FGP: A Genetic Programming Based Financial Forecasting Tool

Jin LI

A thesis submitted for the degree of Doctor of Philosophy

Department of Computer Science

University of Essex

7 October, 2000

To *my wife*, Lei CHEN

Abstract

FGP: A Genetic Programming Based Financial Forecasting Tool

October, 2000

Jin LI, University of Essex

Computers-aided financial forecasting has been made possible following continuous increase in machine power at reduced price, increasingly easy access to financial information, and advances in *artificial intelligence* (AI) techniques. In this thesis, we present a genetic programming based machine-learning tool called FGP (Financial Genetic Programming). We apply FGP to financial forecasting.

Two versions of FGP, namely, FGP-1 and FGP-2, have been designed and implemented to address two research goals that we set. FGP-1 is intended to improve prediction accuracy over the predictions given. FGP-2 is aimed at improving prediction precision.

Predictions are available to users from different sources. We investigate whether FGP-1 has the capability of improving on them by combining them together. Based on the experiments presented in this thesis, we conclude that FGP-1 is capable of improving the given predictions in terms of prediction accuracy. This partly attributes the capability of FGP-1 of finding positive interactions between the predictions given.

Improving prediction precision is highly demanded in financial forecasting. Our investigation is based on a set of specific prediction problems: to predict whether a required rate of return can be achieved within a user-specified period. In order to improve prediction precision, without affecting the overall prediction accuracy much, we invent a novel constrained fitness function, which is used by FGP-2. The effectiveness of FGP-2 is demonstrated and analysed in a variety of prediction tasks and data sets. The constrained fitness function provides the user with a handle to improve prediction precision at the price of missing opportunities.

This thesis reports the utility of FGP applications to financial forecasting to a certain extent. As a tool, FGP aims to help users make the best use of information available to them. It may assist the user to make more reliable decisions that would otherwise not be achieved without it.

Acknowledgements

My most heartfelt thanks must go to my supervisor, Professor Edward Tsang for his insight, patience, and advice that helped me stay focused during this long process. He has been a constant source of critical and reflective ideas throughout my Ph.D. He not only supplied me with intellectual support but also helped me to improve my English. He read all my draft papers and his meticulous comments are deeply appreciated. Without his key eye for detail, enthusiasm and intelligence, this thesis would not have been possible. As a good friend and mentor, Edward taught me how to carry out research and yet remain human. For this I am forever in his debt.

Thanks also should be given to the many people in the department who have provided great help throughout my stay at Essex. Both the members of my supervisory committee, Professor Jim Doran, and Sam Steel, helped me a great deal in clarifying my research goals and gave me valuable advice and guidance. Chris Jennings, Marisa and Graham provided me the harmonious environment in the Department of Computer Science.

This postgraduate study was made possible by the funding provided by *CVCP* (The Committee of Vice-Chancellors and Principals of the Universities of the United Kingdom, under the Overseas Research Students (*ORS*) Awards Scheme) and the studentship provided by the University of Essex.

I wish to thank all my family members for all their love and ongoing support throughout my overseas study in England. I would like to thank my parents for allowing me to go to England to pursue the Ph.D. In particular, I would like to thank my elder sister and my brother-in-law for taking care of my parents during these years when I were away from home.

Last, and most important, I owe all of this work to my wife, Lei Chen. She has provided me with unconditional love, constant understanding, and unfailing encouragement, which followed the highs and lows that go with research. Her constant reassurance helped me maintain a positive attitude and be more confident. This thesis is dedicated to her.

Related Publications

[Li, 1999] Li, J. (1999). FGP: A genetic programming tool for financial prediction. *Proceedings of GECCO-99 PhD Student Workshop*. Orlando, Florida, USA, July 13-19 1999. p374. (A brief summary of the research in this thesis).

[Li & Tsang, 1998] Li, J. & Tsang, E.P.K. (1998). Market efficiency, predictability and genetic algorithms, March 1998. Technical Report CSM-307, University of Essex. (Chapter 2).

[Li & Tsang, 1999a] Li, J. & Tsang, E.P.K. (1999a). Improving technical analysis predictions: an application of genetic programming. *Proceedings of The 12th International Florida AI Research Society Conference.* Orlando, Florida, May 1-5, 1999, 108-112. (Chapter 4).

[Li & Tsang, 1999b] Li, J. & Tsang, E.P.K. (1999b). Investment decision making using FGP: a case study. *Proceedings of The Congress on Evolutionary Computation (CEC'99).* Washington DC, USA, July 6-9 1999, 1253-1259. (Chapter 4).

[Li & Tsang, 2000] Li, J. & Tsang, E.P.K. (2000). Reducing failures in investment recommendations using genetic programming. *Proceedings of 6th International Conference on Computing in Economics and Finance, Society for Computational Economics*. Barcelona, July, 2000 (a revised version was submitted to the Journal of Computational Economics, under review). (Chapter 5).

[Tsang et al., 1998] Tsang, E.P.K., Li, J. & Butler, J.M. (1998). EDDIE beats the bookies. *International Journal of Software, Practice & Experience*. Wiley, Vol.28 (10), 1033-1043. (Chapter 4).

[Tsang & Li, 1999] Tsang, E.P.K., Li, J. (1999). A genetic programming tool for financial forecasting (submitted to *Journal of forecasting*, under review). (Chapter 4).

[Tsang & Li, 2000] Tsang, E.P.K. & Li, J. (2000). Combining ordinal financial predictions with genetic programming. *Proceedings of the Second International Conference On Intelligent Data Engineering And Automated Learning*. (IDEAL2000) December 2000. Hong Kong. (Chapter 4).

[Tsang et al., 2000] Tsang, E.P.K., Li, J., Markose, S., Er, H., Salhi, A., and Iori, G. (2000). EDDIE in financial decision making (submitted to the *Journal of Finance and Management*). (Chapter 4 and Chapter 5).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past few decades, computers have played an increasingly important role in the financial markets. The earlier applications of computers mainly involved automating some routine tasks such as storing, retrieving, and transmitting information. Computers have also been used to automate the trading process itself by executing the trades and crossing orders electronically. Nowadays, perhaps the most interesting and promising aspect of using computers is to find a way of making financial decisions.

The role of computers in assisting users to make financial decisions has been made possible following continuous increase in machine power at reduced price, increasingly easy access to financial information and advances in the *artificial intelligence* (AI) techniques.

Information regarding financial markets has never been so easily collected. Several years of historical data for stocks, commodities and foreign exchange rates are widely available in an electronic format. They can either be purchased readily from several information vendors at low cost or acquired via the Internet. An increasing amount of crucial and commercially valuable information is becoming available on the World Wide Web. For instance, Reuters (http://www.investools.com), Financial Times (http://www.ft.com), Bloomberg (http://www.bloomberg.com), CNN (http://www.cnnfn.com) and so on provide real-time news and quotations of stocks, bonds and currencies, etc. for free.

At the same time, recent developments of AI techniques, in particular, machine learning algorithms have been creeping into the financial sector (Deboeck 1994; Economists 1993; Goonatilake & Treleaven 1995; Lederman & Klein 1995). Examples of these techniques are decision-tree inductions (Quinlan 1986a; Quinlan 1993), neural networks (Rumelhart & McClelland 1986) and genetic algorithms (Holland 1975), and so on.

In this thesis, the goal of this research is to employ AI techniques, more precisely, genetic programming (Koza 1992; Koza 1994), to build a machine learning tool for financial forecasting. The tool is to augment, rather than replace, human intelligence. The aim of the tool is to help users to make the best use of amount of information available to them. As a result, the tool may assist users to make more reliable decisions that would otherwise not be achieved without it.

1.1 What Is Genetic Programming?

Both genetic algorithms (GAs) (Holland 1975; Goldberg 1989) and genetic programming (GP) (Koza 1992; 1994) are machine learning approaches inspired by Darwin's evolution theory (Darwin, 1859). GP differs from GA mainly in the representations used. In GP, the *individual* in a *population* is represented by a tree-like computer program, which dynamically varies both in size and shape. In contrast, the *individual* in GAs is a chromosome, which is traditionally represented by bit-string with fixed length (Holland 1975). Nevertheless, both paradigms share similar fundamentals. Thus in what follows in this thesis, the term GAs will be used to refer to both GAs and GP in general, whilst the term GA is used to specifically refer to the genetic algorithm approach, without genetic programming included, unless otherwise stated explicitly.

GAs have been originally proposed as a general model of adaptive processes, but by far, this techniques have been largely applied to the domain of optimisation (Bäck et al. 1997). This is true for the financial forecasting application here. In fact, GAs are employed as learning and optimisation strategies in a large part of GA applications to finance.

As optimisation strategies, GAs operate by iteratively evolving a population of individuals. On each iteration, all individuals are evaluated in terms of the fitness function. A new population is then generated by probabilistically selecting the most fit individuals from the current population. Some members in the new population are carried forward from the last generation population intact via reproduction operation. The rest are generated by applying genetic operators: *crossover* or *mutation*. Such a process continues until sufficiently fit individuals are found.

The mathematical foundation for GA is Holland's Schema Theorem (Holland 1975). This theory is predicated on survival of the fittest. Individuals that exceed the mean fitness level of population are more likely to pass on their genes. The Schema Theorem was literally translated as Building Block Hypothesis for GP, where the equivalent to schemas is sub-programs or sub-trees (Koza 1992).

GP has been applied by Koza (1992; 1994) to a variety of fields, including optimal control, planning, discovery of game-playing strategies, symbolic regression, automatic programming, or evolving emergent behaviour. Since then Koza's work has sparked a rapid growth of genetic programming, and resulted in a large quantities of research work about this subject. Applications of GP cover a variety of disciplines (for example, see, Angeline et al. 1999; Banzhaf et al. 1999; Koza 1996; 1997; 1998).

Though GAs application to finance is still in its infancy, the amount of research work devoted into this area is increasing. In Chapter 2, we will provide a survey on this topic (cf., Section 2.5).

1.2 Why Use GAs?

The purpose of forecasting is aimed at finding patterns. Forecasting in finance is extremely difficult and complicated due to many inherent interactive factors. The search space is enormous and highly complex (Fogler 1993; Dorsey & Mayer 1995). Indeed, a powerful and suitable search technique is needed.

GAs (Holland 1975) are search techniques that are based loosely on simulated evolution.

Evolution is known to be a successful, robust method for adaptation within biological systems. The simulated evolutions, GAs have also been applied successfully to a variety of machine learning tasks and optimisation problems (Goldberg 1989; Davis 1991; Bäck, 1996). Nevertheless, the potential of GAs in application to finance has not been fully exploited.

Besides, the choice of GAs is certainly due to their inherent strengths, as well as to some properties of financial forecasting problems. As learning and optimisation approaches, GAs have strengths which are listed below.

- They can search spaces of patterns containing complex interacting parts, where the impact of each part on overall pattern fitness may be difficult to model (Mitchell 1997).
- They conduct a search from many points simultaneously and are therefore more likely to find better solutions.
- Given enough data, GAs do not need any prior knowledge in order to find patterns. This makes them suited to solve problems without clear solutions.
- GAs can generate the patterns that are comprehensive to users.
- They are easily parallelized so that there is a potential to speed up any GA-based running systems if necessary.

Unlike other optimising problems, problems in financial forecasting have the following properties.

- Many financial forecasting problems are mathematically intractable and no clear solutions exist.
- Many decision factors (or variables) are both discrete and correlated; thus, the better solutions are combinatorial.

- Any better solutions to forecasting problems are desirably comprehensible so that one is more likely to obtain some insights from them.
- A slight improvement is worth a lot.

Considering both the properties of financial forecasting problems *per se* and strengths that GAs have, we argue that GAs may be one of the most suitable approaches to attacking financial forecasting problems.

1.3 Research Problem

Employing AI techniques, more specifically, GAs could attack various financial problems. Potential topics range from financial forecasting, trade strategy optimisation, to portfolio management, artificial stock markets and other financial theory modelling (see more details of these applications in Chapter 2).

We select a financial topic as our research target based on the principles as follows.

1) People in both financial and non-financial spheres should easily understand the topic.
2) The topic should also be addressed by other machine learning approaches, such as decision-tree induction, neural network and classifier system. Thus, the approach that we propose in this work can be evaluated and compared properly.
3) The topic should not be merely of value to finance. Moreover, it should hold its generality for other disciplines, particularly those in computer science.

Bearing these principles in mind, we are determined to select financial forecasting as the topic that we think is legitimate and appropriate.

1.3.1 A Prediction Problem

Having concentrated on the topic of financial forecasting, we are still faced with a diversity of choices. A further focus is necessary.

Time series forecasting, whose goal is to forecast the value of the time series *k* steps into the future, is a good option. In fact, such a study has already been carried out in finance with a high degree of success. Resultant models include linear regression models like ARIMA (cf., Box & Jenkins 1976; Harvey 1993; Taylor 1986;) and non-linear regression models like ARCH model (Engle 1982) and GARCH (Bollerslev, 1986), etc.

The time series forecasting problem that we tackle is formulated as one whose goal is to forecast the direction and magnitude of change of the time series *k* steps into the future. This is a special type of classification problem, which can be straightforward approached by many machine-learning algorithms.

As a research strategy, it seems wise to start with the simple cases before we attempt the complex cases. Here, we set up a specific prediction problem, which we denote as P_n^r , as follows.

> P_n^r: We predict whether or not the price will increase a required *r*% (e.g. 2%) or more within a user-defined period *n* (e.g., 21 days).

Based on this prediction, each period can be classified into either a *positive* position, which means the price does increase *r*% within *n* time periods, or a *negative* position, which means the price does not increase *r*% within *n* time periods. In what follows, for convenience, an actual positive position sometimes is called an *opportunity* whilst a positive position predicted by our tool sometimes is called a *signal*.

This is a typical two-classification prediction problem. In the field of machine learning, this prediction problem belongs to the category of *concept learning*; the approach that we take belongs to the category of *supervised learning*.

To address this problem, our genetic programming based tool needs to be fed with a finite set of samples of solved cases. The data for each case consists of a pattern of observations and the corresponding correct predefined class. The purpose of our tool is to find a general way of relating any pattern of observations to the corresponding correct class. Such a mapping, represented by decision rules, is expected to have predictability over new cases.

In machine learning, the set of potential observations relevant to a particular problem are also referred to as *features*, as well as other names, including *attributes*, *variables*, *tests* and *measurements* (Weiss & Kulikowski 1991). In the GP world, a "feature" would more likely be referred to as an "input" and the predefined class would more likely be referred to as "output." In this thesis, we also like to use the term *indicator* to denote a *feature*, or an *input*.

We hope that by confining our attention to this kind of prediction problem, the effectiveness and usefulness of the genetic programming based tool can be demonstrated. We also hope that the techniques developed in this tool could be of value for solving problems beyond the scope of financial forecasting domain.

1.3.2 Assumptions of data

Every machine learning method makes some assumptions regarding tasks, together with data that it manipulates. In the case of our research, these assumptions include the following:

Assumption 1: The data set represents a supervised classification problem. Each case consists of a number of indicators, and a single, labelled class predefined based on the prediction problem chosen. In this thesis, we focus on binary classification.

Assumption 2: The data set has no missing values.

Assumption 3: Input indicators in the data set are limited to discrete or continuous numeric values. In the case of discrete non-numerical data type, this involves assigning a distinct natural integer to represent each category.

The tool we develop in this thesis is only applicable when all three assumptions are met for a problem at hand.

1.4 Research Goals

In this thesis, the goal of research is to design and implement a genetic programming based machine-learning tool. We use the financial forecasting domain to focus our research. More specifically, we apply genetic programming techniques to build a machine-learning tool for approaching classification problems relevant to financial forecasting. Ideally, the tool is capable of generating decision trees effectively.

The crucial purposes of the tool we propose are:

Goal_1:	To improve prediction accuracy
Goal_2:	To achieve a low rate of failure

Both goals are not only of value to researchers in academia, but also of great interest to users in investment.

In machine learning, the objective of learning classifications form sample data is to classify and predict successfully about new data. As the most commonly used measure of success, prediction accuracy is usually a primary concern in almost all applications of learning. Much of the research in learning has tended to focus on improving prediction accuracy (Quinlan, 1996a). This is also true for financial forecasting here. Prediction accuracy may be even more important because any slight improvement in its accuracy may potentially translate significant profits. In this thesis, we are particularly interested in improving the accuracy of given predictions by combining them together. In this work, prediction accuracy is also called the *Rate of Correctness*

(RC), which is defined as follows.

$$\text{RC} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}.$$

Apart from prediction accuracy, the *Rate of Failure* (RF) is also one of the most important concerns, particularly in financial forecasting. The rate of failure is the fraction of positive positions predicted that are predicted incorrectly (A formal definition can be seen in Chapter 5). For instance, if a system has capabilities to achieve a low rate of failure, it implies that the system more likely generates correct signals (i.e. improve prediction precision). In another word, the reliability of prediction based on this system is improved. Obviously, such a system is much more attractive to the user. We argue that a low rate of failure is one of the most desirable factors in financial forecasting.

Note that *Goal_2* is only significant in cases where a further improvement on prediction accuracy is either unavailable or very difficult. The reason is simple. For example, perfect prediction accuracy would make *Goal_2* meaningless, as the rate of failure reaches the extreme, zero. However, in practice, given a machine learning approach (e.g. ID3, C4.5 (Quinlan 1986b; Quinlan, 1993)), higher prediction accuracy is difficult to achieve, even impossible in some cases, let alone the perfect prediction accuracy. In such cases, thus, the work aimed at *Goal_2* becomes relatively indispensable and even significant. In fact, recently, more and more efforts has been made in machine learning, which is referred to as cost-sensitive learning (Turney 1995; 1997).

The method used to investigate whether or not we have achieved the two goals addressed in this thesis is empirical evaluation and comparison.

1.5 Approach and Design Goals

We propose the development of a genetic programming based machine learning tool for financial forecasting. The tool is analogous to existing decision rule induction (DRI) algorithms in machine learning. Examples of DRI algorithms include decision trees (Breiman et al. 1984; Quinlan 1993) and GAs based Classifier Systems (De Jong 1993). Underlying the tool are approaches based on the state-of-the-art genetic programming techniques. The tool should be able to generate *genetic decision trees* (GDTs) from data.

In order to make the tool useful and attractive, we propose that the following design goals must be satisfied:

- The tool must use a general purpose DRI algorithm, which is capable of processing data sets without any prior knowledge of the problem.
- The tool should be able to allow users to channel their knowledge into the process of decision trees generation
- The algorithm must achieve a level of predictive accuracy on a par with other DRI techniques when given the same data.
- The generated GDTs should be comprehensible to users.
- It should provide friendly interfaces for users to change selectable parameters conveniently and run the tool easily.

In summary, the tool, in its general form, can be applied to any inductive machine learning problem: given a database of examples, the tool will return a general description applicable to examples both within and outside the database.

1.6 Thesis Overview

The work in this thesis is to develop a genetic programming based machine-learning tool that focuses on financial forecasting. We call our tool FGP, which stands for *Financial Genetic Programming*. FGP provides crucial techniques that are necessary for addressing financial forecasting on a basis of genetic programming techniques. FGP attempts to achieve two goals: a) to improve the accuracy of given predictions and b) to improve the reliability of predictions by reducing the rate of failure. The usefulness and effectiveness of the tool are demonstrated through addressing mainly a series of prediction tasks, P_n^r. In order to evaluate FGP, we compare results achieved by our approach with those by other available machine learning approaches. The applicability of FGP is investigated via extensive experiments. We have organised the content into three parts and a concluding chapter.

Chapter 2 is the first part of this thesis with two separate sections. The first section deals with some fundamental financial issues relevant to our work. This serves two purposes. The first is to investigate whether or not our work stands on firm ground; the second is to derive some potential technical indicators from financial literature that are to be used as input to our tool. The second section presents a literature review on application of GAs to finance. This helps to understand the background of our work.

We begin in Chapter 2 by discussing the concept of *Efficient Market Hypothesis* (EMH) and its three forms associated. We focus on debating on the weak form of EMH in finance by listing relevant publications. We discuss reasons why testing EMH is difficult endeavour. In terms of our survey, we conclude that patterns exist in financial markets. We argue that EMH is an economic model, an unrealistic benchmark that is unlikely to hold in practice. All contents above serve the first purpose.

For the second purpose, we study previous technical analysis rule tests in finance literature. In the light of our study, we derive some technical indicators, which are needed as input to our tool.

The second section in Charter 2 reviews the state-of-the-art GA applications to finance. We carry out our survey along two lines in terms of operation modes of GAs. We review some important work in finance that relies on the adapting mode of GAs. We also review a large quantity of existing financial applications of GAs by virtue of its optimising mode.

Chapter 3 and Chapter 4 construct the second part of this thesis. In this part, we describe the framework of FGP and report our initial applications of FGP to some financial forecasting problems. All work reported in this part is aimed at achieving the first goal of using FGP, i.e. to improve prediction accuracy over base predictions available. We call this first version of our program FGP-1, in which prediction accuracy (i.e. RC) is mainly taken as the fitness function.

In Chapter 3, we describe what basic components of genetic programming techniques are adopted in FGP, and what kinds of new component are added in FGP. Meanwhile, the underlying algorithms are given in detail, together with the description of major parameters used for running GP in FGP.

Chapter 4 demonstrates the effectiveness of FGP-1 through several examples. Given a finite of base predictions available, FGP-1 can be used to improve prediction accuracy by combining them. Examples are categorised into two groups. In the first group, base predictions are ordinal expert forecasts; in the second group, base predictions comes from individual non-adaptive technical analysis rules. We conduct comparison with C4.5 using the examples in the second group. Our experimental results show that FGP-1 is a useful tool. However, caution should be exercised particularly for the choice of GP parameters.

The third part of this thesis consists of Chapter 5 and Chapter 6. In this part, we describe the procedure of developing a constrained fitness function and demonstrate the effectiveness of such a technique for achieving the second goal of FGP, i.e. to achieve a low rate of failure in order to improve reliability of predictions. We call the second version of our program FGP-2, in which a novel constrained fitness function is used.

Chapter 5 describes our developing process toward the invention, a constrained fitness function, which is specially developed for achieving a low rate of failure. Illustration is given by an example, together with analysis in detail. Moreover, FGP-2 is compared against three NNs and a linear classifier system reported in (Saad et al. 1998) with respect to the same prediction task over 10 individual American share prices available to us. Results show that FGP-2 beats the linear classifier and compares favourably against the three NNs. We review closely related work in machine learning, particularly in cost-sensitive learning. No similar technique is found. We conclude that FGP-2 with the constrained fitness function embedded is effective for achieving a lower rate of failure at the cost of missing more opportunities.

The applicability and effectiveness of FGP-2 with the novel constrained fitness function is further investigated in Chapter 6. We test FGP-2 over a variety of data sets and prediction tasks P_n^r with different combinations between *n* and *r*. On the other hand, limitations of the constrained fitness function in FGP-2 are also pointed out.

In Chapter 7, we conclude by summarising research work we have completed in this thesis, and presenting the contribution of our work. We also suggest future work that may improve our current research.

Chapter 2

Technical Analysis Rules and GAs in Finance

2.1 Introduction

In chapter 1, we discussed the motivation behind our research and presented the research goals we want to achieve in this thesis. The overall objective is to build a genetic programming based tool for financial forecasting. In order to illustrate the usefulness and effectiveness of the tool for forecasting, we also defined a specific financial prediction problem for the tool to attack mainly in this thesis.

This chapter is organised in two separate parts each with their own theme. The first part tries to investigate the some fundamental financial issues related to subjects of this research, including whether financial markets are efficient as described in *Efficient Market Hypothesis* (EMH) (Fama 1965; 1970; 1991; Malkiel 1992), whether technical analysis rules have merit to the predictability of markets. The purpose of the investigation is twofold: to make sure that research undertaken here stands on firm ground, and to derive some potential technical indicators from technical rules studied in financial literature as input that are necessary for evaluating our tool. The second part provides a review of GAs applications to finance, which help understand the position of our work.

The first part consists of three sections. The first section tries to investigate a fundamental question as to whether financial markets are indeed what the efficient market hypothesis defines. To address this question, we start by describing the definition of EMH and its three forms. Then we briefly review debates in finance on the weak form of EMH: papers that support or oppose this form are listed; two common approaches to testing the weak form, together with some empirical evidence contrary to EMH's weak form are discussed and provided. Finally, according to our survey, we conclude that testing EMH is a very difficult endeavour and that EMH is an economic model, an unrealistic benchmark that is unlikely to hold in practice.

The second section focuses on discussion of technical analysis. In particular, we review previous technical analysis rule tests in finance literature. We briefly describe the conception of technical analysis and present the state-of-the-art research studies regarding the predictability of technical analysis rules. We concentrate on three types of technical analysis rule, namely moving average rules, trading range break-out rules, and filter rules. We examine how these technical rules are used to generate *buy* and *sell* signals in financial studies, and then we list empirical results of testing these rules by academia in finance.

The third section presents three types of technical analysis indicators. They are derived from corresponding technical analysis rules discussed in the preceding section. Given the lack of other data at our disposal, we intend to employ these derived technical indicators as input to our tool and carry out our tests.

The second part of this chapter reviews the state-of-the-art applications of GAs in finance. The field is still young, though, recently, applying GAs techniques to finance has attracted much attention both from academics and practitioners (Deboeck, 1994; Goonatilake & Treleaven, 1995; Banzhaf et al. 1998). The subjects in finance involved by GAs range from financial forecasting, trade strategy optimisation, and portfolio management, to artificial stock markets.

2.2 Efficiency Markets Hypothesis

Over the last half-century or so, both finance academics and practitioners have debated the concept of market efficiency fiercely. Some argued that the application of sophisticated computational methods to investment management or market forecasting has little value because the market is inherently efficient. The objective of this research is to develop a tool based on the artificial intelligent techniques - Genetic Algorithms, for financial forecasting. If financial markets behave as the *Efficient Market Hypothesis* suggests, where the movements of the stock prices are random walks, then the research work that is being carried out in this thesis would be futile. Whatever tool we might build would be useless, because the possibility of using this tool to improve prediction would be approximately null. According to EMH, there should be no existing patterns concerning market future movements. Of necessity, in this section we need to address this question and present a brief financial literature review on this topic.

2.2.1 Definition and Three Forms

The origins of EMH can be traced back at least as far as the theoretical contribution of Bachelier (1900) and the empirical research of Cowls (1933). In conventional economics, markets are assumed to be efficient if all available information is reflected in current market prices (Fama 1965; 1970; 1991). Recently, Malkiel (1992) offered the following more explicit definition:

> *A capital market is said to be efficient if it fully and correctly reflects all relevant information in determining security prices. Formally, the market is said to be efficient with respect to some information set* ***F****, if security prices would be unaffected by revealing that information to all participants. Moreover, efficiency with respect to an information set,* ***F*** *implies that it is impossible to make economic profits by trading on the basis of* ***F****.*

Note that Malkiel's third sentence suggests a practical way to judge the efficiency of a market, by measuring the profits that can be made by trading on different information sets.

Notionally, Foster (1986) provided the EMH definition with an expression as follows.

$$f\,(R_{i,t}\,,\,R_{j,t}\,\ldots\,|\,\Phi^{M}_{t-1}) = f\,(R_{i,t}\,,\,R_{j,t}\,\ldots\,|\,\Phi^{M}_{t-1}\,,\Phi^{a}_{t-1}\,)$$

where $f\,(\cdot\quad) =$ a probability distribution function

$R_{i.t}$ = the return on security *i* in period *t*

Φ^{M}_{t-1} = the information set used by the *M*arket at *t-1*

Φ^{a}_{t-1} = the specific information item *A*vailable placed in the public domain at *t-1*

Note that market efficiency is defined with respect to an information item (termed Φ^{a}). One cannot address the question, "Is the market efficient?" without specifying Φ^{a}_{t-1}. Depending on the information set, there are three forms of the EMH:

1. **Weak form of EMH**: The information set includes only the history of prices or returns themselves.

If a market would be described as having this property, abnormal profits cannot be acquired from analysis of historical stock prices or volume, that is to say, one is probably wasting one’s time analysing charts of past price and/or trading volume movements, i.e. technical analysis is useless.

