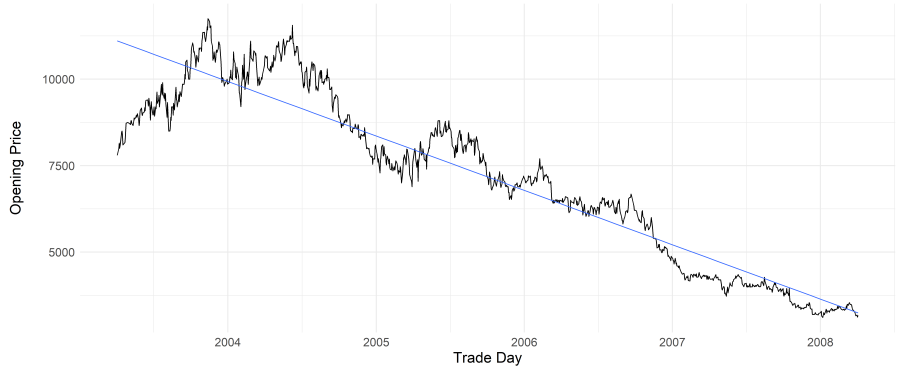



**Figure A.15:** Inverted Standard Bank Ltd share price from 2003–2008.

# Appendix B

# Acronyms

**AAV** Accumulated (Average) Asset Value. 62, 63, 84, 85, 119

**ACM** Association for Computing Machinery. 41

**ADI** Accumulation/Distribution Index. 26

**AGL** Anglo American Plc. 159, 160, 198

**AGPC** Annual Genetic Programming Conference. 41

**ALSI40** All Share Index 40. 76, 108, 160, 197

**ANN** Artificial Neural Network. 39, 41, 67

**ASA** American Statistical Association. 17

**BB** Bollinger Bands. 23, 24, 66

**BEAGLE** Biological Evolutionary Algorithm Generating Logical Expressions. 44, 45

**BIL** BHP Billiton Plc. 160, 161, 199

**CCI** Commodity Channel Index. 25

**CGA** Canonical Genetic Algorithm. 43

**CO** Chaikin Oscillator. 22

**CPIX** Consumer Price Inflation Index. 195

**EA** Evolutionary Algorithm. 31, 37, 38, 40, 46, 54, 56, 61, 69, 70, 84, 88, 100, 108, 109, 111, 119, 123, 135, 142, 144, 154, 156, 157, 159, 165–167, 171, 172, 175

**EC** Evolutionary Computation. 41

**EDDIE** Evolutionary Dynamic Data Investment Evaluator. 67, 68

**EMA** Exponential Moving Average. 21, 22

**EMH** Efficient Market Hypothesis. 17, 18

**ENX** Euronext NV. 200

**EP** Evolutionary Programming. 39, 40

**ES** Evolutionary Strategies. 40

**ESS** Regression Sum of Squared Errors. 69

**GA** Genetic Algorithm. 38, 40–44, 48, 51, 52, 56–59, 61–63, 65, 68, 84, 85, 111, 119, 120

**GECCO** Genetic and Evolutionary Computation Conference. 41

**GFI** Gold Fields Ltd. 200

**GP** Genetic Program. 8, 43–46, 48, 49, 51–53, 59, 65, 67–70, 72–74, 77, 81, 84, 85, 88, 92, 99, 100, 108, 109, 111, 112, 116, 119, 120, 123, 127, 134, 135, 139, 142–144, 154, 156–161, 164, 167, 168, 171, 172, 175–177

**ICGA** International Conference on Genetic Algorithms. 41

**IEEE** Institute of Electrical and Electronics Engineers. 41

**IMP** Impala Platinum. 163, 201

**INVNED** Inverted Nedcor Ltd. 142, 160, 167, 168, 209

**INVREM** Inverted Remgro Ltd. 165, 167, 168, 209

**INVSBK** Inverted Standard Bank Group Ltd. 167, 168, 209

**JSE** Johannesburg Stock Exchange. 14, 15, 27–29, 76, 108, 172, 197–208

**LON** Lonmin. 160, 202

**LSE** London Stock Exchange. 14, 15, 198, 199, 201, 202, 206

**MACD** Moving Average Convergence or Divergence. 22, 69

**MAPE** Mean Absolute Percentage Error. 68

**MD** Mean Deviation of the Moving Average. 23

**MFI** Money Flow Index. 25, 26

**MFM** Money Flow Multiplier. 26

**NED** Nedcor Ltd. 203, 209

**NYSE** New York Stock Exchange. 14, 67, 199, 200, 208

**OBV** On Balance Volume. 26, 63

**RCH** Richmond Securities AG. 204

**REM** Remgro Ltd. 163, 205, 209

**ROC** Rate of Change. 27, 69

**ROI** Return on Investment. 62, 63, 74, 80–86, 90, 95–99, 110, 114, 116–120, 125, 130–136, 140, 144, 156–170, 172, 175–177

**RSI** Relative Strength Index. 24, 25, 61, 69

**SAB** SABMiller Plc. 160, 206

**SBK** Standard Bank Group Ltd. 27, 28, 30, 79, 89, 99, 114, 124, 134, 163, 207–209

**SIX** SIX Swiss Stock Exchange. 200, 204

**SMA** Simple Moving Average. 20, 21, 23, 27, 63, 65, 66, 69, 74

**SOL** Sasol Ltd. 163, 167, 168, 208

**TEP** Typical Earning Price. 25, 26

**WMA** Weighted Moving Average. 21

# Appendix C

# Derived Publications

The following includes a list of publications derived from this thesis.

- J.F. Nicholls, K.M. Malan, and A.P Engelbrecht. Comparison of trade decision strategies in an equity market GA trader. *Computational Intelligence for Financial Engineering and Economics (CIFEr), 2011 IEEE Symposium on* IEEE, pp 1-8. April 2011.
- J.F. Nicholls, K.M. Malan, and A.P Engelbrecht. Evaluation of Fitness Functions for Evolved Stock Market Forecasting. *Artificial Intelligence Economics Research Centre (AI-ECON), 2008 WSPC Proceedings*, October 2008.