2. **Semi-strong form of EMH**: The information set includes all information known to all market participants (publicly available information).

If a market would be described as having this property, abnormal profits cannot be acquired from analysis of public information. In such situation, one may be wasting one’s time analysing annual reports or developing trading rules based on macro-economic data that is readily available.

3. **Strong form of EMH**: The information set includes all information known to any market participant (private information).

This form asserts that abnormal profits cannot be acquired from analysis of public and private information, and even inside traders cannot make abnormal profits.

The weak form efficient market hypothesis is the focus of our review in the next section. It is this weak form of efficiency that is associated with the term 'Random Walk Hypothesis' (Cootner 1964; Campbell et al. 1997). 'Random walk' is usually employed in the financial literature to characterize a price series, where all subsequent price changes represent random departures from previous prices. The random walk hypothesis states that investment returns are serially independent, and that their probability distributions are constant. Random walk constitutes a basis under the weak form efficient market hypothesis.

2.2.2 Is Market Efficiency Testable?

Debates between opponents and proponents of EMH are so fierce that it is impossible to provide a complete survey of the vast literature here. Thus, our reviews focus on empirical literature, in particular, those relating to the weak form of EMH. Readers who are interested in debates on the semi-strong form or/and the strong form of EMH should be referred to the textbook (Levy 1996).

A large number of empirical studies have tested the weak form of the EMH; some are summarized in Table 2.1. In general, early research provides strong evidence in favour of markets being weak form efficient. In contrast, recent papers have uncovered many anomalies, which are events or patterns that may offer investors a chance to earn abnormal return (researchers were so convinced that EMH is true that they felt any contrary evidence must be an anomaly). Those anomalies directly contradict what the weak form of EMH describes and could not be interpreted by EMH. Such facts demonstrate that the weak form of EMH is questionable and may not hold in realistic markets.

AUHORS	YEAR	ASSETS STUDIED	WEAK FORM EFFICIENT?	COMMENTS
Bachelier	1900	French securities	Yes	Tried to test if the French government securities options and futures market was efficient
Roberts	1959	U.S. Stock	Yes	Stock prices resemble random patterns
Osborne	1959	U.S. Stock	Yes	Stock prices similar to random movement of physical particles in water (Brownian motion)
Granger, Morgenstern	1963	U.S. Stock	Yes	Employed spectral analysis (a powerful statistical tool that identifies patterns), but still found no patterns
Fama	1965	U.S. Stock	Yes	Examined serial correlations and other statistical tools to check for patterns, and found no significant patterns
Fama, Blume	1966	U.S. Stock	Yes	Examined technical trading rules and found no abnormal profits
Solnik	1973	Stocks in 9 countries	Yes	Used serial correlations and found no profitable investment strategies
Merton	1980	U.S. Stock	No	Changes in variance are somewhat predictable from past data
French	1980	U.S. Stock	No	Identified a week-end effect
Keim	1983	U.S. Stock	No	Identified a January effect
Gultekin	1983	International markets	No	Identified seasonal patterns
Jaffe, Westerfield	1984	International markets	No	Identified seasonal patterns
Lehmann	1990	U.S. Stock	No	Reversal effects

Table 2.1: Summary of Evidence Related to the Weak Form of EMH (source: Table 12.1 at page 426 in (Levy 1996).

To illustrate how those anomalies are observed in finance research work, in the following two subsections, we give two detailed examples reported in recent finance studies. The two examples are generated based on two primary methods of testing the validity of the weak form of EMH (Levy 1996). They both cannot be explained in terms of the weak form of EMH.

2.2.2.1 Two Illustrative Anomalies

2.2.2.1.1 The first anomaly

The first empirical illustration is about the examination of autocorrelations[1] of security returns.

[1] Autocorrelations or serial correlations examine the extent to which past changes can be correlated current changes, if they are highly correlated (positive or negative), past changes can be used to predict future changes. Please see Appendix A: Tests of stationarity or randomness based on autocorrelation coefficient.

Under the random walk hypothesis, autocorrelation coefficients at any orders should be zero (Gujarati, 1995). If a series of finance data does not obey this law, then the random walk hypothesis is untrue. In this case, an anomaly to the weak form of EMH is found.

Sample Period	Sample Size	Mean	SD	r_1	r_2	r_3	r_4	Q_5
A. Daily Returns								
CRSP Value-Weighted Index								
62:07:03-94:12:30	8179	0.041	0.824	17.6	-0.7	0.1	-0.8	263.3
62-07:03-78:10:27	4090	0.028	0.738	27.8	1.2	4.6	3.3	329.4
78-10:30-94:12:30	4089	0.054	0.901	10.8	-2.2	-2.9	-3.5	69.5
CRSP Equal-Weighted Index								
62:07:03-94:12:30	8179	0.07	0.8	35.0	9.3	8.5	9.9	1,301.9
62-07:03-78:10:27	4090	0.063	0.8	43.1	13.0	15.3	15.2	1,062.2
78-10:30-94:12:30	4089	0.078	0.8	26.2	4.9	2.0	4.9	348.9
B. Weekly Returns								
CRSP Value-Weighted Index								
62:07:10-94:12:27	1695	0.196	2.093	1.5	-2.5	3.5	-0.7	8.8
62:07:10-78:10:03	848	0.144	1.994	5.6	-3.7	5.8	1.6	9
78:10:10-94:12:27	847	0.248	2.188	-2	-1.5	1.6	-3.3	5.3
CRSP Equal-Weighted Index								
62:07:10-94:12:27	1695	0.339	2.321	20.3	6.1	9.1	4.8	94.3
62:07:10-78:10:03	848	0.324	2.46	21.8	7.5	11.9	6.1	60.4
78:10:10-94:12:27	847	0.354	2.174	18.4	4.3	5.5	2.2	33.7
C. Monthly Returns								
CRSP Value-Weighted Index								
62:07:31-94:12:30	390	0.861	4.336	4.3	-5.3	-1.3	-0.4	6.8
62:07:31-78:09:29	195	0.646	4.219	6.4	-3.8	7.3	6.2	3.9
78:10:31-94:12:30	195	1.076	4.45	1.3	-6.3	-8.3	-7.7	7.5
CRSP Equal-Weighted Index								
62:07:31-94:12:30	390	1.077	5.749	17.1	-3.4	-3.3	-1.6	12.8
62:07:31-78:09:29	195	1.049	6.148	18.4	-2.5	4.4	2.4	7.5
78:10:31-94:12:30	195	1.105	5.336	15	-1.6	-12	-7.4	8.9

Table 2.2: Autocorrelation in daily, weekly, and monthly stock index returns (Source: table 2.4 at page 67 in Campbell et al. (1997)).

This empirical method of testing random walk model is used broadly in finance community. Before the early 1970s, quite a few research studies gave test results that supported random walk (see e.g., Kenddall 1953; Fama 1965; Granger 1972). Since then, however, there have been more studies that provide evidence of departures from the random walk theory (see e.g., Gibbons & Hess 1981; Keim 1983; Patell & Wolfson 1984; Fama & French, 1988). Detailed below is an empirical study completed by Campbell, J.Y. with his colleagues: Lo, A.W. and

MacKinlay, A.C. (1997).

The study applies the method of testing on daily, weekly and monthly value- and equal-weighted CRSP[2] stock return indices. The sample ranges from July 3, 1962 to December 30, 1994. Table 2.2 reports the means, standard deviations, autocorrelation coefficients (in percent) and Box-Pierce Q-Statistics (cf., Gujarati 1995) for the overall period and sub-periods.

The panel A in Table 2.2 reports that the daily value-weighted CRSP index and the daily equal-weighted CRSP index have first-order autocorrelation coefficients $\boldsymbol{r}_1$ of 17.6% and 35.0% respectively. Both values are much higher than the standard error of 1.11% ($\sqrt{1/N}$, N = 8179) for $\boldsymbol{r}_1$. This implies that autocorrelations of 17.5% and 35.0% are clearly statistically significant at all the conventional significant levels (e.g., $\alpha = 0.05$ or 0.001). Moreover, the Box-Pierce Q-Statistics (cf., Gujarati 1995) with five autocorrelations (Q_5) have values of 263.3 and 1301.9 respectively. They are also significant at all the conventional significance levels. Based on the results above, one has to reject that stock returns follow a process of random walk here.

The weekly and monthly return autocorrelations reported in panels B and C respectively, exhibit patterns similar to those of the daily autocorrelations: positive and statistically significant at the first lag over the entire sample and for all sub-samples, with tighter and sometimes negative higher-order autocorrelations.

The returns in indices studied here do have patterns that do not obey the random walk hypothesis. Certainly, this is an anomaly to the weak form of EMH, which could not be explained by the theory of EMH.

2.2.2.1.2 The second anomaly

The second illustrative example concerns the investigation into whether abnormal return could be

[2] CRSP stands for Center for Research in Security Prices, which is an organization that supplies security data widely used by academic investment researchers.

obtained by designed trading strategies. In terms of the weak form of EMH, it is impossible to gain any abnormal returns if trading strategies are purely derived based on past information of prices. If some deliberately designed trading strategies are able to obtain abnormal returns, then anomalies occur. This provides an alternative means to test the validity of the weak form of EMH. Quite a lot of work has been done in finance using this method (Sharpe et al. 1995; Levy, 1996).

The designed strategies illustrated here are called *momentum* and *contrarian* strategies. They suggest trading behaviour purely based on past information. They simply examine the returns on stocks over a time period that just ended in order to identify candidates for purchase and sale.

Momentum and contrarian strategies are described in detail below.

Consider ranking a group of stocks based in the size of their returns over some time period that has just ended. Momentum investors seek to purchase those stocks that have recently risen significantly in price. They believe that those stocks will continue to rise due to an upward shift in their demand curves. Conversely, those stocks that have recently fallen significantly in price are sold on the belief that their demand curves have shifted downward. Investors who call themselves contrarians do just the opposite of what momentum investors are doing in the market.

In order to test both strategies, researchers have designed the procedure as follow:

1) Rank those stocks that have been listed on ether the New York Stock Exchange (NYSE) or American Stock Exchange (AMEX) based on their returns over a just-ending time period (e.g., end of month time period).

2) Form the "loser" portfolio which comprise x% (e.g., x=10, 50) of those stocks that have the lowest average return and "winner" portfolio which comprise x% those stocks that have the highest average return.

3) Determine the returns on the winner and loser portfolios over a just-starting subsequent time period (e.g., beginning of month time period).

4) Repeat the analysis all over again, starting with step 1, but moving forward one time period. Stop after exhausting the data tested.

5) Determine the abnormal returns on the winner portfolio by subtracting the returns on a benchmark portfolio having a comparable level of risk (more details can be found at Chapter 22 in Sharpe et al. (1995)); calculate the average of abnormal returns. Similarly, calculate the average of abnormal returns on the loser portfolio.

If stocks are priced efficiently, then their past price behaviour is useless in terms of its predictive value. Neither momentum nor contrarian strategies should "work" in that winner portfolios should have same performance as loser portfolios. Both portfolios should have average abnormal returns of approximately zero. Even more important, the difference in their returns should be approximately zero.

Table 2.3 summarizes test results reported in financial studies with respect to six different time periods. In general, the contrarian strategy works well for both very short (a week or month) and very long (three or five years) time periods. Surprisingly, for intermediate periods such as six months and one year, an exact opposite strategy, i.e. momentum, seems to have merit. Note that the differences of both portfolio returns are statistically significantly different from zero, which empirically contradicts EMH’s weak form. This further demonstrates that patterns do exist in stock prices.

Annualized Abnormal Returns				
	Length of Portfolio Formation and Test Periods	**Winner Portfolio**	**Loser Portfolio**	**Winner Return Less Loser Return**
A	Weekly, 1962-1986: Top 50% and bottom 50% of NYSE and AMEX stocks	-24.9%	89.8%	-114.7%
B	Monthly. 1929-1982: Top 10% and bottom 10% of all NYSE and AMEX Stocks	-11.6%	12.1%	-23.7%
C	Semiannually, 1962-1989: Top 10% and bottom 10% of all NYSE and AMEX stocks	8.7%	-3.5%	12.2%
D	Annually, 1929-1982: Top 10% and bottom 10% of all NYSE and AMEX stocks.	5%	-6.1%	11.1%
E	Three years, 1926-1982: Top 35 and bottom 35 NYSE stocks	-1.7%	6.5%	-8.2%
F	Five years, 1926-1982: Top 50 and bottom 50 YSE stocks	-2.4%	7.2%	-9.6%

Table 2.3: Returns from momentum and contrarian Strategies (Source: Table 23.1 at pages 847 in the textbook (Sharpe et al. 1995)).

A: Lehmann, B.N. (Feb. 1990). Fad, martingales, and market efficiency. Quarterly Journal of Economics, 05, no.1. p16.

B: Jegadeesh, N. (1990). Evidence of predictable behaviour of security returns. The Journal of Finance, 45, no.3. 881-898.

C: Jegadeesh, N. & Titman, S. (March 1993). Returns to buying winners and selling losers: Implications for Stock Market Efficiency. Journal of Finance, 48, No.1. pp79.

D: Jegadeesh, N. (1990), Evidence of predictable behaviour of security returns. Journal of Finance, 45, no.3. 881-898.

E: Werner, F.M., Bondt, D. & Thaler, R. (July 1985). Does the stock market overreact? Journal of Finance, 40, no. 3. pp799.

F: Werner, F.M., Bondt, D. & Thaler, R. (July 1987). Further evidence on investor overreaction and stock market seasonality. Journal of Finance, 42, no. 3, pp561.

2.2.2.2 A Difficult Task

In the two proceeding sub-sections, we have discussed two typical kinds of anomalies reported in financial studies with respect to the weak-form of EMH. Those anomalies are events that are not anticipated and that offers investors chances to earn abnormal profits. Nevertheless, whether those anomalies are exploitable for investors to make abnormal profits in realistic markets is still questionable. Therefore, testing whether or not markets are efficient is inconclusive.

There are a number of serious difficulties in interpreting testing results. First, when testing any forms of EMH, one must assume an equilibrium model in order to define abnormal security returns. Thus, empirical anomalies, rejections of market efficiency are attributed to either a truly market inefficiency or an incorrect equilibrium model assumed. This *joint hypothesis* problem means that market efficiency as such can never be rejected because the fault cannot be located.

In finance, in order to calculate abnormal returns, researchers normally use two major traditional equilibrium models, which are the *Capital Asset Pricing Model* (CAPM) (Sharpe 1964) and an alternative model of asset pricing, know as *Arbitrage Pricing Theory* (APT) (Ross 1976). Both models were built on many assumptions that may not be satisfied in actual markets. For example, two models assume that all investors have the same expectations about mean, variances and covariances of security returns; all investors have a common time horizon (a single period) for investment decision making, etc. These assumptions certainly are untrue in the real world. Such a fact makes any calculated abnormal returns (either positive or negative) based on these models questionable.

Second, EMH was built with many assumptions. EMH bypassed many empirical problems by concentrating on an extreme set of conditions when discussing Φ_t^a (Fama 1970):

a) The information item is equally and instantaneously available to all market participants.

b) Analysis of the information item is costless.

c) All participants agree on the implications of that information item for the current price and the distribution of future prices of each security (the so-called homogeneous expectations assumption).

Given these assumptions, there was the unambiguous prediction that when Φ_t^a is placed in the public domain, the capital market reaction will be instantaneous and unbiased. Once these assumptions are relaxed, even defining Φ_t^a or "the public domain" becomes a difficult task. In reality, the actual markets are characterised by non-instantaneous availability of information to all participants, positive information analysis cost, and the existence of heterogeneous expectations across market participants. These assumptions under EMH do not exist in real capital markets.

Third, even in theory, as Grossman and Stiglitz (1980) pointed out, abnormal returns will exist if there are costs of gathering and processing information. These returns are necessary to compensate investors for their information-gathering and information-processing expenses. They are no longer abnormal when these expenses are properly accounted for.

With these facts, there is now considerable consensus that testing the validity of EMH is very difficult endeavour. We argue that perfect market efficiency is an unrealistic benchmark that is unlikely to hold in practice, even in theory.

2.2.3 Remarks on EHM

One has to be impressed with the substantial evidence suggesting that stock prices display a remarkable degree of inefficiency. Like any other economic model, the efficient markets model is an abstraction of reality. Perfect efficiency is an unrealistic benchmark that is unlikely to hold in practice. On the other hand, one has to admit some aspects of market efficiency to some extent. For example, information contained in past prices or any publicly available fundamental information is quickly assimilated into market prices. Prices adjust well to reflect some important information; if some degree of mispricing exists, it does not persist for long.

In terms of the above views, a notion of relative efficiency has been proposed by Campbell and his colleges (Campbell 1997). The efficiency of one market is measured against another, e.g., the New York Stock Exchange vs. the Paris Bourse, future markets vs. spot markets, or auction vs. dealer markets. Such a conception may be more useful than the all-or-nothing view taken by much of the traditional market-efficiency literature. For instance, one may assert this market is more efficient than other markets by means of measuring autocorrelation coefficient or calculating abnormal returns acquired by some trading strategies.

With respect to our review concerning EMH, in particular, the weak form of EMH, there are several points in summary in relation to our research. First, the evidence reviewed indicates that future prices are, in fact, somewhat predictable, which does not obey the weak form of EMH. Second, since patterns do exist, the work of developing a tool to find those potential patterns is not futile. Our research work stands on firm ground. Third, the tool we developed is not supposed to replace human experts, but, rather, to assist them to understand the law in financial markets. Finally, we should emphasis that it is not our primary concern whether the tool could lead to abnormal returns. We are more concerned with what kind of techniques or means that the tool is able to provide, in order to improve a) the accuracy of the given predictions and b) reliability of predictions for reducing investment risks.

2.3 Technical Analysis

In Section 2.2, we have investigated whether EMH, in particular, the weak form of EMH, is to hold in reality. Substantial evidence indicates that future returns are predictable to some extent on a basis of historical returns. Technical analysis is one of the methods that attempt to exploit potential rules or patterns in markets for the purpose of investment management.

2.3.1 What Is Technical Analysis?

Technical analysis is an approach to seeking some rules or patterns in order to extrapolate future price movement. It makes predictions purely based on historical data, such as price series, trading volume, and other market statistics. In their textbook on technical analysis, Edward and Magee defined technical analysis as (Edwards & Magee, 1992; p4):

> *The study of the action of the market itself as opposed to the study of the goods in which the market deals. Technical analysis is the science of*

> *recording, usually in graphic form, the actual history of trading (price changes, volume of transactions, etc.) in a certain stock or in "the averages" and then deducing from that pictured history the probable future trend.*

Technical analysis is premised on the belief that financial prices are determined by investors' attitudes. Technical analysts assume that human nature is fairly static; that is, when current investors face situations similar to those faced by investors in the past, they behave in a similar fashion. Therefore, to find historical price patterns and relationships with other variables is believed to give clues as to how the market will behave in the future. Such view is well expressed in (Pring 1991, p2-3) as follows:

> *The technical approach to investment is essentially a reflection of the idea that prices move in trends which are determined by the changing attitudes of investors toward a variety of economic, monetary, political and psychological forces…-Since the technical approach is based on the theory that the price is a reflection of mass psychology ("the crowd") action, it attempts to forecast future price movements on the assumption that crowd psychology moves between panic, fear, and pessimism on the one hand and confidence, excessive optimism, and greed on the other.*

Technical analysis originated with the work of Charles Dow in the late 1800s, and is now widely used by practitioners as input mainly for trading decisions. For example, Taylor & Allen (1992) surveyed chief foreign exchange dealers in London, and found that at least 90% of respondents employed technical analysis in forming their expectations. The reliance on technical analysis was pronounced for short prediction, while more attention was put on fundamental analysis for longer prediction (Frankel & Froot 1990).

2.3.2 Predictability of Technical Analysis

Despite its popularity, traditionally, technical analysis has been the "black sheep", regarded by academics with a mixture of suspicion and contempt. Technical analysis has never enjoyed the same degree of acceptance as that others have received, such as fundamental analysis or quantitative analysis. This attitude is partly due to the fact that its proponents have never made serious attempts to test the predictions of various technical rules employed (Neely et al. 1997).

Recently, however, technical analysis has been enjoying a renaissance both with practitioners and academics. For example, on Wall Street, many major brokerage firms publish technical commentary on the market and individual securities, and many of the newsletters published by various “experts" are based on technical analysis (Brock et al. 1992). More importantly, study on technical analysis has attracted many attentions and interests of researchers in academics. Such studies include Allen and Karjalainen (1999), Blume, Easley and O'Hara (1994), Brock, Lakonishok and LeBaron (1992), Brown and Jennings (1989), Gencay (1996), Goldberg and Schulmeister (1988), Hudson et al. (1996), LeBaron (1998), Kho (1996), Levich & Thomas (1993), Lukac, Brorsen and Irwin (1988), Neely, Weller and Dittmar (1997), Neftci (1991), Pau (1991), Raj and Thurston (1996), Silber (1994), Taylor (1994), Taylor and Allen (1992), Treynor and Ferguson (1985), etc. They argued that technical analysis might have merit in relation to the predictability of financial markets.

We should point out that it is not the purpose in this thesis to provide theoretical or empirical justification for the use of technical analysis. Our purpose is to show how genetic programming techniques can be employed to improve financial forecasting, given available predictors as input.

In order to predict future price movements, we must adopt some input as predictors to feed the tool. Potential predictors may be the explicit predictions given by experts; financial fundamental factors such as price-earning ratios, price-to-book value ratios, dividends, etc; and technical analysis factors such as filter rules, moving average rules and trading range break-out rules (all three types of technical rules will be detailed in the next section). Due to lack of data concerning fundamental factors, these will not be considered as input to the tool, whereas technical analysis factors or explicit predictions given by experts are available for us to handle. Our objective is to show that GP can add value to predictors or expert predictions that are input to it.

2.3.3 Technical Analysis Rule Tests in Finance

Technical analysis rule tests have been carried out by academics in finance for decades. Such interest coincided with and probably was motivated by the development of the random walk hypothesis and the subsequent formulation of the theory of EMH.

The tested forms of technical analysis vary according to the way in which historical prices are used. There are so many relevant technical trading rules used among practitioners and academics that it is unrealistic to list them all. In what follows, we only focus on three types of technical analysis rules that have been studied in finance literature. They are *filter rules*, *moving average rules* and *trading range break-out rules*. More important, we will derive technical rule indicators, as input to our tool, from these rules (see Section 2.4).

2.3.3.1 Filter Rules

Filter rules originated in the work of Sidney S. Alexander (Alexander 1961; 1964). Filter rule are mechanical trading rules, which attempt to apply more sophisticated criteria to identify movements in stock prices. An *x% filter rule* (e.g., *x* = 5) is defined as follows:

> *If the daily closing price of a particular security moves up at least x%, buy and hold the security until its price moves down at least x% from a subsequent high, at which time simultaneously sell and go short. The short position is maintained until the daily closing price rises at least x% above a subsequent low at which time one covers and buys. Moves less than x% in either direction are ignored* (Alexander 1961).

Underlying this filter technique is the belief that stock prices have tendencies to move further following an *x%* of rise or fall.

Alexander (1961) tested filter rules ranging in size from x=5 to x=50 on Dow Jones and Standard & Poor's stock indices. In general, filters of all different sizes yielded substantial profits compared with the buy-and-hold strategy. In his later work (Alexander 1964), Alexander took account of the bias pointed out by Mandelbrot (1963) and reworked his earlier results. This time, the profitability of the filter rules was reduced.

Fama (1965) and Fama & Blume (1966) reported a more detailed empirical analysis of filter rules. Filter rules were applied to series of daily closing prices for each of the individual securities of the *Dow-Jones Industrials Average* (DJIA) ranging from January, 1956 to September, 1962 with the 1200 to 1700 cases. After taking account of trading costs and dividends, results show that the filter rule technique only surpassed the buy-and-hold policy for two securities among 30 securities.

Sweeney (1988) selected a subset of stocks that looked most promising reported in Fama & Blume (1966). He followed these stocks from 1970 through 1982 and found statistically significant excess return over a buy-and-hold policy. The return remained positive for transaction costs obtainable by floor-traders. It is worth noting that Sweeney (1988) considered only long equity position and avoided short positions that had performed poorly in the study of Fama & Blume (1966).

2.3.3.2 Moving Average Rules

The study of moving average rules goes back at least to Donchian (1957), as well as Alexander (1961). Moving average rules are a kind of popular technical analysis rules which trigger indisputable buy and sell signals following a regulation below.

> *Buy, if a short-period (1 or 2 days, say) moving average rises above a long-period (50 or 200 days, say) moving average by a band (e.g., 0, 1%); sell, if a short-period moving average falls below a long-period moving average by a band* (Brock et al. 1992).

The idea behind moving average rules is to smooth out an otherwise volatile series in order to detect major downturns and upturns of the market. When the short-period moving average penetrates the long-period moving average, a trend is considered to be initiated.

Alexander (1964) tested moving average rules over the S & P Industrials from 1928-1961 and reported that all rules except one exceeded the buy-and-hold strategy (before commissions).

Strategies type	Annualised Average Return (%)		
	Buy Signal	Sell Signal	Buy return less sell return
A: Moving average tests:			
Variable Length	10.4	-5.8	16.2
Fixed Length	13.3	-4.6	17.9
B: Trading range break-out tests:	11.3	-5.6	16.9

Table 2.4: Returns from moving average and trading range break-out strategies.
Source: Adapted from Brock et al. (1992). Based on one-day short and 200-day long periods during 1897 to 1986 with no band (0) over DJIA; annualized assuming that there are 253 trading days in one year and 25 ten-day trading periods in a year (fixed-length is 10 days).

Brock, Lakonishok, and LeBaron (1992) extensively tested numerous moving averages rules on daily closing of DJIA from 1897 to 1986 with a total of 25,000 trading days. Without losing generality, testing results of two specific moving average rules are illustrated in Table 2.4. One moving average rule is referred to as a variable-length strategy, by which each day can be classified into either a buy day or a sell day. The other is referred to as a fix-length strategy, by which each day can possibly be classified into a holding day within a fix period (e.g., 10 trading days), in addition to a buy day or a sell day. The later strategy greatly reduces the frequency of changing position from buying to selling, or selling to buying. Nevertheless, both strategies show similar results. For one of variable length moving average rules, the annualised average return on buy days was 10.4% and on sell days was –5.8%, resulting in a significant difference of 16.2%. One of fix-length moving average rules performed similarly, with 13.3% on buy days, -5.6% on sell day, and 16.9% difference.

The above empirical results severely contradict market efficiency. According to the weak form of EMH, the average return during buy days should be approximately the same as that during sell days. That is, the difference in their returns should be approximately zero. The contrary facts imply that moving average rules have merit to predictability to future price movements.

Moving average rules can vary according to different sizes chosen for the short period and the long period, as well as the size of band selected. Many other modifications are discussed in Schulmeister (1987), Sweeney (1986), and Taylor (1992). In Brock et al. (1992), we notice an interesting finding that moving average rules with a band of 1% usually outperforms the corresponding rule without a band (i.e. band is 0, which is usually used in practitioners and studied in academics) with respect to the buy-sell difference. This implies that a band (it is also called as a threshold) is an indispensable element in a moving average rule, which has its impact on performances of moving average rules.

2.3.3.3 Trading Range Break-Out Rules

The study of trading range break-out rules was completed by Brock et al. (1992), Goldberg & Schulmeister (1988), and Raj & Thurston (1996). This kind of rule intends to mimic both the resistance level and the support level. Both levels are usually observed by technical analysts in order to identify potential changing points of market trends.

The resistance level is defined as the local maximum. Rationally, investors are willing to sell at the peak. This selling pressure results in resistance to a price rise above the previous peak. However, as soon as the price rises above the peak, it has broken through the resistance area. This break-out is considered to be a buy signal. Conversely, a sell signal is sent out when the price penetrates the support level, which is defined as the local minimum. The idea is that future price should have difficulties penetrating the support level because investors are willing to buy at the minimum price. If the price falls below this level, the price is expected to drift downward further. Concisely, a trading range break-out rule is described as follows:

> *Buy, if the price rises above a local maximum by a band (e.g., 0, 1%); sell, if the price penetrates below a local minimum by a band (a local maximums*

> *and minimums are computed over the preceding numbers of trading days, e.g., 10, 50 or 200 trading days).*

Like moving average rules, varieties of trading range break-out rules were tested by Brock, Lakonishok, and LeBaron (1992). Testing results of one of them are listed in the last row of Table 2.4. This trading range break-out rule is similar to the fix-length moving average strategy. Here the high and low prices over the past 200 trading days are accounted as local maximums and local minimums. The annualised average return on buy days was 11.3% and on sell days was –5.6%, resulting in a significant difference of 16.9%. Again, among six studied rules in Brock et al. (1992), rules with a band (e.g. 1%) exceeds corresponding one without a band in terms of the buy-sell difference. Results here are consistent with findings associated with moving average rules aforementioned.

Goldberg & Schulmeister (1988) applied such rules (rules were called the point-and-figure rules in their study), together with filter rules and moving average rules, in the stock market during the 1970s and 1980s. They examined all rules on both hourly data and daily data. They found that stock price movements do possess systematic price rules and that past prices do contain information relevant for predicting future price movements. One interesting result of the study is that all of the technical rules examined are considerably more profitable with hourly data than they are with daily data.

Raj & Thurston (1996) tested both moving average rules and trading range break-out rules on the Hang Seng Futures Index, traded at the Hong Kong Futures Exchange. They found that moving average rules studied do not produce significant excess returns, but four out of six trading range break-out rules result in significant positive returns for the buy signal.

2.4 Deriving Indicators from Technical Analysis Rules

In Section 2.3, we discussed three types of technical analysis rules: namely, filter rules, moving average rules, and trading range break-out rules. They have been reported to have merit to predictability of future price movements. In the light of these findings, in this section, we would like to derive some corresponding types of indicators from these technical analysis rules. The indicators generated are treated as predicators of future price movements.

As described in Chapter 1, the prediction problem we mainly attack in the thesis is to predict whether a share price will rise by at least *r*% within a defined period, *n*, where *r*>0. Obviously, predictors related to upturns of the prices are of interest to us. Any buy signals triggered by technical analysis rules are believed to be such predictors, whereas sell signals are assumed as predictors of downturns of price trend. Therefore, indicators should be derived in terms of the regulations of technical rules for generating buy signals.

Three types of indicators corresponding to three types of technical trading rules are defined as follows, given a time series of prices $\{P_t\}$.

1) Filter indicator: $I_{Filter\_L} = \frac{P_t - P_{\min(1,\,L)}}{P_{\min(1,\,L)}}$, where $P_{\min(1,\,L)} = \mathrm{Min}\,(P_{t-1}, P_{t-2}, \ldots, P_{t-L})$.

2) Moving average indicator: $I_{MV\_L} = \frac{P_t - MV_{(t,L)}}{MV_{(t,L)}}$, where $MV_{(t,L)} = \frac{1}{L}\sum_{i=0}^{L-1} P_{t-i}$.

3) Trading range break-out indicators: $I_{TBR\_L} = \frac{P_t - P_{\max(1,\,L)}}{P_{\max(1,\,L)}}$, where $P_{\max(1,\,L)} = \mathrm{Max}\,(P_{t-1}, P_{t-2}, \ldots, P_{t-L})$.

Here, several issues need to be pointed out. First, each indicator is associated with a window size, *L*, which is required to calculate a minimum for a filter indicator, a maximum for a trading range break-out indicator, or an averaged level for a moving average indicator. *L* could be assigned a small size (e.g., 5 or 12 time periods) as a short indicator, a middle size (e.g., 22 time periods) as a middle indicator, or a large size (e.g., 50 or 63 time periods) as a long indicator.

Second, each indicator alone only means a ratio depending on the current price and previous *L* number of prices. This reflects the extent of price changing within a selected *L* time period. Notably, only when it is associated with a threshold by means of comparison (e.g. >, < or =), can it exactly function like its corresponding technical rules. This composite is referred to as a *selector*. For example, a selector like "$I_{TBR\_50} > 0.01$", represents a trading range break-out rule with a band of 1%, and a window size of 50 trading time periods. It implies that buy if current price penetrates the maximum price of previous 50 trading time periods by at least 1%. Similarly, we can define a filter indicator I_{Filter_L} and a moving average indicator I_{MV_L} respectively.

Third, note that three types of indicators are derived based on law for generating buy signals. They may be suited to predict upturns of market. In order to predict downturns of market (which is not the target on which we make efforts in this thesis, though), indicators should be modified according to the regulations for generating sell signals. Moving average indicators remain because the same I_{MV_L}, together with a negative threshold and a "<" comparison is able to constitute a selector corresponding a moving average rule for sell. In contrast, both I_{Filter_L} and I_{TBR_L} need to be changed as follows: $I_{Filter\_L} = \frac{P_t - P_{\max(1,\ L)}}{P_{\max(1,\ L)}}$ and $I_{TBR\_L} = \frac{P_t - P_{\min(1,\ L)}}{P_{\min(1,\ L)}}$ respectively. Similarly, a selector can be constituted with one indicator, one threshold, and one comparison. Each selector corresponds to a filter rule or a trading range break-out rule, especially for sell.

Finally, the indicators we derive will be employed as crucial building blocks to generate genetic decision trees. The choice of these is also supported by the analysis of Neftci (1991), who showed that many patterns or trend used in obtaining market signals are almost always related to some sequences of local minima and/or local maxima.

2.5 GAs in Finance

Applications of GAs lie in a broad spectrum of real-life problem domains. Broadly, current existing applications may be classified into three domains: engineering-oriented applications, computer science-oriented applications, and science-oriented applications (Banzhaf et al. 1998).

Engineering-oriented applications include, for instance, online control of real robots (e.g., Howley 1996; Nordin & Banzhaf 1997), design of electrical circuits (e.g., Koza et al. 1997; Koza & Bennett III 1999), and mobile manipulators (Anderson et al. 1992; McDonnell et al. 1992), etc. Computer science-oriented applications involve, for instance, computer animation (e.g., Gritz & Hahn 1997), natural language processing (e.g., Park & Song 1997; Rose 1997), evolving neural networks (e.g., Kitano 1990; Yao 1993), etc. Science-oriented applications, for instance, include biochemistry or medical data mining (e.g., Koza & Andre 1996; Raymer et al. 1996; Bojarczuk et al., 1999), image processing (e.g., Daida et al 1996; Poli & Cagnoni 1997) and applications to finance, which are the theme of this thesis.

GAs application to finance is discussed in this section as the second part of this chapter. Applications of GAs to finance have spread over a diversity of financial subjects, including financial forecasting, trade strategy optimisation, portfolio management, theoretical modelling, and artificial stock markets simulating, etc. Despite the varieties of subjects applicable, the way that they employ GAs mainly relies on two traditional GAs' modes of operation. One mode of operation is based on GAs' ability of learning and adapting to the endogenous context of their environments (Holland 1975; 1992). Another mode of operation is based on GAs' ability of searching and optimising for better solutions (Davis 1991; Goldberg 1989; Yao et al. 1997). The former mode is referred to as *adapting* mode; the latter is referred to as *optimising* mode.

Our review of applications of GAs to finance is carried out along the two lines associated with the two modes.

2.5.1 Applications in an Adapting Mode

In the adapting mode, traditionally, each population element, or *individual* in GAs merely represents a single classification rule. It needs to cooperate closely with other individuals in order to perform well in a frequently varying environment. The selected group of individuals constitutes a solution to the problem solved. In GAs domain, this mode is often closely related to the Michigan approach (Brooker et al. 1989; Holland 1986). It emphasises on co-ordination of each individuals to solve problems whose characteristics are not known in advance with respect to a changing or ill-defined world.

The adapting mode of GAs is usually employed for simulating financial markets or theoretical modelling in order to understand some basic laws underlying in real markets.

The artificial “Sante Fe Stock Market” is a research project at the Santa Fe Institute in American, an interdisciplinary centre for the study of complex systems. A team of researchers, including a finance professor (Arthur, W.B.), an economist (Lebaron, B.), a trader (Tayler, P.), a computer scientist (Holland, J.H.) and a physicist (Palmer, R.) built an artificial stock market to study the emergent price dynamics by means of genetic algorithms (Arthur 1992; Palmer et al. 1994; Taylor 1995).

In the “Sante Fe Stock Market”, investors are represented by agents, each of which holds typical 60 rules in the form of *condition-action*. The condition part consists of a mixture of short-term and long-term information with respect to fundamental factors and technical analysis factors; the action part consists of either *buy* or *sell*. Based on these rules, agents attempt to forecast the movement of prices and making trading decisions. These rules themselves are evolving and developing by adapting to the changing market over time. At intervals, new rules are devised by means of the genetic algorithm approach with reproduction, crossover and mutation. The aggregate behaviour of the agents creates the price series. At the same time, market situations based on the price series determine the behaviour of each agent. Because they both create and exploit the price series, the agents are essentially co-evolving, even though they do not interact directly with one another.

Their results show that the artificial market is able to qualitatively replicate many puzzling empirical features of financial time series, including volume/volatility persistence, and time series evidence showing the profitability of trend-following strategies. Moreover, in their model, phenomena such as bubbles, crashes, and market moods that are common in real markets can emerge. This demonstrates its advantage over most traditional theoretical models of the markets in that markets dynamics can be observed in the process of simulating.

The "Santa Fe Stock Market" is just one of many attempts currently underway to move forward our understanding of the financial markets based on this approach of genetic algorithms.

Izumi and Okatsu (1996) develop an exchange market with artificial adaptive agents, called AGEDASI TOF (**A GE**netic-algorithmic **D**ouble **A**uction market **SI**mulation in **TO**kyo **F**oreign exchange market). Like the "Sante Fe Stock Market," GEDASI TOF acts as a test bed to study the foreign exchange bubble between 1989 and 1991. According to their study, the bubble starts by support of fundamental factors, grows by bandwagon expectations, stops by coincidence of all agents' expectations and collapses by regressive expectations. Unlike the "Sante Fe Stock Market," AGEDASI TOF is also effective for quantitative analysis, in addition to qualitative analysis that both models have. When the model is used to predict exchange rate for different time periods, it outperforms a random walk model and a linear regression model.

Arifovic (1994) investigates a simulated competition market among firms for a single good. The firms make production and sales decisions by means of a genetic algorithm. The algorithm converges to a rational expectation equilibrium for several parameters values, and replicated experimental findings better than the other learning algorithms. Going further, Arifovic (1996) study an overlapping generations model for exchange rates. Agents' decision-making regarding consumption and savings is again encoded via a binary string for GA manipulation. Arifovic finds that the exchange rate does not settle down to any of the known equilibriums in the model, but continues to bounce around. Chen & Yeh (1996) extend this work with genetic programming and show that the market behaviour observed can be explained as well without endowing firms with any prior market-related information.

Examples of similar work are Chen & Duffy (1996), Chen & Kuo (1999), de la Maza & Yuret (1995), Lettau (1997), Marengo & Tordjman (1995), Margarita (1992), Marimon et al. (1990), Nolan et al. (1999), Routledge (1994), Rust et al. (1994), and Vriend (1994), etc. These studies together form a growing area of evolutionary finance, which combines ideas from evolution and learning in order to understand empirical puzzles in financial markets. GA plays an essential role with its inherent mechanisms of learning and adapting to continual changing environments, and of the survival of the fittest in a group of individuals. Evolutionary finance provides new and very different approaches to traditional economic modelling and therefore, reveals very different perspectives on traditional theoretical thinking (LeBaron et al. 1999; LeBaron, 2000).

2.5.2 Applications in an Optimising Mode

In the optimising mode, each individual in GAs represents a set of rules, which works independently. The best individual on training data is often used as a solution to the problem at hand. In GAs domain, this mode is often closely related to the Pittsburgh approach (De Jong et al. 1993; Janikow 1993; Smith 1980, 1983). It emphasises on its global search capability with the similar principle of search that many general-purpose direct-search algorithms also have, such as gradient search or local exchange like hill-climbing, simulated annealing.

The optimising mode of GAs is often employed for trade strategies finding, financial forecasting, and portfolio selecting. Occasionally, as an effective search engine, it is used to attempt some economic modelling tasks as well. The following reviews are represented and discussed separately according to the above four financial subjects.

2.5.2.1 Trade Strategies Finding

Bauer (1994) pioneers the application of GAs to finding market timing strategies. Market timing strategy is represented by a rule with a predefined format encoded into a fixed length binary string. It aims at maximising profits from well-timed movements into and out of various asset classes such as common stocks, treasury bills, and corporate bonds. The trading signals are based on economic variables such as inflation, industrial production rate, unemployment rate, and consumer price index, etc.

For his predefined rule, Bauer designed a kind of primitive predicate as an element. The primitive predicate consists of an economic variable, a comparison operator (e.g., ">",

a *threshold* or *critical value.* The predefined rule can only hold a maximum of *three* primitive predicates through logical combinations. Though the designed primitive predicate is in a good shape, the overall rule generated based on it is limited as the number of primitive predicates is fixed. Consequently, Bauer's system could not find potential better timing strategies that were represented beyond the fixed rule form given. Nevertheless, the idea of using primitive predicates sheds ample light on following research work, such as Mahfoud & Mani (1996), Pictet et al. (1995), Oussaidene et al. (1997), and our work here (see Figure 3.6 at p78).

An interesting aspect in Bauer (1994) is to introduce a novel method, which is referred as "knowledge based hedge". He constructed "hedge portfolio" by going long based on good rules and going short on bad rules. The results showed some promise, although the returns were rather volatile over the three-year test period.

The application of genetic programming to finding trading strategies can be traced to

Karjalainen (1994), which was later on summarised and published in Allen & Karjalainen (1995) and Allen & Karjalainen (1999). In the work, genetic programming is used to infer technical trading rules from the past price. The algorithms were applied to S&P 500. The rules found by genetic programming lead to statistically significant excess returns above the buy-and-hold strategy in the out-of-sample test period of 1970-89.

Two distinguishing aspects of the work are 1) the representation of trading rule as the individual in genetic programming and 2) rigorous statistics tests, i.e. bootstrapping simulation.

The individual, a tree-like structure, consists of several building blocks which belong to either the real-valued block or the boolean-valued block. Real-valued blocks correspond to transformations of past price as well as constants by means of arithmetic functions (+,–, ∗, ÷) and functions that return local extrema of prices ("maximum" and "minimum"). Boolean-valued blocks correspond to transformations of real-valued blocks as well as constants by means of logical functions ("if-then", "if-then-else", "and", "or", "not") and comparison functions (">", "<"). The root node of each rule is restricted to a boolean function, the value of which specifies the trading rule signals, either "buy" or "sell". With all the functions mentioned above as elements for building blocks, the trading rules generated can be much more diverse than those in Bauer (1994) represented by a fixed-length string. Therefore, better solutions are more likely to be found. On the other hand, the fact that Karjalainen (1994) allows genetic programming to construct real-valued blocks, which correspond to pre-selected economic variables or technical indicators in Bauer (1994), means GP explores a larger searching space, which may make genetic programming more difficult to search and cost much time.

Bootstrapping is one of the statistic methodologies adopted in finance. Bootstrapping simulation is used to judge whether the observed performance of a trading rule is likely to have been generated under a given model for the data-generating process. (cf., Brock et al. 1992; Levich & Thomas 1993). Using this methodology by comparison with benchmark models of a random walk, an autoregressive model, and a GARCH-AR model, Karjalainen (1994) finds none of these models of stock returns can explain the findings derived from trading rules found by genetic programming.

Similarly, Neely, Weller & Ditmar (1997) follow Karjalainen (1994) and apply the genetic programming methodology to foreign exchange markets in order to infer technical trading rules as well. Six foreign exchange rates are adopted, including Dollar/Deutsche mark, Dollar/Yen, Dollar/Pound, Dollar/Swiss franc, Deutsche mark/Yen, and Pound/Swiss franc. Like Karjalainen (1994), from their empirical results, they find strong evidence of economically significant out-of-sample excess returns to those rules for each of six exchange rates over the period of 1981-95.

Oussaidene, Chopard, Pictet & Tomassini (1997) have completed similar work on foreign exchange market as well. The most distinctive perspective of the work is that genetic programming is implemented and run through a parallel computer system. Given huge historical data, their method shows its advantage over others, as the system is more likely to generate trading rules faster.

Some hybrid systems, which rely on GAs, together with other AI techniques, have also been examined. Edmonds & Kershaw (1994) combine GP with Fuzzy Logic to generate so-called Fuzzy production trading rules. Promising results are found based on several major price indices, such as S&P 500, NIKKEI, FTSE. Margarita (1991) connects a genetic algorithm to neural networks. Genetic algorithms aim at searching the weights of a recurrent network for trading of the FIAT shares in the Milan Stock Exchange.

2.5.2.2 Financial Forecasting

Mahfoud & Mani (1996) develop a genetic algorithm based system for predicting the future performances of individual stocks. As they claim, the system, in its general form, can be applied to any inductive machine-learning problem or classification. They apply the system to predict directions of the relative returns for more than 1,600 individual stocks. The system is fed with 15 *attributes* or *indicators* representing technical as well as fundamental information about each stock. They report promising results. Moreover, they compare GAs with Neural Networks (NNs) and find that the GAs outperform the NNs on the chosen tasks. They attribute the performance to the fact that GAs abstain from making predictions 27% of the time, while NNs make predictions in nearly all cases. As a result, they argue that a trade-off exits between the number of predictions made and overall prediction accuracy. Though this trade-off is claimed to be a future research topic, they do not give any clue on how to develop a mechanism of adjusting the trade-off and where this mechanism could possibly be embedded into a GA system. In Chapter 5, we shall present a means of adjusting the trade-off between the number of predictions made and the prediction precision, rather than overall prediction accuracy.

White (1998) designs a *Genetic Adaptive Neural Networks* (GANN) where GAs combine with NNs. He concludes that GANN is able to approximate, to a high degree of accuracy, the complex, non-linear option-pricing function, which is used to produce the simulated option prices.

Chen & Lu (1999) employ a genetic algorithm to determine the number of input variables and the number of hidden layers in order to evolve better NNs for forecasting foreign exchange rates of the Dollar/Deutsche mark. After comparing with NNs that are generated based on back-propagation with pre-specified architectures, they find that the best model is the NN evolved by GAs among 16 NN models generated in different designs.

2.5.2.3 Portfolio Selecting

Rabatin (1998) uses a GA to develop an adaptive portfolio trading system. The system generates trading decisions based on its four parts that simultaneously define the portfolio performance: market timing, price risk, portfolio allocation, and portfolio risk. GAs are employed to approach tasks associated with each part. He tested his GA based system to form foreign exchange portfolio. According to his results, the system is able to develop profitable behaviour under the condition of realistic transaction costs.

Vacca (1997) uses a GA to address a common realistic problem faced by any investors who want to hedge their portfolio. The problem is to find a trade-off strategy between the extremes of minimizing risk by frequent re-balancing and minimizing cost by limiting the number of trades. GAs are employed to find portfolio hedge parameters, which determine trading decision. The trading strategy found via his method provides a robust hedging scheme.

2.5.2.4 Economic modelling

Koza (1995a) applied GP to economic modelling. The modelling focuses on a non-linear econometric exchange equation $P=MV/Q$, which relates the money supply M, Price level P, gross national product Q, and the velocity of money V of an economy. Genetic programming was used to discover the above exchange equation from actual observed data.

A large part of work on economic modelling with GP in the optimising mode has been conducted by Professor Chen and his colleagues. GP is applied in connection with *efficient market hypothesis* (Chen & Yeh 1996b; Chen & Yeh 1996c), financial volatility in share prices (Chen & Yeh 1997a), option pricing (Chen & Lee 1997; Chen et al. 1998), and an overlapping generations model related to dynamics of the inflation rate (Chen & Yeh 1997b), etc.

These studies show that GAs can also be applied to economic modelling in the optimising mode.

Chapter 3

GP and Its Use in Financial Forecasting

3.1 Introduction

In Chapter 2, we reviewed some fundamental financial issues relevant to our research. We established that our research stands on firm ground and it is not futile. Furthermore, we discussed some popular technical analysis rules that have been heavily studied in financial literature. Based on their interpretations for predicting market future movement, we derive corresponding technical indicators. These indicators shall be treated as predictors that will be fed to our tool as input variables for financial forecasting.

In this chapter, we focus on technical issues around FGP, a financial forecasting tool based on genetic programming. We begin with some background knowledge with regard to genetic programming techniques. We present basic components in a canonical genetic programming, as well as some advanced techniques in relation to our research. We then present algorithms of FGP using pseudo-codes and describe what mechanisms are incorporated in FGP, in particular, some distinct components we designed especially in FGP for financial forecasting. Finally, we summarise this chapter.

3.2 Literature

Genetic algorithms (GAs) (Holland 1975) comprise a class of search, adaptation, and optimisation techniques based on the principles of natural selection. This places them in the class of algorithms called *evolutionary computation* or *evolutionary algorithms*. Other members in evolutionary computation include *evolution strategies* (ESs) (Rechenberg 1973; Schwefel 1981) and *evolutionary programming* (EP) (Fogel et al. 1966). Genetic programming (Koza 1992) is a variant of genetic algorithms that evolves tree representations instead of strings.

Despite the differences of data structures, these paradigms share a common conception - a population is used to search a space of possible representations. A population of objects compete with each other to perform the task under consideration in search of better solutions. In biological term, the population of possible solutions can be modified in two major ways.

1. Mutation, an asexual reproduction operator, resulting in a minor change in an individual's representation.
2. Crossover, a sexual reproduction operator, resulting in possible new representations by mixing the genetic material composing elements of the population.

The field of evolutionary computation has been widely studied since the 1960s. Recent years, in particular, have witnessed a remarkable and steady (still exponential) increase in the number of publications (see, e.g., the bibliography of Alander (1994), including 3000 GA references, and a web page: http://www.cs.bham.ac.uk/~wbl/biblio/ maintained by Bill Langdon, including 1500 GP references), and conferences in this field. For recent reviews of the state-of-the-art evolutionary computation, the reader is referred to Bäck (1996); Bäck, Hammel & Schwefel (1997); and Yao (1999).

3.2.1 Genetic Algorithms

Genetic Algorithms (GAs), were introduced by Holland (1962 and 1975), subsequently studied by De Jong (1975), Goldberg (1989), and others such as Davis (1991), Koza (1992) and Mitchell (1996). Initially, GAs were proposed as a general model of adaptive processes, but by far, the largest application of the technique lies in the optimisation domain (Bäck et al. 1997). In the financial domain, however, both the adapting mode and the optimising mode are used to a certain extent (see Section 2.5 in Chapter 2).

The standard genetic algorithm proceeds as follows. It operates by iteratively evolving a population of individuals. Each individual is represented by a finite string of symbols (traditionally, binary codes), known as the chromosome, encoding a candidate solution to the problem at hand. An initial population of individuals is generated at random or heuristically. Then, on each iteration, referred to as *generation*, all individuals are evaluated in terms of one pre-specified quality criterion, known as the *fitness function*. A new population is then generated by probabilistically selecting individuals from the current population (this selection strategy is called *fitness-proportionate selection*). Some members in the new population are carried forward from members in the last generation population (*parents*) intact via *reproduction* operation. The rest are generated by applying genetic operators: *crossover* or *mutation*. Such a process continues until sufficiently fit individuals are generated. In practice, the GA process is terminated when either a maximum generation or a maximum running time, both of which are predefined by the user, are satisfied. Its algorithm is summarised in Figure 3.1.

```
1. Create randomly the initial population $P(i)$;  set $i = 0$.
2. Repeat
    a) Evaluate the fitness of each individual in $P(i)$ using fitness function.
    b) Select parents from $P(i)$ using selection strategies.
    c) Generate a new generation P$(i+1)$ using genetic operators (e.g. reproduction,
       crossover, mutation).

3. Until $i < N$ or the time is up; where $N$ is maximum generation set by users.
```

Figure 3.1: A simple genetic algorithm.

The main mathematical foundation for GA is Holland's *Schema Theorem* (Holland 1975). This theory is predicated on survival of the "schemas". Individuals that exceed the mean fitness level of population are more likely to pass on their genes (a detailed description of the schema theorem is given in Appendix A). It was further advocated by Goldberg with *Building Block Hypothesis* (Goldberg 1989). The *Building Block Hypothesis* is typically informally stated that a

GA works by combining short, low-order, and above average fitness schemata (building blocks) to form higher-order ones from generation to generation until it converges to an optimum or near-optimum solutions.

The usefulness of the schema theorem and the building block hypothesis has been argued in academia (see, e.g., Altenerg 1995; Beyer 1997; Fogel & Ghozeil 1998; Grefenstette & Baker 1989; Grefenstette 1992). Such a topic is beyond the scope of this thesis. Interested readers are referred to Salomon (1998). The schema theorem is a widely accepted description of the way that GAs search. Moreover, it has been extended to a theory for GP by defining a concept of schema for parse trees. We refer the reader to O'Reilly (1995), Rosca (1997) and Poli & Langdon (1998).

3.2.2 Canonical Genetic Programming

Genetic Programming (GP), as a machine learning approach, was first clearly defined by Koza (1992). In this work, GP evolved a solution in the form of a Lisp program by way of a similar procedure as GAs. It extends the concepts of the fixed-length representation in GAs with the tree-structured, variable length, and dynamic representation. Such a representation is interpreted as a program. The structure of a program was constructed with a combination of function (arity > 0) and terminals (0-arity function), where the arity of a function is the number of inputs to or arguments of that function. In order to apply GP to a problem, the following setup was required by the user.

1. Define the set of functions, $F = \{f_1, f_2, .., f_n\}$, with arity > 0. Each function in F takes a specified number of arguments, defined as $a_1, a_2,\ldots, a_n$.
2. Define the set of terminals, T = {t1, t2, .., tn} of 0-arity functions (e.g. variable) or constants (e.g. real number, integer).
3. Define the fitness measure used to evaluate how well each program performs the designated task.

4. Define some parameters for controlling the run, such as population size M, the maximum initial depth (depth of program tree), the maximum program depth allowed during the evolving process, and some pre-specified probabilities of genetic operations such reproduction, crossover, mutation, etc.
5. Define the method for designating the result and the criterion for terminating a run.

GP uses an overall approach to creating and evolving tree-based programs similar to GA's and other evolutionary approaches. The following three steps summarise the search procedure used with GP algorithm.

1. Create an initial population of programs, typically randomly generated as compositions of the functions and terminals sets.
2. WHILE *termination criterion* not reached DO
   (a) Execute each program in the population and assign it a fitness value using the fitness measure.
   (b) Create a new population of programs by applying the following operations. Each operation is applied to program(s) selected from the population with a probability based on fitness.
      - Reproduction: Copy the selected program into the new population.
      - Crossover: Create a new offspring program from the new population by recombining randomly chosen parts of two-selected program.
      - Mutation: Create one new offspring program for the new population by randomly mutating a randomly chosen part of the selected programs.
3. Designate the individual program that is identified by result designating method as the result of the run of genetic programming. The result represents a candidate solution to the

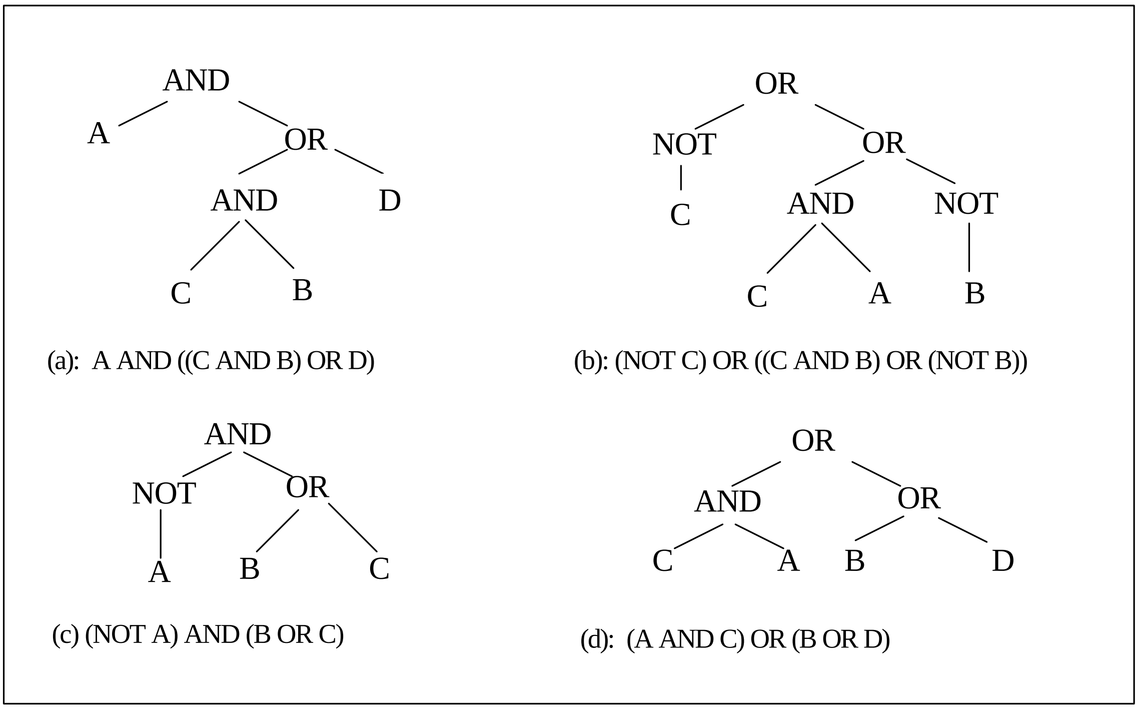
problem.

Some major techniques involved in the above three steps with a canonical GP are described in greater details below.

Creating the initial GP population

The initialisation of a tree structure is fairly straightforward. It starts by selecting a function, f_i, randomly from the set F. For each of f_i arguments, this process is repeated where either a random function or a terminal is to be selected to fill each argument position. If a terminal is selected the generation process is terminated for this branch of function. If a function is selected the generation process is recursively applied to each argument of this function. For example, given the function set $F = \{AND, OR, NOT\}$ and terminal set $T = \{A, B, C, D\}$, several programs that could be generated for the initial population are shown in Figure 3.2. Theoretically, the tree can be built rather large. In practice, a parameter, called the maximum initial tree depth, is specified to limit the depth of the initial tree programs, thereby the size of programs.

Koza (1992) defines two different ways of initialising tree structures, the “full” method and the “grow” method. For a tree generated by the full method, the length along any path form the root to a leaf is the same no matter which path is taken, that is, the tree is of full depth along any path. Trees generated by the grow method need not satisfy this constraint. For example, in Figure 3.2, both (c) and (d) are generated using the full method with a maximum depth of three, whilst both (a) and (b) are initialised with the grow method.



(a): A AND ((C AND B) OR D)

(b): (NOT C) OR ((C AND B) OR (NOT B))

(c) (NOT A) AND (B OR C)

(d): (A AND C) OR (B OR D)

Figure 3.2: Initial GP programs a, b, c, and d, generated based on *F* = {*AND, OR, NOT*} and *T* = {*A, B, C, D*}.

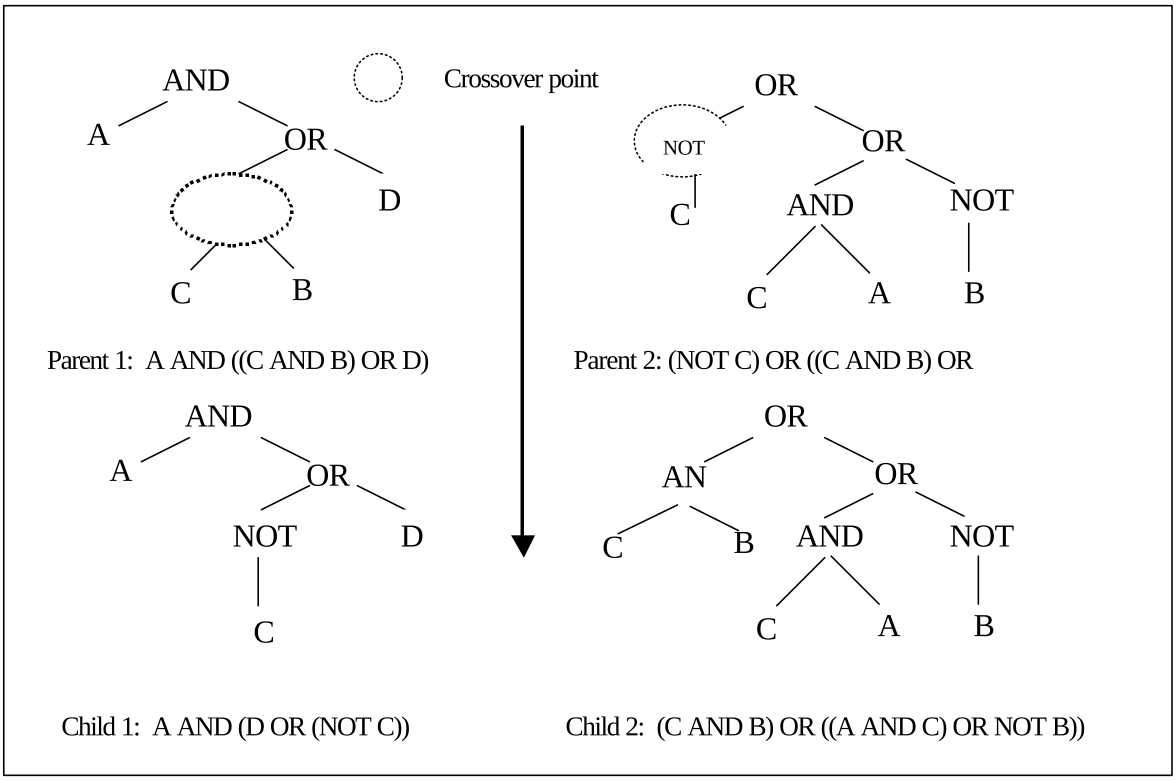
Based on the full method and grow method, *ramped-half-and-half* technique are introduced (Koza 1992). It is intended to enhance population diversity of structure from the outset. In trees the technique is like this. Suppose the maximum depth of tree parameter is 5, the population is divided equally among individuals to be initialised with tree having depths 2, 3, 4, and 5. For each depth group, half of the trees are initialised with the full method and half with the grow method.

Genetic operators

The traditional GP operators are analogous to those used in GAs, but have been adapted to work with trees. Each operator must ensure that the resultant offspring do not exceed the maximum depth of program specified by the user. Reproduction and crossover are considered to be the main genetic operations, whereas mutation is viewed as secondary and used sparingly (Koza, 1992).

Reproduction selects a program based on fitness, and copies this program identically to the next generation.

Crossover selects two parent programs based on fitness, and generates two offspring by swapping sub-trees between two parent programs. The crossover point within each parent is randomly selected, using a normal distribution. An example crossover is shown in Figure 3.3.



Parent 1: A AND ((C AND B) OR D)

Parent 2: (NOT C) OR ((C AND B) OR

Child 1: A AND (D OR (NOT C))

Child 2: (C AND B) OR ((A AND C) OR NOT B))

Figure 3.3: An example of sub-trees crossover.

The common form of mutation is sub-tree mutation. Mutation operates on only one individual tree. A node within the tree is randomly selected and the sub-tree below this node is deleted. A new randomly generated sub-tree replaces the deleted one.

Selection strategy

The selection method is used to select individual program(s) for genetic operators, namely, reproduction, crossover, and mutation. The selection is based on the fitness of each program in relation to other members in entire population.

One of the most common selection methods is *roulette wheel* selection in which the probability of a program being selected is proportional to its fitness value. An alternative common selection method is *tournament* selection. This selection is not based on competition within the entire population but in a subset of the population. A number of individuals, called the *tournament size*, are selected randomly and they compete each other. The best one is to be selected. *Tournament* selection has recently become a mainstream method for selection (Banzhaf et al. 1998). Popularity of this selection method is partly due to its adjustable selection pressure, which is increased or decreased by simply increasing or decreasing the tournament size (Miller & Goldberg 1995).

Termination criterion for GP

A GP run is terminated when a specified maximum number of generations have been reached or perfect fitness is achieved.

Result designation

Koza (1992) uses a *best-so-far* individual as the result of a run of genetic programming. The *best-so-far* individual is the best individual that ever appeared in any generation of the population.

Sufficiency and closure

Koza (1992) states that two requirements may be necessary before GP could be applied to a specific problem. The concept of *sufficiency* states that the functions and terminals (in combination) must be capable of expressing a solution to the problem. The concept of *closure* states that "*each function in the function set be able to accept, as its arguments, any value and data type that may possibly be returned by any function in the function set and any value and data type that may possibly be assumed by any terminal in the terminal set*" (see, Koza 1992; p81). Briefly speaking, closure is a condition that all the arguments for functions, and values returned from functions, must be of the same data type. Thus, closure allows unrestricted composition of the available functions and terminals in the program trees.

As a requirement, closure is easily satisfied when merely one type of function is involved for generating program trees (for example, either boolean or mathematical functions are merely adopted). However, if program solutions need to be represented with a combination of different function types (e.g., boolean functions {AND, OR, NOT} and mathematical functions {+, −, ×, ÷}), to satisfy closure is not possible. To correct this deficiency, several syntactic constraint techniques have been developed. These techniques eliminate the closure constraint necessary for traditional GP, and hence improve GP on its expressive capability. A concise discussion on this issue is given in the following section.

3.2.3 Advanced Genetic Programming

The publication of the book (Koza, 1992) is a milestone in the field of genetic programming. Koza's studies provide a basis for further development of GP. Substantial work, since then, has been conducted in numerous aspects for improving traditional GP techniques.

Two aspects attract a considerable number of GP researchers. One is to develop syntactic constraint techniques by means of grammar or typing mechanism. This aims at improving tree representation by relaxing the closure condition in traditional GP. The other is to provide a modularisation mechanism by means of encapsulating useful components in overall program trees. This intends to adapt GP representations particularly for solving large and more complex problems. Note that both syntactic constraint techniques and modularisation mechanism intend to enhance the expressive power of evolvable programs.

3.2.3.1 Syntactic Constraints

As discussed earlier, *closure* condition is not held if the adopted function set consists of members belonging to different types. This problem was noted by Koza (1992), where he proposed an informal *syntactic constraint* approach to attacking the difficulty. The proposed method made it possible to construct valid programs without abiding by the closure condition. Several examples were given to illustrate how the method worked and where the program structure had to be constrained (see Chapter 19 in Koza 1992).

In one example, which addresses the Fourier series problem, Koza defined three syntactic constraints as follows:

- The root of the tree must be the special function, &, which was stated as the ordinary arithmetic addition function with two arguments.
- The functions allowed immediately below an & must be a xsin, xcos, or & function.
- The functions allowed below an xsin or xcos function is either an arithmetic function (+, –, ×, %) or a random, real-valued, constant.

Using English to describe these constraints worked adequately for the Fourier series problem. However, this method appeared to be an ad hoc approach that would need to be modified for each new problem. In addition, the way of stating constraints using an English explanation may not be in a proper declarative manner. One of the better methods to provide a formal specification of syntactic constraints is to use grammar.

Syntactic constraints using grammar

Grammar is a set of rules used to specify the syntax of a language. The class of *context-free* grammar is the most popular, as they are simple and yet widely applicable to many problems (Chomsky 1956). *Backus Normal Form* (BNF) (Backus 1959) is one of the forms used to represent context-free grammar. The grammar for generating program trees of GDTs in FGP system is specified in BNF. Some other GP work using context-free grammar to deal with syntactic constraints is briefly discussed as follows.

A more general way of representing syntactic constraint was presented by Gruau (1996). The syntactic constraints are represented by context-free grammar. By virtue of the rewriting rules provided in the grammar, a valid GP tree can be generated recursively by starting from rewriting the axiom of the grammar. Following is an example of such context-free grammar for generating a boolean expression in *Disjunctive Normal Form* (DNF).

```
<axiom> :: = <DNF>
<DNF> :: = or (<term>) (<DNF>) | <term>
<term> :: = and (<literal>) (<term>) | <literal>
<literal> :: = <letter> | not (<letter>)
<letter> :: = A | B | C | D
```

Figure 3.4: An example of context-free grammar.

In this example, <axiom>, <DNF>, <term>, <literal> and <letter> are all non-terminals that must be rewritten recursively, whereas three logic operators (or, and, not) and four terminal letters (A, B, C, D) are terminals that do not need to be rewritten. Various programs can be generated in accordance with this grammar; “or (and A B) D” and “or (and C D) (not A)” are two instances. The grammar we adopt for generating GDTs is similar to this, but it strictly complies with BNF.

Whigham (1996) also employs context-free grammar in a genetic programming system. The context-free grammar is represented by a four-tuple (*N*, *S*, *P*, *S*), where *N* is the alphabet of nonterminal symbols, *S* is the alphabet of terminal symbols, *P* is the set of productions and *S* is the designated start symbol. The productions are of the form x → y, where x is a member of *N* and y is any composition of symbols from $\{S \cup N\}$.

To illustrate the declarative nature of this grammar, Koza's syntactic constraints previously described in the England language could be represented by the grammar $G_{syn\text{-}con}$.

$$
\begin{aligned}
&G_{syn\text{-}con} = \{S, N = \{\mathrm{A}, \mathrm{T}\}, S = \{\ \&, xsin, xcos, +, -, *, \%, \quad \},\\
&\qquad \mathrm{P} = \{S \ \circledR \ \ \& A\, A\\
&\qquad\qquad A \ \circledR \ \ \& A\, A \mid xsin\ T\, T \mid xcos\ T\, T\\
&\qquad\qquad T \ \circledR \ \ + T\, T \ \mid T\, T \ \mid T\, T \ \mid T\, T \ \mid\\
&\qquad\qquad \}\\
&\qquad \}
\end{aligned}
$$

Based on context-free grammar, Whigham investigated the language declarative bias. According to his study, the language bias can be easily defined via this context-free grammar. An explicit language bias provided by context-free grammar has the following advantages over the standard genetic programming framework. It can provide an unambiguous statement of the arity, typing constraints, and overall structure of the components that would describe the solution. Moreover, the form of the initial population of programs may be explicitly biased, reflecting the belief of the user that certain components of the language are more likely to be important.

Whigham's assertion regarding the language bias is convincing. We take this point when we develop FGP system in this thesis. Context-free grammar is adopted for constructing program trees, GDTs (*genetic decision trees* in FGP). The grammar reflects the belief of structures of potential patterns that would have predictability to future market movement. Rather than using Whigham's four-tuple representation, the grammar we take is represented in BNF.

Strong typed genetic programming (STGP)

The issue of syntactic constraints has also been investigated by Montana (1995), who proposes a generalization of Koza's constrained syntactic structures, called *strong typed genetic programming* (STGP). STGP eliminates the closure constraint and hence allows the user to define functions which take any data types as arguments and return values of any data type. This is achieved by specifying the required argument types for each function and the return type of each function and terminal.

The advantage of using type constraints is arguably to be able to reduce the size of search space, and make resultant programs easier to understand. Due to type constraints, the number of possible programs that may be formed is reduced compared with without type constraints. Consequently, the likelihood of discovering a program solution with some time and a reduced space is increased. For example, Haynes et al. (1995) demonstrated that STGP outperformed standard GP for the problem of evolving cooperation strategies in a predator-prey environment. They attributed the improved performance to the reduced search space, which resulted from the typed system. They also showed that programs generated by STGP tend to be easier to understand.

Nevertheless, this typing mechanism has its weaknesses. It only enforces a simple level of relationship between one function or argument and another function or argument. Compared to context-free grammar technique, typing alone cannot represent structural constraints beyond that simple level of relationship (Whigham 1996).

Structure-preserving operations

No matter what kind of syntactic constraint techniques one uses, the way of creating an initial population and performing genetic operators should be adapted accordingly. The initial population of random individual must be created in accordance with the syntactic rules given in constraints. Structure-preserving crossover and mutation must be adopted to ensure that offspring generated conform to the syntactic constraints.

Structure-preserving crossover is usually achieved with the following steps:

1. To select a crossover point, referred to as P_{c1}, randomly in the first parent.

2. To select a crossover point that has the same type as that of P_{c1}, from the second parent.

3. To perform a normal crossover operation on parent 1 and parent 2.

Structure-preserving mutation is similar. A mutation point is randomly selected in an individual program. The sub-tree below the mutation point is replaced by a generated sub-tree, whose type must be identical to that of the deleted sub-tree.

3.2.3.2 Modularisation in Genetic programming

The standard GP paradigm has no explicit mechanisms for creating modules and reusing them. According to the Building Block Hypothesis, programs with shorter effective length have better chances of survival compared with programs with larger effective length. All modularisation techniques are arguably assumed to be able to encapsulate those effective shorter programs. Such encapsulated shorter programs become subroutines and can be called repeatedly from the main program or form other subroutines. Various modularisation techniques have been proposed, including *Automatic Defined Functions* (ADFs) (Koza 1994), *Automatically Defined Macros* (ADMs) (Spector 1996), and *Module Acquisition* (MA) (Angeline & Pollack 1992).

Automatic Defined Functions

ADFs are the most thoroughly evaluated method. It is the subject studied in Koza's second GP book (Koza 1994). The individual program with ADFs consists of two parts or branches:

1. The *result-producing branch*, from which the fitness of overall program is calculated; *and*

2. The function-defining branch, which contains definitions of one or more ADF.

Each ADF is a complete subroutine, requiring a definition of the arguments, functions, and terminals from which it is composed. The main program body (i.e. the result-producing branch) is allowed to call any ADF with arguments defined from the terminals and function set. For further knowledge about ADFs, the reader is referred to Koza (1994).

The use of ADFs has been empirically shown to be effective in numerous applications and domains (see, e.g., Koza 1994; 1994a; Handley 1994; Kinnear Jr., 1994a). Nonetheless, a major weakness of the ADF approach has also been found in that the structure of the overall program has to be specified by the user beforehand. Defining the structure involves selecting the number of function-defining braches in the overall program and the number of arguments (if any) possessed by each function–defining branch, etc. Once these parameters are specified, each program in the population has the same structure. Therefore, GP has no ability to explore more potential structures. In order to overcome this limitation, Koza (1995b) designed six architecture-altering operations by which program structures can be changed during GP runs. Although initial results have demonstrated that the approach is promising (Koza 1995b; 1995c), significant results using those operations have not yet been reported.

In a testing version of FGP, we adopted ADFs. In our numerous experiments, however, we did not find any improvement compared against the one without adopting the ADF approach. Thus, we shall not discuss such a version further in this thesis.

Automatically Defined Macros

Spector (1996) has proposed a variant of ADFs called Automatically Defined Macros (ADMs). A common macro transformation is *substitution*, where frequent code fragments in program are replaced by macros. Spector shows how substitution macros can be evolved simultaneously with the main program in a way similar to the ADF method. The evolved macros can be treated as special control structures, producing, for example, specific forms of iteration or conditional execution.

The ADMs approach was evaluated on the obstacle-avoiding robot problem and the lawnmower problem. While the AFMs method demonstrated its advantage for the obstacle-avoiding robot problem over the ADFs method, it did not perform better for the lawnmower problem based on his experiments.

Module Acquisition

Module Acquisition (MA), proposed by Angeline and Pollack (1992), is another approach to modularising code for reuse. Module acquisition acts on individuals. A subtree that is chopped from the chosen individual is defined as a module. This operation is called *compression*. This creates a new function, with arguments based on the branches that have been cut. The created module or function is put into a library of modules from where it can be referenced by other individuals in the population. *Expansion* is another operation, which takes a module and substitutes it back into a program which is using the module. Module acquisition provides the desirable feature of allowing useful blocks of a program to be held and used by many different programs at the same time. However, in one comparative study (Kinnear Jr. 1994), the use of MA does not show any obvious advantage for the problems tested.

3.3 Algorithms in FGP

In this thesis, we introduce FGP (Financial Genetic Programming), a GP that we develop especially for financial forecasting. Some background knowledge on genetic programming related to FGP has been discussed in the preceding section. This section describes techniques adopted in FGP and some distinct components provided in FGP.

An overview of FGP algorithms shall be presented in Section 3.3.1 with pseudo-codes, together with some major parameters that is required for running FGP.

Designing a genetic programming system for financial forecasting involves a number of issues. Two essential issues are the representation and evaluation of program trees. The performance of genetic programming depends crucially on the choice of representation and the choice of fitness function (Mitchell 1997). In the case of financial forecasting, program trees represent potential predictive patterns that are possibly of value to the user. The effective representation would be advantageous for finding promising predictive patterns. The fitness function defines the criterion to assess how well the found patterns perform. From a viewpoint of the user, a fitness function provides him/her with a way to communicate their intention to the process of GP. That is to say, modification of the fitness function can results in desirable patterns or rules, which may meet the preferences of the user. On the other hand, from the search point of view, an appropriate fitness function may change the landscape of GP search space, and therefore may direct GP to explore some promising space more thoroughly. Details in FGP related to these two issues are discussed in Section 3.3.2 and Section 3.3.4 respectively.

Section 3.3.4 briefly describes a hill-climbing method for adapting numeric constants contained in GDTs. The reason of using this method in FGP is also given.

As required in a tool, some useful interfaces and some facilities are provided in FGP system. Detailed implementations are explained in the final section.

3.3.1 Overview of FGP

FGP builds on the framework of the standard genetic programming. It adopts major components that a standard genetic programming normally possesses. For example, for creating the initial population, it takes the approach of ramped-half-and-half. Genetic operators provided in FGP include reproduction, crossover, and mutation. Two selection strategies, namely, *roulette wheel*, and *tournament*, are supplied in FGP. Termination criteria given by FGP are the maximum number of generations or the maximum time that FGP is allowed to execute. The termination criterion of perfect fitness having been achieved is not supplied in FGP, partly because FGP is mainly used for financial forecasting, and it is hard to find perfect matching rules even over training data. FGP uses the best-so-far rule as the result of an individual run.

```
Procedure FGP ( )
Begin
Partition whole data into training data and testing data;
      /* While training data is employed to train FGP to find the best-so-far-rule; the test data is used
         to determine the performance of predictability of the best-so-far-rule */
Pop ← InitializePopulation (Pop);     /* randomly create a population of decision trees. */
Evaluation (Pop);                     /* calculate fitness of each individual in Pop */
Repeat
        Pop ← Reproduction (Pop) + Crossover (Pop);   /*new population is created after genetic
          operators of reproduction which reproduces $M*P_r$ individuals and crossover which creates $M*(1-P_r)$ individuals.
          In our case $P_r=0.1$, M is population size */
        Pop ← Mutation (Pop);                 /*apply mutation to population */
        Evaluation (Pop);
Until (TerminationCondition( ))          /* determine if the termination condition is fired */
Apply the best-so-far rule to the test data;
End
```

Figure 3.5: The pseudo-code of FGP algorithms.

The overview of FGP algorithm is presented by the pseudo-code in Figure 3.5. The overall procedure in FGP is similar to that of a canonical genetic programming. It is not necessary to reiterate the procedure here. Some major parameters required for running FGP are listed in Table 3.1. The listed parameters are changeable by the user. Values shown in the brackets in the table are usually taken in our experiments.

Variables	**Abbr.**	**Type**	**Specification**
PopulationSize	M	Integer	The number of individuals in one population (500 - 2000)
GenerationSize	Gen	Integer	The maximum of number of generation (30 -100)
RunTimeAllowed (Minutes)	RTA	Integer	The running time FGP is allowed to execute
ProbablityCrossover	P_c	Real	The probability of crossover (90 - 95%)
ProbablityReproduction	P_r	Real	The probability of reproduction (5 - 10%, which is determined by the formula (1- P_c))
ProbablityMutation	P_m	Real	The probability of mutation (0.1 - 1%)
ProbablityOptimization	Po	Real	The probability of Optimisation (0.1 - 1%)
InitialMaxTreeDepth	IMD	Integer	The maximum depth of tree for initially generated individuals (2 - 6)
SelectionStrategyFlag	SSF	Boolean	IF (SSF= TURE) THEN *roulette wheel* selection strategy is chosen ELSE *tournament* selection strategy is chosen (SSF=FALSE)
MaximumTreeDepth	MTD	Integer	The maximum depth of GP tree which is allowed to exist (17)

Table 3.1: Some major parameter required for running FGP (values shown in brackets are default values that are usually taken in our experiments).

3.3.2 Grammar-Based Representation

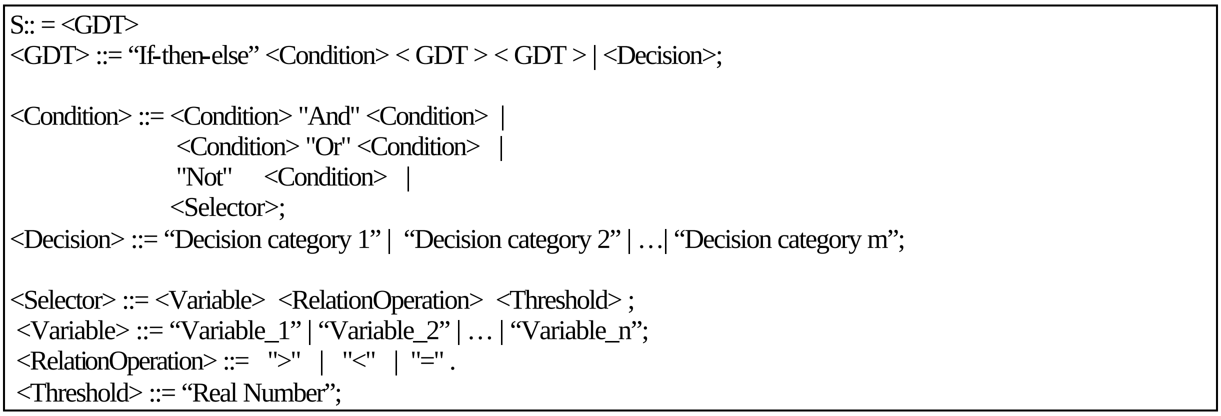
As presented in Section 1.5, several design goals have been set up for developing FGP. We hope that FGP should allow users to channel their knowledge into the process of decision tree generation. Moreover, the GDTs generated by FGP should be comprehensive to users. Both design goals are closely related to the issue of tree representation.

In order to achieve the first design goal, there should be some mechanisms that allow users to feed into the tree structures some features and variables that they may think are useful. As with DRI (Decision Rule Induction) methods such as ID3, C4.5, usually a feature or input variable (its type may be ordinal, discrete or continuous) is associated with a value, also called a threshold, by means of a relation operator (e.g., >, <, =). This constructs a decision primitive. The return value of a primitive is boolean, i.e. *True* or *False*. Multiple primitives combine one another by means of conjunction or disjunction to form various predicates required in an overall tree (this is also can be treated as a set of rules). With this idea in mind, syntactic constraints are required to provide the mechanism of creating the primitive form. In FGP, we call this primitive a *selector* (e.g. "variable_1 > 2.3", please refer to formal definition in Figure 3.6).

Many machine learning algorithms produce production rules as outputs. Such algorithms include AQ14 (Mozetic, 1985), ID3 and C4.5 (Quinlan 1986a; 1993), GABIL (De Jong et al. 1993), etc. Production rule representations are easy to understand. Therefore, in order to achieve the second design goal, it would be a better choice to make GDTs have similar forms as production rules.

To do that, we force the root node of every GDT representation to be an "if-then-else" node which requires three branches: namely, "condition" branch, "then" branch, and "else" branch (theoretically, the root node could also be a decision category; e.g., a positive position or a negative position, so that the whole production rule is a single prediction node). A condition branch consists either of a single selector or multiple selectors. Multiple selectors interact with each other by means of one of logic operations {“and”, “or” or “not”}. When the “then” branch branch is constructed, a single decision category could be selected. Then the branch expanding is halted. Alternatively, another “if-then-else” node could be chosen as the root for the branch so that the branch expanding continues. This recursively building process may lead to as much complicated trees as required. In practice, a parameter called the maximum initial tree depth is employed to limit the tree expanding.

In terms of the principle described above for constructing a valid GDT that is of interest to us, syntactic constraints are required. We prefer to use the *Backus Normal Form* (BNF) (Backus 1959) to present context-free grammar for specifying these syntactic constraints. Figure 3.6 shows the BNF grammar that FGP uses for building a GDT.



```
S:: = <GDT>
<GDT> ::= “If-then-else” <Condition> < GDT > < GDT > | <Decision>;

<Condition> ::= <Condition> "And" <Condition>  |
                 <Condition> "Or" <Condition>   |
                 "Not"     <Condition>   |
                <Selector>;
<Decision> ::= “Decision category 1” |  “Decision category 2” | …| “Decision category m”;

<Selector> ::= <Variable>  <RelationOperation>  <Threshold> ;
 <Variable> ::= “Variable_1” | “Variable_2” | … | “Variable_n”;
 <RelationOperation> ::=   ">"  |  "<"  |  "=" .
 <Threshold> ::= “Real Number”;
```

Figure 3.6: The BNF grammar that FGP uses for constructing GDTs (where variables are input features based on the choices of the user).

The symbols in pointed brackets are the non-terminals, whereas the symbols in quotation marks are the terminal nodes. The rule of S ::= <start symbol> defines the starting node of derivation tree. A BNF-rule like <non-terminal symbol> :: = derivation_1 | derivation_2 | … | derivation_n, defines the all possible derivations (or subtrees) for this non-terminal symbol. Decision is an integer, representing a class. Since we allow FGP to deal with two-class classification mainly or three-class classification occasionally (see Section 4.2.3.1), the value of m is given of 2 or 3. A simplistic example tree built using the above BNF grammar, is illuminated in Figure 3.7.

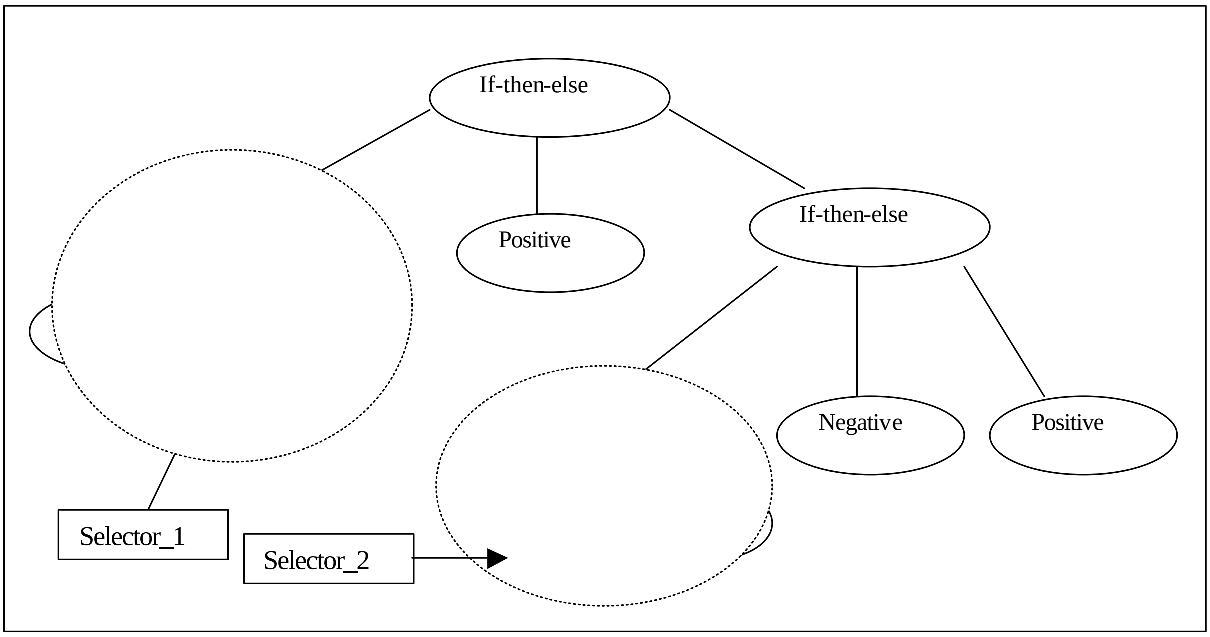


Figure 3.7: A simplistic GDT derived based on the FGP BNF grammar.

In this GDT, there are two “if-then-else” nodes, two selectors embedded with two different input variables, and three decision leaf nodes, with a decision of either positive or negative. Each path from the root of this GDT to a leaf decision node gives one production rule. The left-hand side of the rule contains all conditions that are interactions of all selectors involved by the path, and the right-hand side specifies a decision category at the leaf. Therefore, three individual production rules can be generated as follows.

- Rule_1: if (Selector 1= *True*) then *Positive,*
- Rule_2: if (Selector_1= False AND Selector_2 = True) then *Negative*,
- Rule_3: if (Selector_1= False AND Selector_2 = False) then *Positive*;

where selector_1 = (Var_1 > 16.5), Selector_2 = (Var_2 < 6.6).

The overall ruleset represented by this GDT is (Rule_1, Rule_2, Rule_3) with order. It is worth noting that selectors and their interaction structures are important for the success of a GDT.

With the above grammar-based presentation, FGP allows users to input variables that they think are relevant to the problem to be solved. More importantly, FGP makes selectors by means of selecting a variable and finding an appropriate threshold (see Section 3.3.3). Furthermore, FGP combines selectors into a tree. The interaction structures of these selectors, in fact, become the left-hand sides of a set of production rules with order.

3.3.3 A Hill-Climbing Method Embedded

As mentioned in the preceding section, a selector is an important element, which contributes a lot to success of the overall tree. A selector with a proper threshold would be advantageous. FGP BNF grammar shown in Figure 3.6 is able to make a selector to be created as required, including an input variable, a relation operation, and a random threshold. However, the grammar provides no means of finding an appropriate value for the threshold.

Canonical genetic programming also suffers difficulty in discovering proper numeric constants for the terminal nodes in trees. This is partly because existing genetic operations, such as crossover or mutation, affect only the structure of the trees, not the composition of the nodes. The numeric constants in nodes thus cannot benefit from them.

An early simple approach to facilitating the creation of constants is to use the ephemeral random constant, $\Re$, proposed by Koza (1992). In creating an initial population, each time the ephemeral random constant is selected as a terminal, it is replaced by a randomly generated number within some specified range. Thus, many different numeric constants are available. This method is not sufficient as no further action of changing the numeric constants is taken beyond the generation 0.

Recently, an improved method called *numeric mutation* has been proposed by Evett & Gernandez (1998). Numeric mutation replaces all of the numeric constants with new ones in the individual. The new numeric constants are chosen at random from a uniform distribution within a specific selection range, which is defined as the old value of that constant plus or minus a *temperature factor*. The temperature factor is calculated based on the fitness of the best individual of the current generation. The purpose of using the temperature factor is to control the extent of changing the constant. When the best individual of a population is a relatively poor solution, a larger selection range is applied so that a great potential for change in the numeric constants of the individual is allowed. In contrast, over successive generations, as the best of generation tends to improve, the temperature factor decreases. A smaller selection range is applied so that a smaller change is permitted. This method has been tested in several symbolic regression problems. Statistically significant improvements over traditional GP have been found.

Inspired by the above studies on discovering useful numeric constants in GP trees, we take a simple *hill-climbing* method specifically for finding appropriate thresholds of all selectors contained in a GDT.

The method starts with taking a means of using the ephemeral random constant. All thresholds required in selectors for each GDT are initially assigned the same symbol $\Re$ at the generation 0. Then, in each selector, $\Re$ is to be replaced by a random value that is chosen randomly from a uniform distribution within a specific range. The range is determined by the minimum and maximum of the input variable in the selector.

Over successive generations, a portion of individuals is selected based on the selection strategy. Aimed at finding suitable thresholds in all selectors contained in each individual chosen, we carry out a process called *threshold optimisation* using a *hill-climbing* search technique. First, threshold optimisation needs to enumerate all selectors in the GDT that is to be optimised. We locate these selectors by conducting a *depth-first search* procedure. Second, with all other nodes fixed, a hill-climbing search technique is applied to the threshold. This is aimed to find a possible better value that may results in an enhanced GDT with a better fitness. This is repeated for all the remaining selectors in the GDT.

According to the experiments we have done, the hill-climbing method that we use is useful, in that it is capable of augmenting fitness values to majorities of individuals to which it is applied. This is partly because selectors play an important role in GDTs. Meanwhile, an appropriate threshold is of importance for a selector to be effective.

3.3.4 Fitness Function

As mentioned early, the fitness function is one of the most important factors that affect the performance of a GP system. It imposes its influence by way of the selection strategy, and the result designation method. Whilst GP evolves at each generation, GP must select which members of the population should be subject to genetic operators such as reproduction, crossover, and mutation. This task is completed by a selection strategy, which aims to choose individuals based on values, calculated from a fitness function. At the end of a GP run, the fitness values are used to choose the best individual from the population as the result. In essence, the goal of the fitness function is to guide GP to seek better individuals that contain proper interaction structures with suitable contents incorporated.

The choice of the compositions for constituting a fitness function depends on what kind of the problem one needs to solve. It is common that the fitness function is specified by a single objective function with merely one criterion. Examples of this kind of the fitness function include the number of hits for solving the even-n-parity problems (Koza 1992); the sum of squared differences between actual output and the output generated by the GP tree based on training data (Banzhaf et al. 1998) for approaching the symbolic regression problems; and the number of correctly classified examples in a classification task, etc.

In the case of the applications of financial forecasting addressed in this thesis, intuitively, prediction accuracy is considered as the fitness function. More specifically, a fitness function called the *Rate of Correctness* (RC = $\frac{\text{number of correct predictions}}{\text{total number of predictions}}$) is taken in some applications that will be presented in Chapter 4.

Although a fitness function with one criterion is sufficient in some cases, it may not be suited to solving problems in other cases, where multiple criteria may need to be considered. It is also common that the fitness function is defined with multiple objective functions. Many multi-objective optimisation problems need such a fitness function. In the case of financial forecasting, higher prediction accuracy is always desirable. However, it may not be available. Thus, while maintaining a reasonable level of prediction accuracy, to achieve a low rate of failure or to reduce missing investment opportunities may also be desired. It is often difficult to make trade-offs between these conflicting objectives.

Classical methods to deal with multi-objective optimisation problems are to aggregate the multiple objectives into a single, parameterised objective function. The weighted method is one of representatives. This method converts multi-objective problems to a single objective problem by forming a linear combination of the objectives.

In order to achieve the second research goal, *Goal 2* (see Section 1.4), which is to obtain a low rate of failure, we adopt the weighted method to generate the fitness function. The fitness function consists of several weighted factors. However, it is still not sufficient and effective to attain *Goal 2*. Eventually, a novel constraint is put into the fitness function (we call such a fitness function *the constrained fitness function*). The constraint is able to change the landscape of the search space which may allow the discovery of a solution that would otherwise have been overlooked. The effectiveness of the constrained fitness function is demonstrated in our numerous experiments, which shall be presented in Chapter 5 in depth.

3.3.5 Implementation as a Tool

The kernel of FGP algorithms is implemented using Borland C++. In order to promote the FGP algorithms into a practical tool which can help us with the research, we build numerous useful interfaces around the kernel of FGP. Considering spreadsheets like Microsoft Excel are popularly used by investors, we build all interfaces on top of the spreadsheet. Interfaces are associated with corresponding Macros programmed using *Visual Basic Application* (VBA) for Excel.

Interfaces that we build have three parts. This first part is related to pre-processing data and building up running environments for the tool. This includes creating relevant indicators; setting up running environments as to which directory is used to store necessary files. All experimental data are placed in one spreadsheet. Each column represents either an input variable (an indicator in our case) or a predefined class. Each row represents one of sample cases.

The second part is used for setting up FGP running parameters. For example, parameters required for running a canonical GP system have to be specified, including the population size, the maximum generation allowed to run, the probability of crossover and mutation, the selection strategy, and others. Moreover, some parameters especially required by FGP also have to be given. These include which kind of the fitness function to be chosen, the weights to be assigned if the novel constrained fitness function is selected, and the probability of applying *threshold optimisation* using the hill-climbing method, etc.

The third part focuses on post-processing results. For example, after the GDT is generated, interfaces provided allow the user to apply the GDT to any unseen data and generate corresponding predictions. Other important interfaces are also supplied, including calculating performances of the GDT according to some specified criteria, obtaining some statistics for a resultant GDT and displaying the GDT.

All experiments reported in this thesis were carried out by using this FGP tool. So far, it is reliable and friendly from our experiences.

3.4 Summary

In this chapter, we have described major components that a canonical genetic programming normally has, including the ramped-half-and-half approach for generating the initial population; genetic operators: reproduction, crossover and mutation; two selection strategies: roulette wheel and tournament; and the criteria for terminating a GP run. Furthermore, we discussed two advanced GP techniques: syntactic constraint techniques and modularisation mechanisms, both of which are intended to enhance the expressive power of evolvable programs in GP.

FGP builds on the framework of a canonical genetic programming. By taking ideas in the advanced GP techniques presented in this chapter, it introduces some distinctive components, including

- Grammar-based representation.

Sparked by the syntactic constraint techniques, we take specific grammar (cf. the BNF grammar in Figure 3.6) to generate GDTs that we think are appropriate for solving financial forecasting problems in this thesis. Using this grammar, selectors associated with input variables are formed as primitives. GDTs eventually are created by the combining these selectors in the form of either conjunction or disjunction. GDTs generated can be treated as a set of production rules which are comprehensible to the user. This is a desirable factor in the case of financial forecasting, as the user almost always prefer to know the insight of decisions.

- The hill-climbing method for adjusting real number terminals in selectors.

A canonical genetic programming seldom makes the effort to search for proper numeric constants for the terminal nodes in trees. However, in the GDTs we build, the numeric constants which exist in selectors are important thresholds, which may affect the performances of the

GDTs to some extent. The hill-climbing method is especially employed to adapt the values of these thresholds to the problems to be solved. Our empirical results show that the performances of the majority of the GDTs to which this method is applied are improved.

- A novel constrained fitness function.

A novel constrained fitness function is developed especially in FGP for financial forecasting. This novel constrained fitness function shall be elaborated in Chapter 5 and discussed further in Chapter 6.

Around FGP algorithms are interfaces and facilities that allow the user to manipulate data (e.g., training data, test data), change parameters (e.g., population size, crossover rate, mutation rate, etc.), and assess the performances of a resultant GDT easily.

In the chapters to follow, we shall employ FGP to attack financial prediction problems. We would like to see whether FGP is of help to the user. Studies of the effectiveness of FGP shall be conducted in the light of two goals set up in this research: 1) to improve the accuracy of given predictions; and 2) to improve predictive reliability by reducing the rate of failure.

Chapter 4

Financial Forecasting Using FGP-1

4.1 Introduction

In the preceding chapter, we presented the framework of FGP and its algorithms. We now turn our attention to showing how FGP can be employed to approach financial forecasting.

In this chapter, we shall present our initial applications of FGP. The primary purpose of the use of FGP is to see whether it is capable of improving financial forecasting over base predictions available with respect to prediction accuracy. In what follows, we would like to call FGP for this purpose FGP-1. Such applications are motivated by the two facts: 1) given a set of base predictions, there are sometimes opportunities to improve on them by combining them *and* 2) even a slight improvement in finance prediction could be worth a lot (Colin 1994; Leinweber & Arnott 1995).

Base predictions are available to users from different sources. For example, ordinal forecasts concerning market trends are provided in newspapers by a finite number of experts regularly (Fan et al. 1996); a number of predictions for buying or selling in stock markets can also be generated by different technical trading rules in terms of their regulations respectively. Meanwhile, base predictions can also come from a similar source. For example, independent base classifiers are generated over a number of partitioned instance space (Chan & Stolfo 1996); based rules are obtained from independent trails of the GA over the same instance space (Mehta & Bhattacharyya 1999); multiple individuals in a population evolved in evolutionary learning can also be treated to be able to produce base predictions as well (Yao and Liu 1998). The question is how to make use of them in order to achieve better predictions.

The studies in this chapter are to examine whether and how FGP-1 can be used to generate more accurate prediction than the best individual base prediction available. Two cases of base predictions are investigated here. The first one (Case A) is that base predictions are ordinal forecasts given (Tsang & Li 1998). The second case (Case B) is that base predictions come from a number of non-adaptive technical analysis rules in their normal usages (Li & Tsang 1999a). The conception of non-adaptive technical analysis rules in their normal usages shall be explained later on (see, Section 4.3.2). Prediction problems involved are the weekly movements in the Hong Kong stock market and some research prediction problems of P_n^r with different choices of n and r.

Given a finite number of base predictions for a prediction problem, FGP-1 shall be applied to generate *genetic decision trees* (GDTs) either by merely combining them in the case A, or by combining them with adapting in case B. For both exercises to be of value, the hope is that GDTs should generate better prediction than the best of available base predictions in terms of prediction accuracy.

In order to show that FGP-1 is a useful tool under the above two different cases, two sets of experiments are conducted. One set of experiments serves the study of case A, where the base predictions are ordinal forecasts given; another set of experiments serves the study of case B, where the base predictions come from a number of technical analysis rules in their normal usages. Applications of FGP-1 for both case A and case B are described in Section 4.2 and Section 4.3 respectively. Moreover, for case B, we compare FGP-1 against both the random walk model and C4.5 (Quinlan, 1993). We shall show comparative results. For each case, we illustrate via two examples. Our depiction follows a similar organization:

- Introduction

- The specific representation used
- Two illustrative examples

Finally, in Section 4.4, we shall discuss what we have achieved by using FGP-1 and summarise our work reported in this chapter. According to our empirical results, we conclude that FGP is a useful tool for improving financial forecasting with respect to prediction accuracy, though caution should be exercised.

4.2 Combining Ordinal Forecasts

4.2.1 Introduction

Ordinal data could be useful in financial forecasting, as Fan et al. (1996) quite rightly pointed out. For example, forecast by experts may predict that a market is “bullish”, “bearish” or “sluggish”. A company’s books may show “deficit” or “surplus”. A share’s price today may ”, “fallen” or “remained unchanged” from yesterday’s. The question is how to use such data.

Let Y be a series, gathered at regular intervals of time (such as daily stock market closing data or weekly closing price). Let Y_t denote the value of Y at time t. Forecasting at time t with a horizon h means predicting the value of Y_{t+h} based on some information set I_t of other explanatory variables available at time t. The conditional mean

$$F_{t,h} = E[Y_{t+h} \mid I_t]$$

represents the best forecast of the most likely Y_{t+h} value (Granger 1992). In terms of properties of value Y, forecast could be classified into *point forecast*, where Y_t is a real value, or *ordinal forecast*, where Y_t is an interval estimate. In terms of the property of I_t, forecast could be classified into *time-series forecast*, where I_t consists of nothing but Y_{t-i} where $i \geq 0$, or *combining forecast*, where I_t only includes a finite direct forecast results of individual forecasts $\{E_{1, t}$, $E_{2,}$

$_{t}$,…, $E_{N,\,t}$}, where N is the number of sources; $E_{i,\,t}$ denotes the prediction of source e$_i$ at time t.

In the past two decades, point forecast on time series has played an important role in financial forecasting research. This includes the popular linear model ARIMA (Box & Jenkins 1970) and recently heavily studied non-linear models ARCH (Engle 1982) and GARCH (Bollerslev 1986). Nevertheless, there has been growing interest in combining forecasts; for example, see (Wall & Correia 1989; Lobo 1991; MacDonald & Marsh 1994) for combining point forecasts and (Fan et al. 1996; Cesa et al. 1997) for combining ordinal forecasts. The consensus of the literature is that mean forecast (combining point forecast) may outperform most time series models on average and combined ordinal forecast may outperform individual forecasts on average. The methodologies adopted in these researches are mainly statistical methods and operation research methods. AI techniques are seldom used. Although artificial neural networks have already been used to approach forecast combining (Donaldson & Kamstra 1996; Harrald & Kamstra 1997). The full potential of genetic algorithms (Holland 1975; Goldberg 1989; Davis 1991) has yet to be realized.

We follow the study of Fan and his colleagues and focus on combining ordinal forecasts. We demonstrate the potential of FGP-1 in combining and improving base predictions in two different data sets:

(i) a small data set involving the Hong Kong Heng Seng Index as reported by Fan and his colleagues (Fan et al. 1996); and

(ii) a larger data set involving S&P 500 index from 2 April 1963 to 25 January 1974 (2,700 trading days).

4.2.2 The specific representation

The problem of combining ordinal forecast can be formally described as follows. Let {P$_1$, P$_2$, …, P$_m$} be a set of discrete forecasting categories, where m is the number of forecasting categories.

At any time t, given N predictions $\{E_{1,t}, E_{2,t},\ldots, E_{N,t}\}$ where prediction $E_{i,t}$ can only take on one discrete value from the set $\{P_1, P_2, \ldots, P_m\}$, the goal in combining ordinal forecast is to produce a forecast of the same type:

$$E_{f,t} = f(E_{1,t}, E_{2,t}\ldots, E_{N,t}) \qquad \text{where } E_{f,t} \in \{ P_1, P_2, \ldots, P_m \}. \qquad (4.1)$$

For this exercise to be of value over a range of discrete time in the future T, the average accuracy of $E_{f,t}$ for $t \in T$ should be better than that of the best of $\{E_{1,t}, E_{2,t},\ldots, E_{N,t}\}$.

S:= <GDT>;
< GDT > := "If-then-else" <Condition> < GDT > < GDT > | <Decision>;
<Condition> := <Condition> "And" <Condition> | <Condition> "Or" <Condition> | "Not" <Condition> | <Selector>;

<Decision>:= "P_j" ;
< Selector > := "$E_{i,t}$" <RelationOperation> "P_j "
<RelationOperation>:= ">" | "<" | "=";

Figure 4.1: The BNF grammar that FGP uses for combining ordinal forecast (where, $1 \le i \le N; 1 \le j, k \le m$).

To construct GDTs, we need to take the FGP grammar (cf., the BNF grammar in Figure 3.6, p78) with a little variant. In the case of the Hong Kong stock market, the set of possible decisions is {bullish, bearish, sluggish, uncertain}. In the case of the S&P 500 index data, the set of decisions is {buy, not-buy}. Note that the set of forecast categories $\{P_1, P_2, \ldots, P_m\}$ is treated as an ordered list when “<” and “>” are applied. The specific syntax used in FGP-1 for combining ordinal forecasts can be precisely described by using the grammar, as shown in Figure 4.1.

A simple example tree is illuminated in Figure 4.2.

If-then-else (($E_{1,\,t}$ = P_2); (P_2); (If-then-else ($E_{2,\,t}$ > P_1); (P_2); (P_1)))

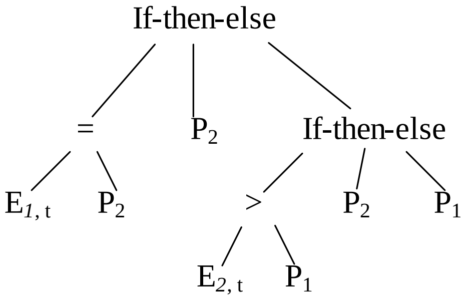


Figure 4.2: A simple rule and its corresponding tree structure.

The rule in Figure 4.2 means that if $E_{1,t}$ at time t ($E_{1,t}$) is P_2, then this rule predicts P_2 as well; else the prediction depends on the $E_{2,t}$. If $E_{2,t}$ is greater than P_1 (This means $E_{2,t}$ must be P_2 or P_3 if available) then this rule generates P_2 prediction, otherwise it predicts P_1.

4.2.3 Two Illustrative Examples

4.2.3.1 Application of FGP-1 to the Hong Kong Stock Market

FGP-1 was first applied to a particular prediction problem in the Hong Kong Stock Market. The data set given in the appendix of Fan et al. (1996) includes 103 data cases, each of which consists of nine expert predictions for the following week plus the actual market state. Predictions by each of the 9 experts fall into four categories. Fan et al. (1996) labelled the four categories as:

1. *bullish*, which is defined as the index rises by over 1.3% in the next week;
2. *bearish*, which means the index falls by over 1.3% in the next week;
3. *sluggish*, which means the index is neither bullish nor bearish; or
4. *uncertain*, which means the expert refuses to make a prediction.

The period under study was from 25 May 1991 to 16 October 1993. The Hong Kong stock market prediction can be formalized as a combining forecast problem as defined in the previous section 4.2.2:

At time t, given 9 predictions $\{E_{1,t}$,..., $E_{9,t}\}$ supplied by 9 experts $\{e_1,\ldots, e_9\}$ respectively, the prediction $E_{i,t}$ can only takes on one category value in the set {1, 2, 3, 4}, i.e. $E_{i,t} \in \{1, 2, 3, 4\}$. The goal here is to combine the predictions by the experts to generate more accurate predictions. In other words, we want to produce a function f such that:

$$E_{f,t} = f(E_{1,t}, \ldots, E_{9,t}) \qquad \text{where } E_{f,t} \in \{1, 2, 3\}. \qquad (4.2)$$

The hope is that the average accuracy of $E_{f,t}$ outperforms that of all of $\{E_{1,t}, \ldots, E_{9,t}\}$.

We partitioned the data given in Fan et al. (1996) into two mutually exclusive subsets: training data set (in-sample data set) and test data set (out-sample data set). We ran our FGP on the training data set. Each run generated one best-so-far prediction rule, which was then applied to the test data set in order to measure its performance.

Fan et al. (1996) used the "leave-one-out cross-validation strategy" to assess the forecasting accuracy. This means that in order to generate a forecasting for time t, all but the experts' predictions at time t were used to generate a combined prediction. Predictions generated in this way were evaluated. For simplicity without lost of generality, we used "3-fold cross-validation" to estimate the performance of FGP-1: we partitioned the data set into three mutually exclusive subsets (the folds):

D_1: 34 data cases from 25 May 1991 to 11 January 92;
D_2: 35 data cases from 18 January 1992 to 5 December 1992;
D_3: 34 data cases from 12 December 1992 to 16 October 1993

Each of these data sets was used as the test data set once, whilst the remaining two sets were employed as the training data set. The mean forecasting accuracy was the overall number of correct forecasts divided by number of cases in the whole data set (Kohavi 1995). For each of

D_1, D_2, D_3, we ran FGP-1 10 times, so a total of 30 runs were completed in our experiments.

Objective	Find GDTs which have the higher accurate prediction for the movement of Stock in next week.
Input terminals (forecasts of 9 experts)	$E_{1,t}$, $E_{2,t}$, $E_{3,t}$, $E_{4,t}$, $E_{5,t}$, $E_{6,t}$, $E_{7,t}$, $E_{8,t}$, $E_{9,t}$.
Prediction terminals	0, 1, 2. Where 0 means "Bullish"; 1 means "Bearish"; 2 means Sluggish".
Function set	If-then-else, And, Or, Not, >, <, =.

Data	D1: 34 data cases from 25 May 1991 to 11 January 92;
	D2: 35 data cases from 18 January 1992 to 5 December 1992;
	D3: 34 data cases from 12 December 1992 to 16 October 1993;
	Using 3-fold cross-validation to estimate FGP-1 forecasting performance.

Fitness function	RC (*Rate of Correctness*) = $\frac{\text{number of correct predctions}}{\text{total number of predictions}}$
Crossover rate	0.9
Mutation rate	0.01
Parameters	M (Population size) = 1000; G (Maximum generation) = 40.
Termination criterion	Maximum number of G of generation has been reached or FGP-1 programme has run for more than 2 hours.
Selection strategy	Tournament Selection, size = 4.
Max depth of individual program	17
Max depth of initial individual program	3
Run times (hours)	1-2
Hardware and operating system	Pentium PC 200MHz running Windows 95 with 64M RAM
Software	Borland C++ (version 4.5)

Table 4.1: Tableau for experiments on Hong Kong stock data.

Major running parameters in our experiments are depicted in Table 4.1. The fitness function is the *Rate of Correctness* (RC), the proportion of correct predictions out of all predictions. The FGP-1 forecast accuracy is presented in Table 4.2.

Runs	Number of correct forecasts or accuracy tested on D_1	Number of correct forecasts or accuracy tested on D_2	Number of correct forecasts or accuracy tested on D_3
1	20	14	15
2	19	16	16
3	21	16	14
4	22	16	15
5	21	17	17
6	21	16	16
7	23	15	14
8	22	17	14
9	20	16	17
10	18	15	16
Mean	20.7	15.8	15.4
Accuracy(RC)	0.6088	0.4514	0.4529
Mean Accuracy	0.5039		

Table 4.2: The 30 FGP-1 run performances and the mean forecast accuracy.

Methodologies	Expert 7	Multinomial logic (MNL)	Linear programming (LP)	Genetic programming (FGP-1)
Accuracy (RC)	0.4369	0.5016	0.4563	0.5039

Table 4.3: Accuracy of four "forecasts combining" methods in cross-validation.

Table 4.3 compares the FGP-1 result with the best expert result and the results of two methods discussed in (Fan et al. 1996). The mean accuracy of FGP method (50.39%) is slightly higher than the accuracy of the *Multinomial Logic Method* (50.16%) reported in (Fan et al. 1996). This in turn out-performs *Linear Programming Method* (45.63%) and the best individual expert (which is expert 7, at 43.69%). It is encouraging to see that MNL, LP and FGP-1 can all improve the accuracy of the best expert's forecast.

However, caution should be exercised when interpreting empirical results (Markowitz & Xu 1994; Hooker 1995), particularly in this example, which only involves relatively small data cases. Therefore, one should not generalize the results without further experimentation.

4.2.3.2 Application of FGP-1 to the S&P 500 Index

Encouraged by FGP-1's promising forecasting performance on the Heng Seng Index, we tested FGP-1 on the S&P-500 daily closing index. Available to us were data from 2 April 1963 to 25 January 1974 (2700 data cases). Our goal is to see whether FGP-1 could improve forecasting accuracy on textbook-type predictions.

Six technical rules (three different types, see Section 2.3) are treated as 6 experts {e_1, e_2,…, e6}. They are used to approach the following prediction problem at any given day, which is denoted by P_{63}^{4} (a general form of this type of prediction was introduced in Section 1.3.1).

P_{63}^{4} : whether the index will increase at least 4% within 63 trading days (3 months).

Each of the six rules will make either a "buy" or "not-buy" prediction every day {$E_{1,t}$, …, $E_{6,t}$}, where $E_{i,t} \in$ {0, 1; where "0" means "not-buy" and "1" means "buy"}. The six technical rules we used were as follows:

- Type A: moving average rules:

Given a price time series {P(t), t ≥ 0}, *simple moving average* is defined as:

SMV(*L*, t) = $\frac{1}{L}\sum_{i=0}^{L-1} P(t-i)$, where *L* is the length of moving average, *L*≥1. This rule is defined as “Buy if today’s price is greater than the average price of previous *L* periods”.

Rule 1 (**SMV_12**): "If today’s index price P(*t*) is greater than the SMV(12, *t*), then buy; else do not buy."

Rule 2 (**SMV_50**): "If today’s index price P(t) is greater than the SMV(50, t), then buy; else do not buy."

- Type B: trading range break-out rules:

Given a period of length *L*, this rule is defined as "Buy if today’s price P(*t*) is greater than the maximum of the previous L periods."

Rule 3 (**TRB_5**): "If today index price P(t) is greater than Max(P(t-1), P(t-2),…,P(t-5)), then buy; otherwise not buy."

Rule 4 (**TRB_50**): "If today index price P(t) is greater than Max(P(t-1), P(t-2),…,P(t-50)), then buy; otherwise not buy."

- Type C: filter rules:

This rule is defined as "Buy when the price rises *y* percent above its past local low."

In this case, the two filter rules are defined as follows:

Rule 5 (**Filter_5**): "If today index price P(t) rises 1% greater than Min(P(t-1), P(t-2),…, P(t-5)), then buy; otherwise not buy."

Rule 6 (**Filter_10**): "If today index price P(t) rises 1% greater than Min(P(t-1), P(t2),…, P(t-10)), then buy; otherwise not buy."

In this study, each of six technical rules is assumed to be one expert, whereas the “buy” -buy” signals generated by each technical rule are viewed as ordinal forecasts. Here, we are only concerned about whether FGP-1 is capable of making predictions with higher accuracy by combining ordinal forecasts generated from these technical rules. Therefore, the quality of the individual technical rule is not crucial to our study.

Objective	Find GDTs that have the higher accurate prediction for P_{63}^{4} .
Input terminals (signals of 6 Rules)	$E_{SMV\_12, t}$, $E_{SMV\_50, t}$, $E_{TRB\_5, t}$, $E_{TRB\_50, t}$, $E_{Filter\_5, t}$, $E_{Filter\_10, t}$.
Prediction terminals	0, 1. Where 1—"Buy" Class, 0— "not-buy" Class.
Data	Total data cases: 2700 (01/04/1963 -- 25/01/1974) The training data cases: 1800 (02/04/1963 -- 02/07/1970) The testing data cases: 900 (06/07/1970 -- 25/01/1974)
Fitness function	Fitness Function = w_1*RC - w_2*RMC - w_3*RF Where in this case, w1 = 1, w2 = 0.2, w3 = 0.3.
Parameters	M=1200 (Population size); G = 40 (Max generation).

Table 4.4: Tableau for experiments on S&P 500.

The FGP-1 algorithm is the same as that in the first example. In addition to the *rate of correctness* (RC), we added two factors to the fitness function: the *rate of missing chance* (RMC) and the *rate of failure* (RF). RMC and RF are defined as follows (identical definitions shall be given in Chapter 5):

$$\text{RMC} = \frac{\#\ of\ erroneous\ not\text{-}buy\ signals}{\#\ of\ actual\ buy\ opportunities} \tag{4.3}$$

$$\text{RF} = \frac{\#\ of\ erroneous\ buy\ signals}{\#\ of\ buy\ signals} \tag{4.4}$$

Weights were given to RC, RMC and RF (see Table 4.4). By adjusting these weights, we can reflect the preference of investors. For example, a conservative investor may want to avoid failure and consequently put more weight on RF.

Individual technical rule performance		FGP-1 rule performances	
Rules	Prediction Accuracy (RC)	FGP-1 Rules	Prediction Accuracy (RC)
SMV_12	0.4956	GDT 1	0.5400
SMV_50	0.5189	GDT 2	0.5389
TRB_5	0.4733	GDT 3	0.5400
TRB_50	0.4756	GDT 4	0.5522
Filter_5	0.4944	GDT 5	0.5444
Filter_10	0.4889	GDT 6	0.5367
		GDT 7	0.5389
		GDT 8	0.5356
		GDT 9	0.5433
		GDT 10	0.5300
Highest	0.5189	Highest	0.5444
Lowest	0.4733	Lowest	0.5300
Mean	0.4911	Mean	0.5420
STD	0.0165	STD	0.0056

Table 4.5: Performance comparisons between individual technical rules and 10 GDTs (STD means the standard deviation).

We ran FGP-1 10 times. Major parameters that differ from those in the pervious experiments are listed in Table 4.4. For each run, a GDT evolved on the training data was applied to the test data. The results of GDTs and six individual rules on the test data were recorded in Table 4.5. Among the six technical rules, the SMV_50 rule was the best individual rule for this set of data. It achieved an accuracy of 51.89%. In contrast, GDTs achieved an average accuracy of 54.20%, hence out-performed the rule SMV_50 by 2.31%. Even the poorest GDT (GDT 10) achieved an accuracy of 53.00%, which was still better than that of the SMV_50 rule. So although we only generated 10 decision trees, the results were conclusive: FGP-1 was able to produce better forecasting consistently by combining individual base forecasts.

However, it is worth noting that caution should be exercised for choosing running parameters of FGP-1, in particular, the fitness function adopted. Here, the fitness function we used was a weighted linear function with three criteria involved, rather than RC adopted in the preceding application. The reason is that the results by using RC as the fitness function are not good as results presented here. In our experiments, we also found that some improper weights did lead to bad results.

4.3 Combining and Adapting Technical Analysis Rules

4.3.1 Introduction

As has been discussed in Chapter 2 (see, Section 2.3.2 and Section 2.3.3), technical analysis has been enjoying a renaissance both in practitioners and in academics. Technical analysis rules have been argued to have merit to predictability of movements of future market prices.

There are two general approaches in technical analysis: one involves qualitative techniques and the other quantitative techniques (Goldberg & Schulmeister 1988). The qualitative techniques rely on the interpretation of the form of geometric patterns in the series, such as double bottoms, head-and-shoulders, and support and resistance levels; whilst the quantitative techniques try to create indicators for market timing such as moving average (MV), relative strength indicators (RSI), etc. Notably, both techniques can be characterised by appropriate sequences of local minima and/or maxima (Neftci 1991).

Quantitative technical rules are often used to generate “buy” or “sell” signals based on each rule interpretation. One may want to use technical rules to answer questions such as "is today a good time to buy if I want to achieve a return of 4% or more within the next 63 trading days?" and "is today the right time to sell if I want to avoid a loss of 5% or more within the next 10 days?" However, the way that technical rules are commonly used may not be adequate to answer these questions. How to use them and adapt them to these specific prediction problems to lead a better solution is a non-trivial task.

In fact, any answer to the above questions could be one of the solutions to a class of problems P_n^r. In this study, we ask FGP-1 to address such problems. Our purpose is to examine whether FGP-1 is a useful tool if we are faced with some available technical analysis rules, together with some knowledge of using them in their normal ways. In particular, we shall examine whether FGP-1 can help to improve predictive accuracy over non-adaptive individual technical rules available (non-adaptive technical analysis rules will be explained later on), with respect to prediction problems P_n^r .

We choose two examples of the prediction problem P_n^r using the same Dow Jones Industrial Average (DJIA) index data. One is a short-term prediction $P_{21}^{2.2}$. The other is a middle-term prediction P_{64}^{4} . With the two instances of prediction problems, we would like to see whether FGP-1 could achieve consistent results.

Furthermore, in order to evaluate FGP-1, we compare FGP-1 against the random walk model and C4.5, a well-known machine learning classifier system. We shall see whether GDTs generated by FGP-1 have any superiority over random decisions and rulesets generated by C4.5 in terms of prediction accuracy.

4.3.2 The Specific Representation

Involved in this study are also six similar technical analysis rules, which were used in the second example in Section 4.2 (see p96). However, the way of using these rules are different. The difference lies in what kind of data information is used as input to FGP-1. In the preceding application (see Section 4.2.3.2), the ordinal forecasts: “buy” or “not-buy” were used that are generated based on the interpretation of each technical rule. In comparison, here, indicators are to be used to feed FGP-1. Six indicators are derived from corresponding technical rules. They are defined as follows:

(1) $I_{MV\_12}(t) = P_t - MV_{(t, 12)}$, which is related to **SMV_12.**

(2) $I_{MV\_50}(t) = P_t - MV_{(t, 50)}$, which is related to **SMV_50.**

(3) $I_{TRB\_5}(t) = P_t - P_{max\,(1, 5)}$, which is related to **TRB_5**

(4) $I_{TRB\_50}(t) = P_t - P_{max\,(1, 50)}$, which is related to **TRB_50.**

(5) I_{Filter_5} (t) = $P_t - P_{min\,(1,\,5)}$, which is related to **Filter_5.**

(6) I_{Filter_63} (t) = $P_t - P_{min\,(1,\,63)}$, which is related to **Filter_63.**

Where P_t is the current price; $P_{min\,(1,\,L)}$ = Min (P_{t-1}, P_{t-2}, … , P_{t-L}); $P_{max\,(1,\,L)}$ = Max (P_{t-1}, P_{t-2}, … , P_{t-L}); $MV_{(t,\,L)} = \frac{1}{L}\sum_{i=0}^{L-1} P_{t-i}$.

As discussed in Chapter 2 (see Section 2.4), "buy" signals generated from trading rules imply potential price rising in the future. Therefore, a series of "buy" signals are presumably treaded as predictive solutions to problem P_n^r . Each technical rule has its own interpretation for generating these signals. Interpretation is based on the past prices, as well as an important item, i.e. a threshold.

The threshold is necessary to trigger signals for each rule. For example, a value of "0" is normally used for moving average rules and trade range break-out rules (e.g., **SMV_12, SMV_50, TRB_5,** and **TRB_50** in Section 4.2.3.2), whilst a value of "1%" is normally used for Filter rules (e.g., **Filter_5** and **Filter_63**). If six technical rules trigger signals with the corresponding thresholds mentioned above, the way of using them is said to be in their normal usages, which are usually employed by practitioners for market timing and usually studied in finance literature (Brock et al. 1992). Nevertheless, in our cases, these rules in their normal usages are considered to be non-adaptive rules in the sense that they may not be suited to solve a specific prediction task P_n^r with a fixed n and r. We argue that a technical rule with a suitable value of threshold might be more helpful.

However, choices of these thresholds are numerous. It may be difficult to adjust thresholds in order to adapt them for a specific task P_n^r at hand. We expect FGP-1 might help in this regard. By using the hill-climbing method, FGP-1 helps to find proper thresholds for these individual technical rules. Moreover, by using the FGP grammar, FGP-1 can generate GDTs which can combine these individual technical rules together. The hope is that the generated GDTs might make predictions with higher accuracy that could not be achieved by any of the individual technical rules in their normal usages.

In this study, we ask FGP-1 to identify investment opportunities where a return of *r*% or more can be achieved within the next n period times. Recommendation in this application is either positive (which suggests that a return of r% or more can be achieved within the next n period times) or negative (otherwise).

The specific grammar for constructing GDTs in this application is described in Figure 4.3.

```
S ::= < GDT >;
<GDT> ::=  "If-then-else"  <Condition>  <GDT>  <GDT> | < Recommendation >;

<Condition> ::= <Condition> "And" <Condition>  |  <Condition> "Or" <Condition>  |"Not" <Condition>  |
                <Selector>;
<Recommendation> ::= "Positive" | "Negative";
<Selector> ::= <Indicator> <RelationOperation> <Threshold>;

<RelationOperation> ::=   ">"   |   "<"   |  "=" ;
<Indicator> ::= "I_MV_12(t)" | "I_MV_50(t)" | "I_Filter_5(t)" | "I_Filter_63(t)" | "I_TRB_5 (t)" | "I_TRB_50 (t)";
<Threshold>::= “Real Number”;
```

Figure 4.3: The BNF grammar that FGP-1 uses for constructing GDTs.

Figure 4.4 shows an example of a simple GDT built by using the above grammar. A useful GDT in the real world might be a lot more sophisticated than this.

```
(IF  (MV_50(t)  <  -18.45) THEN Positive
ELSE (IF  ((TRB_5 (t)  >  -19.48) AND (Filter_63 (t) < 36.24))
          THEN Negative
          ELSE Positive))
```

Figure 4.4: A (simplistic) GDT for decision making.

This GDT is assumed to makes predictions daily. It suggests that if today’s price is 18.45 or more below the average price of the last 50 trading days, then a return of *r*% within n days might be achievable if one invest today (we call today a positive position). Otherwise, decision depends on the values of TRB_5(t) and Filter_63(t). If today's price is no more than 19.48 above the maximum price of the previous 5 trading days or today's price is more than 36.24 above the minimum price in the last 63 trading days, then today is also an alternative good opportunity to make a "buy" decision.

This rule, in fact, makes a decision based on combination of the decisions from three individual technical rules: i.e. a moving average rule with window size of 50, a trade range break-out rule with a window size of 5, and a filter rule with a window size of 63. Notably, the threshold of each rule takes a value, which is not identical to the value adopted in their normal usages. These values are called adaptive thresholds in the sense that they are discovered based on historical data in favour of solving the prediction task at hand.

The search space for GDTs is enormous. Apart from looking for proper elements (these elements include indicators, relational operators as well as thresholds) in order to constitute positive selectors, meanwhile one has to search for positive combinations of those selectors. The hope is that FGP-1 can effectively explore this search space.

4.3.3 Two Illustrative Examples

4.3.3.1 Experimental Data

The data we chose are the closing prices of the Dow Jones Industrial Average (DJIA) Index from 7 April 1969 to 5 May 1980, which includes 2,800 data cases. We took the index data from 7 April 1969 to 11 October 1976 (1,900 cases) as the training data, and took the index data from 12 October 1976 to 5 May 1980 (900 data cases) as the test data. The whole data series can be visualised in Figure 4.5.

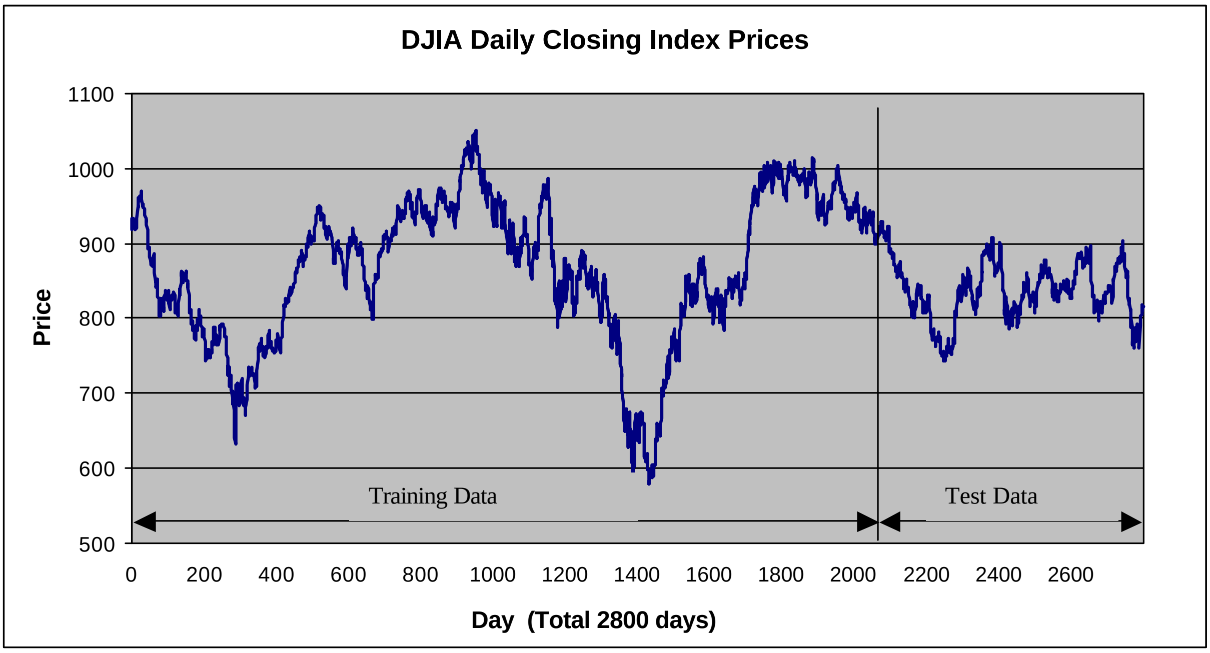


Figure 4.5: Experimental DJIA index Data.

The whole training data and test data contain roughly 50% of positive positions. More precisely, we define (***f***$_{tr}$*,* ***f***$_{te}$) to denote percentages of actual number of positive positions of total number of positions for a training data set and a test data set respectively.

$$f_{\text{tr}} = \frac{\text{\# of actaul postitive postitions in training data}}{\text{\# of training data}} X\ 100\% \qquad (4.5)$$

$$f_{te} = \frac{\text{\# of actaul postitive postitions in test data}}{\text{\# of test data}} X\ 100\% \qquad (4.6)$$

Although both predictions, P_{64}^{4} and $P_{21}^{2.2}$ are investigated based on the same data, due to different n and r chosen, a pair of values, (***f***$_{tr}$*,* ***f***$_{te}$) is different. For P_{64}^{4}, (***f***$_{tr}$*,* ***f***$_{te}$) is (52.84%, 49.22%); for $P_{21}^{2.2}$, (***f***$_{tr}$*,* ***f***$_{te}$) is (52.47%, 47.11%). By asking FGP-1 to attack both P_{64}^{4} and $P_{21}^{2.2}$, we may examine the robustness of the tool with respect to the length of prediction period.

4.3.3.2 Performance Criteria

In this study, the aim of using FGP-1 is to improve prediction accuracy. RC remains a principle performance criterion. As the application here lies in finance, performance criterion related to investment return would be interesting and desirable for the purpose of reference. Therefore, we develop an investment performance criterion, i.e. *average annualised rate of return* (AARR) based on following hypothetical trading behaviour:

> ***Hypothetical Trading Behaviour***: *We assume that when a positive position is predicted by a GDT, one unit of money was invested in a portfolio reflecting the current closing price. If the closing price does rise by r% or more at day t within the next n trading days, then we sell the portfolio at the closing price of day t. If not, we sell the portfolio on the nth day, regardless of the price.*

Given a positive position predicted, for example, the i^{th} positive position, for simplicity, we ignore transaction cost, and annualise its return by the following formula:

$$ARRi = \frac{253}{t} * \frac{P_t - P_0}{P_0} \tag{4.7}$$

Where P_0 is the buy price, P_t is the sell price, t is the number of days in markets, 253 is the number of total trading days in one calendar year.

Given a GDT that generates N_+ number of positive positions over the period examined, its average ARR is:

$$\text{AARR} = \frac{1}{N_+} \sum_{i=1}^{N_+} ARR_i. \tag{4.8}$$

It should be emphasised here that RC should be the main criterion for evaluating the performance of rules generated because it is what FGP-1 is asked to maximize. AARR should only be used for reference.

4.3.3.3 Experimental Results on $P_{21}^{2.2}$

In this section, we shall report our experimental results of FGP-1 on $P_{21}^{2.2}$. First, we shall examine whether FGP-1 is useful for improving forecasting. We present the mean performances of GDTs in comparison with those of the non-adaptive six individual technical rules. Then we evaluate those GDTs by comparison with random decisions and the rulesets generated by C4.5.

Objective	Find GDTs which have the higher accurate prediction for the $P_{21}^{2.2}$
Input terminals (forecasts of 9 experts)	MV_12 (t), MV_50 (t), TRB_5 (t), TRB_50 (t), Filter_5 (t), Filter_63(t), and real number as thresholds.
Prediction terminals	{0, 1}, with 1 representing "Positive"; 0 representing "Negative"
Function set	If-then-else, And, Or, Not, >, <, =.
Data	Total data cases: 2800 (07/04/1969 to 05/05/1980) The training data cases: 1900 (07/04/1969 to 11/10/1976) The test data cases: 900 (12/10/1976 to 05/05/1980)
Fitness function	RC (*Rate of Correctness*) = $\frac{\text{number of correct predctions}}{\text{total number of predictions}}$
Crossover rate	0.9
Mutation rate	0.01
Parameters	M (Population size) =1200; G (Maximum generation) = 30.
Termination criterion	Maximum number of G of generation has been reached or FGP-1 programme has run for more than 2 hours.
Selection strategy	Tournament Selection, Size = 4
Max depth of individual program	17
Max depth of initial individual program	3
Run times (hours)	1-2
Hardware and operating system	Pentium PC 200MHz running Windows 95 with 64M RAM
Software	Borland C++ (version 4.5)

Table 4.6: Tableau for the experiments on $P_{21}^{2.2}$.

In our experiments, we ran FGP-1 10 times. The termination condition was set to 2 hours on a Pentium PC (200 MHz) or 30 generation, whichever reached first. The main parameters of the experiments are displayed in Table 4.6. For each run, a GDT generated, based on the training data, was applied to the test data. The results of the 10 GDTs on the test data are recorded in Table 4.7. Six technical rules were also applied to the test data based on their interpretations in their normal usages respectively. Results of each individual trading rules are listed in Table 4.7 as well.

Experimental results are promising. The average RC of 10 GDTs is 54.78% in contrast to 49.65%, the average RC of six technical rules. Any one of 10 GDTs outperforms any one of six individual technical rules in terms of RC. Notably, even the poorest GDT (GDT-6), which achieves a RC of 54.00%, is better than the best individual technical trading rule TRB_50, which achieves a RC of 51.11%. As a result, any GDT generated achieves better performance than any trading rule with respect to AARR. Better performances of GDTs are partly due to their ability to recognise more actual positive positions. Among a total of 424 actual positive positions over the test period, GDTs correctly identify a mean of 52.59% (223/424×100%) of them, in comparison with a mean of 39.74% (168.5/424×100%), by six technical rules in their normal usages. Based on these empirical results, we argue that FGP-1 is a useful tool, which is capable of achieving better performance as opposed to each of the six individual technical rules in their normal usages.

In order to assess the quality of GDTs generated by FGP-1, we compare its results with those of random decisions and the rulesets generated by C4.5, both of which are reported in

Technical Rules	Performance on Test Data						FGP-1 Rules	Performance on Test Data					
	RC	AARR	TP	FP	TN	FN		RC	AARR	TP	FP	TN	FN
SMV_12	0.4978	0.3137	213	241	235	211	GDT-1	0.5544	0.4501	209	186	290	215
SMV_50	0.5089	0.3247	177	195	281	247	GDT-2	0.5467	0.4356	230	214	262	194
TRB_5	0.4978	0.2652	99	127	349	325	GDT-3	0.5567	0.4775	189	164	312	235
TRB_50	0.5111	0.0189	22	38	438	402	GDT-4	0.5444	0.4513	260	246	230	164
Filter_5	0.4967	0.3244	176	205	271	248	GDT-5	0.5444	0.4329	237	223	253	187
Filter_63	0.4667	0.3350	324	380	96	100	GDT-6	0.5400	0.4740	197	187	289	227
							GDT-7	0.5533	0.4767	210	188	288	214
							GDT-8	0.5478	0.4735	268	251	225	156
							GDT-9	0.5500	0.4653	230	211	265	194
							GDT-10	0.5400	0.4699	200	190	286	224
Highest	0.5111	0.3350	324.0	380.0	438.0	402.0	Highest	0.5567	0.4775	268.0	251.0	312.0	235.0
Lowest	0.4667	0.0189	22.0	38.0	96.0	100.0	Lowest	0.5400	0.4329	189.0	164.0	225.0	156.0
Mean	**0.4965**	**0.2637**	**168.5**	**197.7**	**278.3**	**255.5**	**Mean**	**0.5478**	**0.4607**	**223.0**	**206.0**	**270.0**	**201.0**
STD	0.0159	0.1224	102.5	114.6	114.6	102.5	STD	0.0058	0.0170	26.7	28.1	28.1	26.7

Table 4.7: Technical rule performances and FGP-1 rules performances on test data (900 daily cases: from 12/10/1976 to 05/05/1980) for $P_{21}^{2.2}$,

where TP (*True Positive*): the number of positive positions correctly predicted;
FP (*False Positive*): the number of negative positions incorrectly predicted as positive;
TN (*True Negative*): the number of negative positions correctly predicted;
FN (*False Negative*): the number of positive positions incorrectly predicted as negative.

Table 4.8.

Comparison with random decisions is motivated with the weak form of EMH, which was discussed in Chapter 2. According to weak form of EMH, stock prices follow a random walk behaviour and therefore no trading rules could out-perform random decisions. However, our empirical results contradict the random theory. For GDTs, the mean RC and AARR are 54.78% and 46.07% respectively. They are much higher than the mean RC (49.53%) and AARR (36.84%) achieved by the 10 random decisions. In fact, even the poorest results of 10 GDTs (54.00% and 43.29% for RC and AARR respectively) are better than the best results of the 10 random runs in terms of RC and AARR (53.67%, 42.35% respectively). Results here are consistent with our results achieved in the past over S&P 500 data (Tsang et al. 1998), which shows that FGP–1 is capable of out-performing random decisions in terms of RC and AARR.

Random	Performance on Test Data						Rulesets	C4.5 Performance on Test data					
Decision	RC	AARR	TP	FP	TN	FN	-c	RC	AARR	TP	FP	TN	FN
Random-1	0.5044	0.3548	224	246	230	200	100	0.5511	0.4608	122	102	374	302
Random-2	0.4822	0.3575	206	248	228	218	75	0.5467	0.4492	126	110	366	298
Random-3	0.5000	0.3551	213	239	237	211	50	0.5467	0.4422	134	118	358	290
Random-4	0.5089	0.4051	220	238	238	204	25	0.5489	0.4746	125	107	369	299
Random-5	0.4644	0.3336	191	249	227	233	10	0.5289	0.4093	159	159	317	265
Random-6	0.5367	0.3968	232	225	251	192	5	0.5211	0.4022	154	161	315	270
Random-7	0.4978	0.3794	222	250	226	202	1	0.4944	0.3736	246	277	199	178
Random-8	0.4667	0.3254	205	261	215	219							
Random-9	0.4867	0.3524	211	249	227	213							
Random-10	0.5056	0.4235	220	241	235	204							
Highest	0.5367	0.4235	232.0	261.0	251.0	233.0	Highest	0.5511	0.4746	246.0	277.0	374.0	302.0
Lowest	0.4644	0.3254	191.0	225.0	215.0	192.0	Lowest	0.4944	0.3736	122.0	102.0	199.0	178.0
Mean	**0.4953**	**0.3684**	**214.4**	**244.6**	**231.4**	**209.6**	**Mean**	**0.5340**	**0.4303**	**152.3**	**147.7**	**328.3**	**271.7**
STD	0.0214	0.0318	11.7	9.6	9.6	11.7	STD	0.0208	0.0361	43.8	62.0	62.0	43.8

Table 4.8: Performances of random decisions and rulesets generated by C4.5 for $P_{21}^{2.2}$.

C4.5 is one of the most commonly used decision tree learning classifier systems, which was developed by Quinlan (1986, 1993). Both FGP-1 and C4.5 take the same type of input and generate decision trees, which C4.5 converts to rulesets that is more easily understood by people. We fed C4.5 with the same six technical rule indicators that we adopted for FGP-1. We ran C4.5 system on the same training data and applied the generated rulesets to the same test data. There follows an example of a single rule generated by C4.5:

If (PMV_50 > -33.075) And (PMV_50 <= -28.0292) And
(TRB_50 <= -69.15 And (Filter_5 > -0.26)
Then Positive Position

A parameter that significantly affects the performance of the rulesets generated by C4.5 is the "certainty factor" (run with -c CF), which ranges from 0 to 100. The certainty factor is used to control pruning, details of which will not be elaborated here. The value -c 25 represents default pruning in C4.5. Small values usually lead to small rulesets, whereas large values imply less pruning and therefore large rulesets. In Table 4.8 under the row of "Rulesets" are -c options with seven different CF values and their performances of the corresponding rulesets generated. Mean results for RC and AARR are 53.40% and 43.03%, each of which is lower than the corresponding mean result of GDTs, but higher than the mean results of random runs respectively.

Groups	FGP-1 Vs C4.5		C4.5 Vs Random Runs	
Criteria	For RC	For AARR	For RC	For AARR
t values	2.013*	2.341*	3.700*	3.738*
p values	0.0312	0.0167	0.0011	0.0009
df (degrees of freedom)	15	15	15	15

Table 4.9: t-statistics for comparing the mean performances of two groups for $P_{21}^{2.2}$ (FGP-1 versus C4.5 and C4.5 versus Random Runs). * indicates the statistically significant difference between the means results of the two considered methods with ($\alpha = 0.05$).

To determine whether the difference between the means of two methods is statistically significant, we use one tailed unpaired *t*-test with $\alpha = 0.05$, and with df = (15 = 10+7–2). The critical value obtained from the *t*-test table is 1.753. If an observed *t*-value exceeds this critical value, we can conclude that there is a significant difference between the means of the two considered methods. In Table 3 are the test results for both comparisons of FGP-1 versus C4.5 and C4.5 versus random runs. *t*-values for comparing FGP-1 against C4.5 are 2.013 and 2.341

(both of which are greater than 1.753) for RC and AARR respectively. These suggest that the mean results of FGP-1 are at least better than those of C4.5 in terms of both RC and AARR at the conventional statistical significant level (α = 0.05). Meanwhile, t-values for comparing C4.5 against the random runs demonstrate that C4.5 also outperforms random runs at the conventional statistical significant level (α = 0.05).

It is encouraging that both GDTs and the rulesets seem to grasp plausible hidden patterns in financial data as to achieve better performances that cannot explained by random decisions. More important is that FGP-1 outperforms C4.5 statistically significantly in this case. Poor performance of C4.5 may contribute to its overfitting problem. On the training data, the results of rulesets are much better than results of GDTs in terms of RC (both results are not shown here). This means rulesets are too overfitting on the training data to be as good as GDTs on the test data.

4.3.3.4 Experimental Results on P_{63}^{4}

In order to study the robustness of FGP-1 for improving prediction accuracy, we would like to investigate whether FGP-1 can achieve consistent results if it is given a similar prediction problem P_n^r but with different n and r. More specifically, we choose P_{63}^{4} for FGP-1 to attack. Similarly, we shall report both results of GDTs and the six individual technical rules in order to examine whether FGP-1 still achieves better performances than any one of six technical rules in their normal usages. We shall show results of C4.5 and random decisions as well in order to observe whether the situation observed in the preceding task retains for the similar but slightly varied prediction problem, P_{63}^{4}.

Technial	Performance on Training data						FGP-1	Performance on Test Data					
Rules	RC	AARR	TP	FP	TN	FN	Rules	RC	AARR	TP	FP	TN	FN
SMV_12	0.5144	0.2068	230	224	233	213	GDT-1	0.6022	0.2756	228	143	314	215
SMV_50	0.4256	0.1694	149	223	234	294	GDT-2	0.6200	0.3171	238	137	320	205
TRB_5	0.4944	0.1818	107	119	338	336	GDT-3	0.6067	0.2880	272	183	274	171
TRB_50	0.4744	-0.0534	15	45	412	428	GDT-4	0.5800	0.3655	136	71	386	307
Filter_5	0.5267	0.2303	199	182	275	244	GDT-5	0.6022	0.2823	276	191	266	167
Filter_63	0.5056	0.2277	351	353	104	92	GDT-6	0.5511	0.2976	162	123	334	281
							GDT-7	0.6133	0.3052	251	156	301	192
							GDT-8	0.5789	0.2716	243	179	278	200
							GDT-9	0.6067	0.2875	269	180	277	174
							GDT-10	0.6244	0.2593	223	118	339	220
							GDT-11	0.5678	0.2588	266	212	245	177
							GDT-12	0.5611	0.2685	193	145	312	250
							GDT-13	0.6056	0.2966	249	161	296	194
							GDT-14	0.5478	0.2543	196	160	297	247
							GDT-15	0.5600	0.2582	264	217	240	179
							GDT-16	0.6056	0.2918	259	171	286	184
							GDT-17	0.5367	0.2357	234	208	249	209
							GDT-18	0.5300	0.2382	234	214	243	209
							GDT-19	0.5367	0.2418	216	190	267	227
							GDT-20	0.5578	0.2634	200	155	302	243
Highest	0.5267	0.2303	351.0	353.0	412.0	428.0	Highest	0.6244	0.3655	276.0	217.0	386.0	307.0
Lowest	0.4256	-0.0534	15.0	45.0	104.0	92.0	Lowest	0.5300	0.2357	136.0	71.0	240.0	167.0
Mean	**0.4902**	**0.1604**	**175.2**	**191.0**	**266.0**	**267.8**	**Mean**	**0.5797**	**0.2779**	**230.5**	**165.7**	**291.3**	**212.6**
STD	0.0363	0.1075	114.4	104.8	104.8	114.4	STD	0.0307	0.0306	37.6	36.9	36.9	37.6

Table 4.10: Technical rule performances and FGP-1 rules performances on test data from 12/10/1976 to 05/05/1980) for P_{63}^{4}.

In this set of experiments, parameters for running FGP-1 are similar to those adopted in the preceding experiments on $P_{21}^{2.2}$. However, we ran FGP-1 20 times rather than 10 times. 20 GDTs were generated. Table 4.10 displays performances of both 20 GDTs and the six individual technical rules in their normal usages over the test data.

For P_{63}^{4}, FGP-1 illustrates its consistent capability of achieving better performances in terms of RC and AARR. Mean RC and AARR of GDTs are 57.97% and 27.79%, which are better than RC and AARR of any individual rule respectively. Even the poorest GDT (GDT_18, which has a RC of 53.00% and an AARR of 23.82%) is superior to the best rule (Filter_5, which has a RC of 52.67% and an AARR of 23.03%). These consistent results further demonstrate that FGP-1 is a useful tool. It can generate GDTs that are more accurate by combining individual technical rules, as well as adapting thresholds to the specific problem at hand.

Random	Performance on Test Data						Rulesets	C4.5 Performance on Test Data					
Decision	RC	AARR	TP	FP	TN	FN	-C	RC	AARR	TP	FP	TN	FN
Random-1	0.4911	0.2487	202	217	240	241	100	0.5044	0.2058	118	121	336	325
Random-2	0.5133	0.2262	227	222	235	216	75	0.5011	0.1794	124	130	327	319
Random-3	0.5022	0.2205	225	230	227	218	50	0.5078	0.2047	115	115	342	328
Random-4	0.5044	0.2282	218	221	236	225	25	0.5211	0.2205	128	116	341	315
Random-5	0.4800	0.2415	208	233	224	235	10	0.5033	0.2298	223	227	230	220
Random-6	0.5089	0.2509	234	233	224	209	5	0.4700	0.1565	118	152	305	325
Random-7	0.4811	0.2446	220	244	213	223	1	0.5178	0.2586	205	196	261	238
Random-8	0.4933	0.2447	220	233	224	223							
Random-9	0.4700	0.1976	214	248	209	229							
Random-10	0.4944	0.2348	200	212	245	243							
Random-11	0.5233	0.2206	295	281	176	148							
Random-12	0.4811	0.1956	205	229	228	238							
Random-13	0.4689	0.2044	223	258	199	220							
Random-14	0.5011	0.2614	215	221	236	228							
Random-15	0.5022	0.2295	219	224	233	224							
Random-16	0.4833	0.2161	210	232	225	233							
Random-17	0.5211	0.2325	227	215	242	216							
Random-18	0.5156	0.2347	233	226	231	210							
Random-19	0.5111	0.2054	230	227	230	213							
Random-20	0.5144	0.2481	236	230	227	207							
Highest	0.5233	0.2614	295.0	281.0	245.0	243.0	Highest	0.5211	0.2586	223.0	227.0	342.0	328.0
Lowest	0.4689	0.1956	200.0	212.0	176.0	148.0	Lowest	0.4700	0.1565	115.0	115.0	230.0	220.0
Mean	**0.4981**	**0.2293**	**223.1**	**231.8**	**225.2**	**220.0**	**Mean**	**0.5037**	**0.2079**	**147.3**	**151.0**	**306.0**	**295.7**
STD	0.0165	0.0186	19.9	16.0	16.0	19.9	STD	0.0166	0.0333	46.1	44.1	44.1	46.1

Table 4.11: Performances of random decisions and rulesets generated by C4.5 for P_{63}^{4}.

Performances of both random decisions and the rulesets generated by C4.5 are reported in Table 4.11. The mean RC of 20 GDTs (i.e. 57.97%) is better than the mean RC of 7 rulesests (i.e. 50.37%), and the mean RC of 20 random decisions (49.81%). A similar fact exists, for AARR, though as a reference criterion. It worth noting that the poorest GDT (GDT_18, RC = 53.00%) is better than the best rulesets (the one generated under -c 25, RC = 52.11%) and the best random decision (Random_11, RC = 52.33%).

Results of the statistical one-tailed unpaired *t*-test for both groups (FGP-1 Vs C4.5 and C4.5 Vs Random Runs) are presented in Table 4.12 under criteria of RC and AARR respectively. Again, in terms of both RC and AARR, results of GDTs generated by FGP-1 are statistically significantly better than those of the rulesets generated by C4.5 (the critical value for this *t*-test with ($\alpha = 0.05$) and df = (25 = 20+7–2) is 1.708). FGP-1 beats C4.5 in this case. However, in this

Groups	FGP-1 Vs C4.5		C4.5 Vs Random Runs	
Criteria	For RC	For AARR	For RC	For AARR
t values	6.2585*	5.0943*	0.7697	2.1147*
p values	0.000001	0.000015	0.2244	0.0223
df (degrees of freedom)	25	25	25	25

Table 4.12: t-statistics for comparing the mean performances of two groups for P_{63}^{4} (FGP-1 versus C4.5 and C4.5 versus Random Runs). * indicates the statistically significant difference between the means results of the two considered methods with ($\alpha = 0.05$).

case, under RC, results of C4.5 are not statistically significantly better than results of random runs because *t* value for RC (i.e. 0.7696) is less than 1.708, though they are under AARR.

4.4 Summary and Conclusion

This chapter has presented the applications of FGP-1 to financial prediction problems. The aim of using FGP-1 is to improve prediction accuracy over base predictions given. More specifically, two instances of base predictions have been studied here. One is that base predictions consist of ordinal forecasts from experts considered. The other is that base predictions come from non-adaptive technical analysis rules considered to be in their normal usages.

In the first part of this chapter (Section 4.2), we have studied the effectiveness of FGP-1 for improving forecasting if we are faced with only ordinal forecasts. We presented FGP-1 system for combining discrete forecasts in order to produce more accurate forecasting. FGP-1 has been tested on two sets of data: it was used to combine weekly expert predictions on the Hong Kong Heng Seng Index. It was also used to combine the forecasts generated by the six trading rules. In both cases, results generated by FGP-1 are conclusive and consistent in the sense that the GDTs generated by FGP-1 outperform the best base prediction given. We conclude that FGP-1 is a useful tool. It is capable of generate more accurate predictions by combining only individual ordinal forecasts available. However, caution should be exercised for the choices of parameters in the fitness function. Improper settings of weights could possibly lead to bad results.

Ordinal forecasting is a common practice in numerous situations. Apart from instances studied in finance, examples of ordinal forecasts include the outcome of a football match (win, tie, lose) and the weather (sunny, cloudy, rainy), and future state of the economy (boom, recession). The issue of combining forecasts has many practical applications. While methodologies adopted to approach this issue are mainly statistical methods and operation research methods, AI forecasting techniques, more specifically, the genetic programming based method presented here provides an alternative avenue.

In the second part of this chapter (Section 4.3), we have investigated the effectiveness of FGP-1 for improving forecasting if only individual non-adaptive technical rules are given. First, we derive the six indicators from the six technical rules respectively. Second, we use the FGP grammar to form an essential element, called a selector. One selector has the form [*Indicator relation threshold*], where the relation belongs to the set {=, <, >} and the threshold is a real number. The threshold in each selector could possibly be adjusted during evolving. Moreover, FGP-1 looks for the interactive combination structures between those selectors. The way of selector combination is either conjunction or disjunction. By doing so, FGP-1 is capable of evolving GDTs that are capable of achieving better prediction accuracy.

We tested FGP-1 over two similar prediction problems: $P_{21}^{2.2}$ (whether a 2.2% price increase or more is achievable within 21 days) and P_{63}^{4} (whether a 4% price increase or more is achievable within 63 days). Experimental results show that GDTs generated by FGP-1 have higher predictive accuracy over any individual non-adaptive technical rules available. GDTs are compared against random decisions and the rulesets generated by C4.5 as well. Empirical comparative results demonstrate that FGP-1 beat random runs and C4.5 statistically in terms of RC and AARR. Results are consistent and conclusive for both $P_{21}^{2.2}$ and P_{63}^{4}.

Based on the experiments presented in this chapter, we conclude that FGP-1 is capable of improving prediction accuracy over individual forecasts, as well as non-adaptive individual technical rules with respect to the prediction tasks we address. The successful story here is partly attributed to FGP-1's search capability. FGP-1 helps us to search in the space of GDTs, which represent interactions between base predictions. Without FGP-1 or other comparable tools, it is difficult to search the space of GDTs. FGP-1 is a useful tool which may help the user to make better predictions over the best of the base predictions available. However, we do not wish to give the false impression that FGP-1 may succeed in every case if base predictions are given. Our position is: if the promising patterns of combining base predictions exist, FGP-1 stands a chance of finding them.

Chapter 5

Achieving a Low Rate of Failure Using FGP-2

5.1 Introduction

In the preceding chapter, we have shown that FGP-1 can be used to make financial predictions. FGP-1 aims to achieve the first research goal, i.e. to improve the accuracy of given predictions. We have demonstrated the effectiveness of FGP-1 through several examples. Given a finite of base predictions available, by combining them, FGP-1 can generate GDTs which is capable of making predictions of higher accuracy. Examples are categorised into two groups. In the first group, base predictions are ordinal forecasts; in the second group, base predictions come from individual non-adaptive technical analysis rules. We conducted comparison with C4.5 with respect to the examples in the second group. We conclude that FGP-1 is a useful tool that is capable of improving prediction accuracy over the given base predictions in our experiments.

However, in financial prediction, prediction accuracy is not the sole issue that concerns. For example, apart from prediction accuracy (or the *Rate of Correctness* (RC)), one may be more concerned with grasping every possible opportunity by reducing the *Rate of Missing Chances* (RMC), or with making each forecasting more reliable by reducing the *Rate of Failure* (RF), etc (formal definitions of RC, RMC, RF can be found in Section 5.2.1). In this thesis, we investigate methods to achieve a low rate of failure. A failure means a positive position predicted by the system turns out to be wrong. Another reason for us to focus on achieving a low rate of failure is that higher prediction accuracy is not available or even impossible in some cases of financial forecasting. Therefore, it would be of great value to reduce RF while an overall prediction accuracy is not affected much.

In this chapter, we shall describe our developing process toward the invention, a constrained fitness function. We call the second version of our FGP system with the constrained fitness function FGP-2. FGP-2 is intended to achieve the second research goal, i.e. to achieve a low RF. Illustration is given by an example using the Dow Jones Industrial Average (DJIA) closing index, together with analysis in detail. Moreover, FGP-2 is compared against three NNs and a linear classifier system reported in (Saad et al. 1998) with respect to the same prediction task over several individual American share prices. Results show that FGP-2 beats the linear classifier and compares favourably against the three NNs.

We review closely related work in machine learning, particularly in cost-sensitive learning. No similar technique is found. We conclude that the constrained fitness function is effective for achieving low rate of failure according to our experimental results.

5.2 Preliminary Issues

Before presenting the developing procedure toward a constrained fitness function, in this section, we need to describe some preliminary issues related. Three formal definitions of RC, RMC, and RF criteria used to assess performances of GDTs, are given, as well as two complementary criteria concerning investment performances. The DJIA data that we use in the illustrative example are visualised; major parameters for running FGP-2 are also displayed.

5.2.1 Performance Criteria

As represented in Chapter 1 (Section 1.4), FGP is designed to mainly tackle the prediction problems, P_n^r . The generated GDTs are used to predict whether or not the price will rise a required r% (e.g. r=2.2) or more within a user-specified period n (e.g., 21 days).

*# of True Negative Positions (Normal) [**TN**]*	*# of False Positive Positions (False Alarm) [**FP**]*	**Actual # of negative positions (O_-) = TN+FP**
*# of False Negative Positions (Miss) [**FN**]*	*# of True Positive Positions (Hit) [**TP**]*	**Actual # of positive positions (O_+) = FN+TP**
# of negative positions predicted (N-) = TN+FN	**# of positive positions predicted (N_+) = FP+TP**	Number of Cases

$$\text{RC} = \frac{TP+TN}{O_+ + O_-} = \frac{TP+TN}{N_+ + N_-}; \qquad \text{RMC} = \frac{FN}{O_+}; \quad \text{RF} = \frac{FP}{N_+};$$

Table 5.1: A contingency table for the two-class classification, where a specific prediction rule is invoked.

P_n^r is a two-class classification problem. Each period can be classified into either a positive position or a negative position. For simplicity, a positive position predicted by the GDT is sometimes called a *signal*, while an actual positive position is sometimes called an *opportunity*. For each GDT, we define the *rate of correctness* (RC), the *rate of missing chance* (RMC), and the *rate of failure* (RF) as its prediction performance criteria. The formula for each criterion is given through a contingency table (see Table 5.1) as follows.

Note that RMC is related to a traditional terminology, i.e. Recall = TP/ $\mathbf{O}_+$, which is identical to (1-RMC); whereas RF is related to another terminology, i.e. Precision = TP/ $\mathbf{N}_+$, which is identical to (1-RF) (cf., Manning & Schutze 1999). FGP-2 is to be instructed to use the above three criteria to form the fitness function (details will be discussed in Section 5.3). Therefore, any GDT generated should be assessed in terms of these criteria. However, an investor might like to know the expected investment performances if that GDT were used for making investment. Thus, for reference, we define two investment performance criteria: i.e. the *average annualised rate of return* (AARR), and the *rate of positive return* (RPR).

AARR has already been used as a reference performance criterion for FGP-1 (see its definition of equation 4.8, p105), whereas RPR has not been used before. RPR refers to the ratio of the number of signals, which turn out to achieve positive returns, to the total number of positive positions predicted, where a specific GDT is invoked for a finite period:

$$\text{RPR}= \frac{1}{N_+}\sum_{i=1}^{N_+}\text{I}_{\text{i}} \qquad \text{where I}_{\,i} = \begin{cases} 1 & \textit{if} \;\; \textit{ARRi} > \mathit{0} \\ 0 & \textit{otherwise} \end{cases};\, 0 < i \text{ £ } N_+ \,,$$

where N_+ is the number of positive positions generated by the GDT, and ARR_i is an *annualised rate of return* for the i^{th} signal, which has also been specified in equation 4.7 (see page 105)

We emphasis again that AARR and RPR should only be treated as references with respect to investment because both are not involved in the fitness function. The goal for modifying the fitness function is to achieve a lower RF. We shall describe an emerging overall picture concerning performances on four criteria: RC, RMC, AARR, and RPR as the RF is reduced.

5.2.2 Experimental Data

Except for those 10 individual American stock data in Section 6, all results reported in this study are based on DJIA closing index data. The data include a total of 3035 trading days from 07/04/1969 to 09/04/1981. Figure 4 displays the price series. We took the data from 07/04/1969 to 11/10/1976 (1,900 trading days) as training data (or in-sample data) to generate GDTs, and tested them on the data (or out-of-sample data) from 12/10/1976 to 09/04/1981 (1135 trading days).

The prediction problem P_n^r that we study on the DJIA data is $P_{21}^{2.2}$ (i.e. r = 2.2% and n = 21). Thus, the (f_{tr}, f_{te}) (see Equation 4.5 and 4.6, p104) for $P_{21}^{2.2}$ based on the DJIA data is (52.47%, 47.11%), which means both the training date and the test data contain roughly 50% of positive positions. For an in-depth analysis, we purposefully partition the whole test period into three mutually exclusive periods with different characteristics, i.e. a down-trend period from 12/10/1976 to 12/04/78 (378 trading days), a side-way-trend period from 13/04/1978 to 27/03/1980 (496 trading days) and a up-trend period from 28/03/1980 to 09/04/81 (261 trading days).

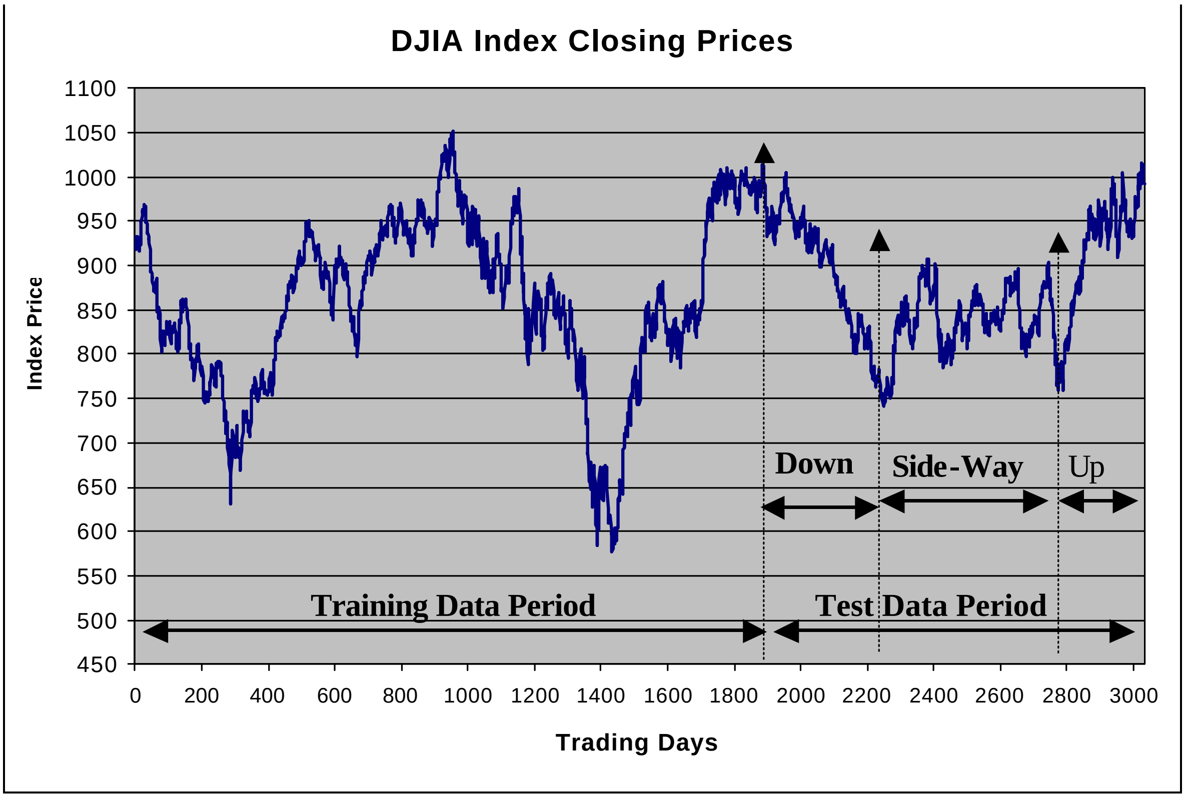


Figure 5.1**:** Experimental DJIA index data form 07/04/1969 to 09/04/1980 (3035 trading days), including 1900 trading days as training data (07/04/1969 to 11/10/1976) and 1135 trading days as test data (12/10/1976 to 09/04/1981), where (f_{tr}, f_{te}) = (52.47%, 47.11%).

5.2.3 Parameters for Running FGP

All the experiments presented in this chapter were carried out on a Pentium PC (200MHz) using a population size of 1,200. The termination condition was 30 generations or maximum of 2 hours running, whichever reached. For each independent run, when it terminated we chose a best-so-far GDT in terms of the fitness value over the training data. Then, we applied it to the test data for prediction. All results reported in the study are performances over the test data. Major parameters are displayed in Table 5.2.

The above parameter setting was applied to the experiments described in next section for running both FGP-1 and FGP-2. For the experiments on the 10 American stock data in Section 6, the parameter setting retains except for the termination condition, which was set to be maximal 50 generations.

Objective	To find GDTs that can achieve the low RF for $P_{21}^{2.2}$
Input terminals (six technical Indicators and real values)	$I_{MV\_12,\ t}$, $I_{MV\_50,\ t}$, $I_{TRB\_5,\ t}$, $I_{TRB\_50,\ t}$, $I_{Filter\_5,\ t}$, $I_{Filter\_63,\ t}$ (see definitions in section 4.3.2., p. 100) and Real values as thresholds.
Prediction terminals	{0, 1}: 1 representing "Positive"; 0 representing "Negative".
Non-terminals	If-then-else, And, Or, Not, >, <, =.
Crossover rate	0.9.
Mutation rate	0.01.
Population size	1,200.
Maximum number of generations	30.
Termination criterion	The maximum number of generations has been run or FGP-2 has run for more than 2 hours.
Selection strategy	Tournament selection, Size = 4.
Max depth of an GDT	17.
Max depth of an initial GDT	4.
Maximum run times allowed (hours)	2.
Hardware and operating system	Pentium PC 200MHz running Windows 95 with 64M RAM.
Software	Borland C++ (version 4.5).

Table 5.2: Tableau for parameters of FGP-2 experiments.

5.3 Toward A Constrained Fitness Function

5.3.1 A Linear Fitness Function

In our earlier work, the fitness function mainly used in FGP-1 is RC. Though RC is suited to search for GDTs that are able to outperform random runs and individual technical analysis rules in terms of RC, it does not allow us to focus on finding GDTs that are capable of achieving a lower RF. Thus we examined a linear fitness function[3] as follows.

$$f_{(1)} = w\_rc * RC - w\_rmc * RMC - w\_rf * RF. \qquad \textit{Where } 0£\ w\_rc,\ w\_rmc, \text{ and } w\_rf\ £\ 1 \qquad (5.1)$$

[3] If we take *w_rc=1, w_rmc=0,* and *w_rf= 0,* then $f_{(1)}$ = RC. RC is only an instance of $f_{(1)}$, so $f_{(1)}$ has a more generalised form.

It involves three performance values, i.e. RC, RMC and RF, each of which is assigned a different weight: *w_rc*, *w_rmc*, or *w_rf* respectively. Obviously, the goodness of a GDT is no longer assessed only by its RC, but by a synthetical value, which is the weighted sum of its three performance rates. By adjusting the three weights, we may attempt to place more emphasis on one performance than on the others.

In order to achieve a low RF, for example, one may try the following:

1. To assign *w_rc* a higher value (e.g. *w_rc*=1) to highly award the GDT that has a good RC performance;
2. To set *w_rf* a higher value to heavily penalise the GDT that has a poor RF performance; and
3. To assign *w_rmc* a smaller value or even zero to slightly penalise GDT that has poor performance of RMC or even not to penalise it at all.

Thus, it might be possible that $f_{(1)}$ with appropriate weights might work and lead FGP to find the GDTs with lower RFs. However, our substantial trials showed that it did not work as we expected. To a certain extent, $f_{(1)}$ does allow us to reduce RF. However, it has two drawbacks: 1) A GDT's performance is very sensitive to the sizes of the three weights and 2) results are unconsistent. We illustrate problems by one of our series of preliminary experiments in which we used the following three weights:

$$w\_rc = 1; \quad w\_rmc = 0 \quad \text{and} \quad w\_rf = a \qquad 0 < a\ £\ 1$$

First, to determine a suitable size of a, the process of trial and error is needed. Two extreme phenomena occurred. A slightly bigger a (e.g. 0.8) almost always resulted in such a GDT that did achieve a lower RF performance, even zero over training period, but triggered no signals (positive positions predicted) over the test period. That is to say, a heavier penalty on RF almost always results in a more conservative GDT in the sense that it seldom generates signals.

We refer to this extreme as the "*over-strictness*" problem. In contrast, a slightly smaller a usually resulted in such a GDT that it did not show any improvement on RF compared to that by using RC as the fitness function. We refer to this opposite extreme as the "*no-effect*" problem.

Second, even though a plausible a seems to be found (e.g. $a = 0.62$), it is too sensitive to make FGP able to generate effective GDTs reliably. For example, among 10 runs, only two runs generated a GDT each, which predicted a few correct positive positions on the test period, while other 8 runs demonstrated two similar problems, i.e. either "*over-strictness*" or "*no-effect*".

It was clear that the fitness function $f_{(1)}$ was not able to guide FGP effectively to search for good solutions in terms of the performance of RF. We need to provide a mechanism in the fitness function that is able to lead FGP to effectively seek for better solutions. In the following subsection, we shall demonstrate that a constraint is capable of taking this role.

5.3.2 A Novel Constrained Fitness Function

To resolve the above undesirable predicaments of the linear fitness function $f_{(1)}$, we introduce a constraint to $f_{(1)}$, which is the expected range of ratio of the number of positive positions predicted to the total number of training data cases. We denote the constraint with R, which consists of two elements represented by percentage, given by

$$R = [\mathrm{C_{min}}, \mathrm{C_{max}}]$$

$$\text{where } \mathrm{C_{min}} = \frac{\mathrm{P_{min}}}{\mathrm{N_{tr}}} \times 100\% , \mathrm{C_{max}} = \frac{\mathrm{P_{max}}}{\mathrm{N_{tr}}} \times 100\% , \text{ and } 0 \le \mathrm{C_{min}} \le \mathrm{C_{max}} \le 100\%;$$

$\mathrm{N_{tr}}$ is the total number of training data cases,
$\mathrm{P_{min}}$ is the minimum number of positive position predictions required, and
$\mathrm{P_{max}}$ is the maximum number of positive position predictions required.

The range of the constraint R is determined by $\mathrm{C_{min}}$, the minimum and $\mathrm{C_{max}}$, the maximum. Since ranges of *Rs* chosen are mutually exclusive in all our experiments, we would like to introduce a comparison notion for constraints. A constraint R is said to be tighter than another R'

($R < R'$) if and only if $0 < C_{min} < C_{max} \le C'_{min} < C'_{max}$.

We still take a similar formula of the fitness function as $f_{(1)}$, i.e.

$$f_{(2)} = w\_rc' * RC - w\_rmc * RMC - w\_rf * RF \quad \text{where } 0 \le \; w\_rmc, \text{ and } w\_rf \le 1 \qquad (5.2)$$

The fitness value is still a composite value which takes all three performances into account with corresponding weights. However, *w_rc'*, the weight for RC, does not take a constant like *w_rc* in $f_{(1)}$. It could also possibly take the value of zero on conditions of the size of C_+ and the constraint *R*. *w_rc'* is defined by

$$w\_rc' \;=\; \begin{cases} w\_rc & \text{if } C_+ \in R[C_{\min,} C_{\max}] \\ 0 & \text{otherwise} \end{cases} \qquad \textit{where } 0£ \; w\_rc \; £ \; 1 \qquad (5.3)$$

Note that C_+ is the percentage of the number of positive positions predicted by a GDT based on training data to the total number of training data cases, given by

$$C_+ = \frac{N+}{N_{tr}} \times 100\%, \qquad \text{where } N_+ \text{ is the number of positive positions predicted by the GDT.}$$

The range of *R* is specified with setting up two elements: C_{min} and C_{max} by the user for each run.

With the constraint *R* embedded in the fitness function, only GDTs that can satisfy the constraint are awarded to a great extent. In contrast, those GDTs, which cannot satisfy the constraint, are heavily penalised by being assigned negative fitness values, and consequently would be extinct during evolution. We use $f_{(2)}$ to denote *the constrained fitness function*. Notably, $f_{(2)}$ has a more generalised form compared to $f_{(1)}$. $f_{(1)}$ can be treated as a specific case of $f_{(2)}$, where *R* is taken as [0%, 100%], which makes *w_rc'* equal to *w_rc* because any GDT satisfies the constraint.

We call the FGP system using the constrained fitness function $f_{(2)}$ FGP-2. Note that before running FGP-2, four parameters need to be set in $f_{(2)}$, i.e. the constraint R and the three weights: *w_rc, w_rmc,* and *w_rf*. Efficacy of the constraint fitness function in FGP-2 is first demonstrated in the following experiment by comparison with FGP-1.

5.3.2 Baseline Performance

In this experiment, we took R = [35%, 50%], *w_rc* = *w_rf* = 1 and *w_rmc* = 0 for $f_{(2)}$. Such parameter choices emphasised equal importance on RC and RF, whilst fully ignoring RMC. Moreover, in each run, FGP-2 was guided to generate a best GDT, which has to be capable of invoking C_+ between [35%, 50%]. Note that C_+ satisfies the constraint R over the training data. However, over the test data, the percentage of the number of positive positions predicted by the GDT to the total number of test data cases might not lie in [35%, 50%].

We run FGP-2 10 times. The best GDT found in each run is kept. Performances of 10 GDTs with respect to the means of three prediction performances: RF, RMC, and RC; of two investment performances: AARR and RPR; and of four elements in the contingency table: TP, FP, TN and FN, are shown in Table 5.3 respectively. Note that the number of signals generated by a GDT is identical to the sum of TP and FP.

Unlike the consequences of using $f_{(1)}$, first, a GDT's performance is not sensitive to the sizes of the three weights. For other weight combinations, such as (*w_rc = 1*, *w_rf* = 0.7, 0.8 or 0.9 and *w_rmc* = 0) or (*w_rc* = 0.7, 0.8 or 0.9; *w_rf* = 1 and *w_rmc* = 0), experimental results are broadly similar. Secondly, the 10 GDTs generated have nearly equivalent performances in terms of RF and RC. Thus, results are consistent. It appears that the two major weaknesses of the fitness function $f_{(1)}$ are overcome through embedding the constraint into it.

To see whether RF is reduced, we compare these results against those by running FGP-1 in which only RC is used to measure the fitness (results are listed in Table 5.4). Results show that by using $f_{(2)}$, the mean RF is reduced by nearly 3.5% from 43.51% to 40.06%. Consequently, the

RULES	RF	RMC	RC	AARR	RPR	TP	FP	TN	FN
GDT 1_R (35,50)	0.4034	0.6402	0.5392	0.6068	0.7059	213	144	399	379
GDT 2_R (35,50)	0.4122	0.6267	0.5366	0.6383	0.6755	221	155	388	371
GDT 3_R (35,50)	0.4012	0.6622	0.5366	0.6198	0.7096	200	134	409	392
GDT 4_R (35,50)	0.4006	0.6639	0.5366	0.6260	0.7078	199	133	410	393
GDT 5_R (35,50)	0.4025	0.6740	0.5339	0.6402	0.6966	193	130	413	399
GDT 6_R (35,50)	0.4103	0.5946	0.5427	0.5826	0.6929	240	167	376	352
GDT 7_R (35,50)	0.4147	0.6639	0.5295	0.6299	0.6735	199	141	402	393
GDT 8_R (35,50)	0.3994	0.6875	0.5330	0.6398	0.6916	185	123	420	407
GDT 9_R (35,50)	0.3982	0.6655	0.5374	0.6341	0.7173	198	131	412	394
GDT 10_R (35,50)	0.3640	0.6959	0.5463	0.7202	0.7244	180	103	440	412
MEAN	0.4006	0.6574	0.5372	0.6338	0.6995	202.8	136.1	406.9	389.2
STD	0.0141	0.0299	0.0048	0.0353	0.0167	17.7	17.5	17.5	17.7

Table 5.3: The results of 10 GDTs generated by FGP-2 using *R* (35, 50). Note that each GDT name is appended with the constraint value *R* which was used in FGP-2 for generating the GDT.

RULES	RF	RMC	RC	AARR	RPR	TP	FP	TN	FN
GDT 1	0.4111	0.4459	0.5656	0.5782	0.6661	328	229	314	264
GDT 2	0.4389	0.4493	0.5410	0.5240	0.6609	326	255	288	266
GDT 3	0.4235	0.5355	0.5427	0.5504	0.6897	275	202	341	317
GDT 4	0.4502	0.4307	0.5322	0.5496	0.6460	337	276	267	255
GDT 5	0.4409	0.4409	0.5401	0.5233	0.6368	331	261	282	261
GDT 6	0.4458	0.5253	0.5269	0.5402	0.6588	281	226	317	311
GDT 7	0.4333	0.5051	0.5392	0.5490	0.6557	293	224	319	299
GDT 8	0.4361	0.3885	0.5507	0.6034	0.6636	362	280	263	230
GDT 9	0.4336	0.4527	0.5454	0.5382	0.6521	324	248	295	268
GDT 10	0.4379	0.5034	0.5357	0.5509	0.6558	294	229	314	298
MEAN	0.4351	0.4677	0.5419	0.5507	0.6586	315.1	243.0	300.0	276.9
STD	0.0111	0.0471	0.0107	0.0242	0.0139	27.9	25.1	25.1	27.9

Table 5.4: The results of 10 GDTs generated by FGP-1 (RC is the fitness function).

mean AARR dramatically increases from 55.07% to 63.38% and the mean RPR rises from 65.86% to 69.95%. It is not surprising that the mean RMC gets worse from 46.77% to 65.74%, as RMC is ignored in the fitness function by being assigned a "zero" weight. On the other hand, the constraint *R* [35%, 50%] favours such a GDT that tends to generate a fewer signals[4], and therefore results in missing more chances. Nevertheless, the mean RC only slightly decreases from 54.19% to 53.72%. The reason is that RC is still factored to be maximised in the fitness function $f_{(2)}$.

To determine whether the differences in the results are statistically significant, we use a

[4] The mean percentage of positive positions predicted over the test data produced by FGP-2 is 29.86%, which is much smaller compared to 49.16% produced by FGP-1.

two tailed paired *t*-test with α=0.001 and with df = (20 - 2) =18. The critical value obtained from the *t*-test table is 3.922. If an observed *t*-value (the absolute value) exceeds this critical value, then we conclude that there is a significant difference between the means of the two considered results. In Table 5.5 are *t*-values and their corresponding *p*-values under each criterion. Results indicate that by using the novel constrained fitness function, the generated GDTs statistically exhibit better performances under criteria of RF, AARR, and RPR at significant level of α = 0.001, though they statistically significantly grow worse with respect to RMC[5]. However, it is intriguing that that they do not show a statistically significant difference for RC (the absolute value of *t*-value is 1.16, which is less than the critical value, 3.922). That is to say, the difference in RC between the two groups could be due to chance.

Criteria	For RF	For RMC	For RC	For AARR	For RPR
t values	-4.64*	6.33*	-1.16	4.69*	4.71*
p values	0.000205	0.000005*	0.261247	0.000182*	0.000175*

Table 5.5: *t*-statistics for comparing the mean performances of the two groups with respect to the five criteria listed respectively. (Results using RC versus results using the constrained fitness function with R = [35%, 50%]). * indicates the statistically significant difference between the means of the two considered results with (α = 0.001).

In summary, the preliminary promising results using the constraint *R* [35%, 50%] demonstrate that the constraint of *R* in the fitness function is crucial for FGP-2 to achieve a lower RF steadily and effectively. We argue that the constraint embedded in the fitness function changes the landscape of search space for FGP-2, and therefore, allows FGP-2 to search for solutions to the problem addressed here in the favourable space. The strength of the constrained fitness function is further demonstrated in the following series of experiments.

[5] Since *w_rmc* is assigned zero, we do not penalise any GDT with poor performance on RMC. The generated GDT may have a very much higher RMC, as we do not care.

5.4 Analysis

5.4.1 An Overall Picture: Effects of the Constraint

Section 5.3 presents the promising results of using the constrained fitness function. The effectiveness of $f_{(2)}$, however, is only demonstrated by taking a specific *R* [35%, 50%]. We do not know what kinds of influences there are on performances of GDTs if we adjust the constraint. To further investigate the impact of the constraint *R* on FGP-2 for reducing RF, additional 5 non-overlapped *Rs* were adopted in the fitness function $f_{(2)}$ respectively. They are five mutually exclusive ranges with an interval of 5% each for *R1* [5%, 10%], *R2* [10%, 15%], and *R3* [15%, 20%], and an interval of 15% each for *R4* [20%, 35%] and *R5* [50%, 60%]. For each *R*, we ran FGP-2 10 times using all the same parameters (*w_rc* = *w_rf* =1 and *w_rmc*= 0), and then calculated its mean performance on the test data with respect to RF, RMC, RC, RPR, AARR and the mean of TP, FP, TN and FN.

R **[C_{min},C_{max}]**		**RF**	**RMC**	**RC**	**AARR**	**RPR**	**TP**	**FP**	**TN**	**FN**
[5, 10]	Mean	0.1348	0.9914	0.4819	2.2403	0.9222	5.1	1.1	541.9	586.9
	STD	0.1485	0.0063	0.0026	2.2924	0.1086	3.8	1.4	1.4	3.8
[10, 15]	Mean	0.2860	0.9405	0.4970	1.3681	0.8295	35.2	14.1	528.9	556.8
	STD	0.0622	0.0165	0.0076	0.3052	0.0440	9.8	5.2	5.2	9.8
[15, 20]	Mean	0.3102	0.8569	0.5174	0.9958	0.7902	84.7	40.4	502.6	507.3
	STD	0.0521	0.0641	0.0167	0.2550	0.0547	38.0	26.5	26.5	38.0
[20,35]	Mean	0.3600	0.7525	0.5341	0.7568	0.7361	146.5	83.3	459.7	445.5
	STD	0.0259	0.0550	0.0119	0.0955	0.0341	32.6	23.4	23.4	32.6
[35,50]	Mean	0.4006	0.6574	0.5372	0.6338	0.6995	202.8	136.1	406.9	389.2
	STD	0.0141	0.0299	0.0048	0.0353	0.0167	17.7	17.5	17.5	17.7
[50, 65]	Mean	0.4673	0.4547	0.5131	0.5226	0.6257	322.8	283.4	259.6	269.2
	STD	0.0137	0.1040	0.0164	0.0163	0.0167	61.6	55.2	55.2	61.6

Table 5.6: The mean performances on test data using six different constraint values of *R*.

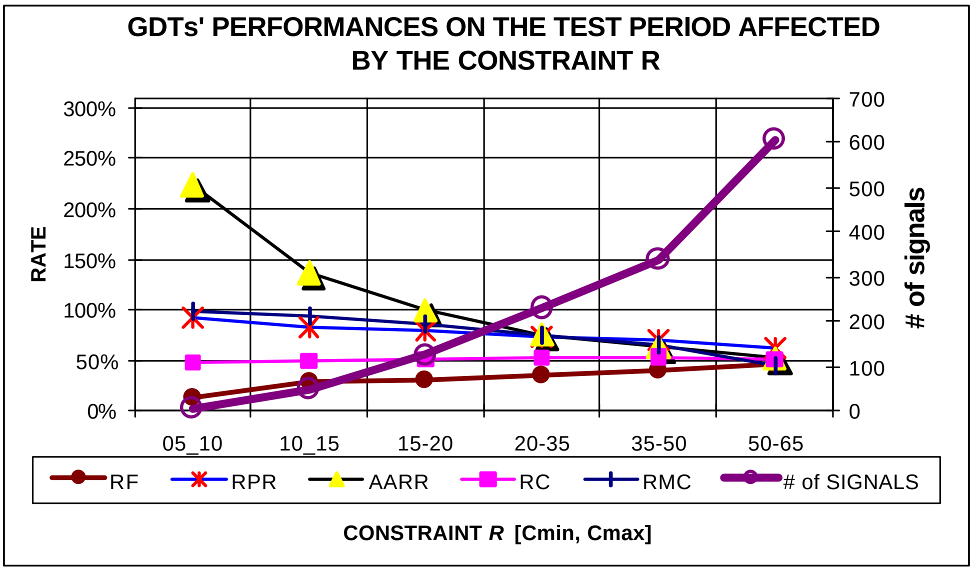


Figure 5.2: GDTs' mean performances affected by the constraint *R* based on the test data as a whole.

For brevity, here, for each *R*, we do not list details of experimental results of all 10 runs, but the mean of 10 runs under each criterion. All experimental results are shown in Table 5.6, including the preceding results using *R* [35%, 50%]. We visualise all data in Figure 5.2.

The Figure 5.2 shows that RF decreases progressively as *R* is reduced. The lowest RF (13.48%) is obtained by using the tightest *R* [5%, 10%], whereas the highest RF (46.76%) is achieved by using the loosest *R* [50%, 65%]. Six mean RFs plotted in the graph suggest that taking a reduced *R* in the fitness function may result in a lower RF.

Reduction in RF obviously benefited RPR and AARR. RPR rises from 62.57% to 92.22%. AARR increases dramatically from 52.26% to 224.03%. As a result, tighter constraints would be preferable to investors as the resultant GDT is likely to have a lower RF. That is to say, signals generated by the GDT are more reliable. The only drawback of using a tighter constraint is that the number of signals decreases accordingly. For example, the mean number of signals is 49.3 (49.3 = 35.2 + 14.1; # of signals = TP + FP) for *R* [10%, 15%] over the 1135 trading day period, in contrast to 6.2 (6.2 = 5.1 + 1.1) for *R* [5%, 10%] over the same period. If we took an even tighter *R* further, for example, *R* = [2%, 5%], eventually we would not expect any signals at all from the generated GDT, as was verified by our experiments. Such a drawback is simultaneously reflected on the increasing RMC, which ranges from 45.47% for *R* [50%, 65%] to 99.14% for *R* [5%, 10%]. So it is crucial to choose a proper *R*, which is a non-trivial task. The point that we are trying to make here is that *R* is a useful handle for turning RF in FGP-2.

Two points are worth noting further. First, as *R* reduces, RF decreases from 46.73% to 13.48% whilst RMC increases form 45.47% to 99.14%. However, RC almost remains unchanged regardless of the choices of *R*. RC ranges from 48.12% for *R* [5%, 10%] to 53.2% for *R* [35%, 50%], which are balanced around 50%. This is desirable in financial prediction. A lower RF is achieved at the cost of a higher RMC, but without sacrificing the overall prediction accuracy, i.e. RC. This will suit applications where RMC is not a major concern.

Second, when the range [C_{min}, C_{max}] is above the proportion of actual positive positions in the training data, FGP-2 could perform worse compared to FGP-1 with RC being only the fitness measure. The data in the last row of Table 5.6 are the results obtained by using *R* [50%, 65%], whose range is beyond the actual proportion (50%) of positive positions in the training data. Both RF, which is 46.73%, and RC, which is 51.31%, are worse than 43.51% and 54.19% obtained by using RC as the fitness function (see Table 5.4). This fact implies that in order to achieve a lower RF, the constraint *R* [C_{min}, C_{max}] is recommended to have a range in which C_{max} should be smaller than the proportion of actual positive positions in the training data.

In summary, in terms of the above experiments results, the size of *R* chosen has significant effect on the RF performance of the GDT generated. A tighter *R* may result in quite a lower RF, but run the risk of not generating signals for a long time period (e.g., even several years here), and therefore miss more chances. In contrast, an inappropriate *R* may result in a higher RF, which could be avoidable. The beauty is that RC is not affected significantly. This overall picture reflects the effects of the constraint on the performances of GDTs generated. Besides, our experimental results show that in order to make $f_{(2)}$ effective, one is suggested to choose a constraint *R* [C_{min}, C_{max}], where C_{max} is smaller than the actual proportion of actual positive positions in the training data. In practice, the choice of *R* might be affected by many factors such as attitudes of investors, current market trend, capital adequacy, etc. This is beyond this study. This study here focuses on the impact of choices of the constraint *R* on the RF performance of the GDT generated by FGP-2.

5.4.2 In-Depth Analysis of the Resulting GDTs

We have analysed the performances of GDTs over the test period as a whole. We have found that varying the constraint *R* can lead to varied results as expected. In general, a tighter constraint usually results in a lower RF. Meanwhile, RC is not affected significantly, though RMC increases accordingly. With regard to those generated GDTs, one may ask whether such an overall picture remains under different market circumstances. To answer this question, an in-depth analysis of those generated GDTs is necessary.

5.4.2.1 Results over Three Sub-Periods with Different Market Characteristics

To further understand the properties of the GDTs under different market characteristics, for the purpose of analysis, the whole test period was divided into three partitions, namely, down-trend period, side-way-trend period, and up-trend period (see Section 5.2.2). The differences of the three periods are not simply reflected only in the distinctive market trends visualised in Figure 5.1, but also in different proportions of opportunities (an opportunity is one actual positive position) over the period. Certainly, there are a larger proportion of opportunities (i.e. 74.3%) in the up-trend period, followed by a median proportion of opportunities (i.e. 53.0%) in the side-way-trend period and a smaller one (i.e. 35.7%) in the down-trend period. We summarise performances of GDTs over the above three sub-periods respectively and display results in Table 5.7.

Experimental results show that a similar overall picture does exist in each of the three sub-periods regardless of the market characteristics. A tighter constraint leads to a lower RF, a higher RMC, and a relatively stable RC (except for the RCs in the up-trend period using *R* [5%, 10%], *R* [10%, 15%] and *R* [15%, 20%]. A higher RC is not achievable as the proportion of actual positive positions is highly skewed from the ranges defined by those constraints (74.3% vs [5%, 10%], [10%, 15%], or [15%, 20%])). A similar overall picture can also be illustrated by the three similar performance patterns visualised for the three periods in Figure 5.3, Figure 5.4, and Figure 5.5 respectively.

Besides, we find an intriguing characteristic of the GDTs. They generate a significant different number of signals in terms of the trend types of market. During the down-trend period, the GTDs generate fewer signals compared to those in both the side-way-trend and the up-trend periods, given the same constraint *R*. We illustrate this by using the *Signal Frequency* (SFQ), given by:

$$\text{SFQ} = \frac{\text{N}_{+}(\text{the number of positive positions predicted})}{\text{The total number of cases}} \text{ x } 100\%$$

All SFQs for each *R* under the three market situations are reported in Table 5.8. Results show that given the same constraint, GDTs always have a lowest SFQ in the down-trend market, followed by that in the side-way-trend and that in the up-trend in order (except for the example of

R **[C_{min},C_{max}]**		**RF**	**RMC**	**RC**	**AARR**	**RPR**	**TP**	**FP**	**TN**	**FN**
Down Trend Period (N_{te} = 378, O_+ = 135 and *f* $_{te}$ =35.7%)										
[5, 10]	Mean	n/a	1.0000	0.6429	n/a	n/a	0.0	0.0	243.0	135.0
	STD	n/a	0.0000	0.0000	n/a	n/a	0.0	0.0	0.0	0.0
[10, 15]	Mean	0.0500	0.9778	0.6492	0.6909	0.9667	3.0	0.6	242.4	132.0
	STD	0.1225	0.0318	0.0072	0.1528	0.0816	4.3	1.9	1.9	4.3
[15, 20]	Mean	0.3367	0.9193	0.6511	0.4555	0.6917	10.9	7.8	235.2	124.1
	STD	0.3290	0.1206	0.0118	0.2982	0.3308	16.3	16.9	16.9	16.3
[20,35]	Mean	0.3442	0.8911	0.6553	0.3857	0.6923	14.7	10.0	233.0	120.3
	STD	0.3091	0.1034	0.0111	0.3890	0.3010	14.0	13.2	13.2	14.0
[35,50]	Mean	0.4710	0.7467	0.6513	0.3099	0.5882	34.2	31.0	212.0	100.8
	STD	0.0228	0.0930	0.0079	0.0454	0.0288	12.6	12.9	12.9	12.6
[50, 65]	Mean	0.6126	0.4556	0.5254	0.1811	0.4608	73.5	117.9	125.1	61.5
	STD	0.0280	0.3171	0.0887	0.0375	0.0290	42.8	74.3	74.3	42.8
Side-Way Trend Period (N_{te} = 496, O_+ = 263 and *f* $_{te}$ =53.0%)										
[5, 10]	Mean	0.1527	0.9878	0.4746	2.5580	0.8082	3.2	0.8	232.2	259.8
	STD	0.1483	0.0107	0.0043	1.7883	0.3337	2.8	1.0	1.0	2.8
[10, 15]	Mean	0.3829	0.9274	0.4845	1.2226	0.7660	19.1	11.8	221.2	243.9
	STD	0.0549	0.0219	0.0087	0.2238	0.0660	5.8	3.9	3.9	5.8
[15, 20]	Mean	0.4056	0.8582	0.4923	0.9563	0.6979	37.3	26.1	206.9	225.7
	STD	0.0539	0.0571	0.0143	0.2838	0.0734	15.0	12.9	12.9	15.0
[20,35]	Mean	0.4756	0.7627	0.4806	0.6646	0.6510	62.4	57.0	176.0	200.6
	STD	0.0299	0.0481	0.0133	0.1117	0.0429	12.7	13.2	13.2	12.7
[35,50]	Mean	0.4978	0.6749	0.4720	0.5628	0.6471	85.5	84.4	148.6	177.5
	STD	0.0217	0.0541	0.0154	0.0473	0.0133	14.2	10.8	10.8	14.2
[50, 65]	Mean	0.4962	0.5030	0.4748	0.4873	0.6343	130.7	128.2	104.8	132.3
	STD	0.0348	0.0906	0.0383	0.0933	0.0673	23.8	19.3	19.3	23.8
Up Trend Period (N_{te} = 261, O_+ = 190 and *f* $_{te}$ =74.3%)										
[5, 10]	Mean	0.0833	0.9902	0.2628	2.6002	0.9583	1.9	0.3	66.7	192.1
	STD	0.1543	0.0082	0.0055	3.3519	0.1179	1.6	0.7	0.7	1.6
[10, 15]	Mean	0.0847	0.9325	0.3004	2.2016	0.9573	13.1	1.7	65.3	180.9
	STD	0.1192	0.0376	0.0257	1.2585	0.0615	7.3	2.5	2.5	7.3
[15, 20]	Mean	0.1534	0.8119	0.3716	1.2460	0.9432	36.5	6.5	60.5	157.5
	STD	0.0542	0.0875	0.0584	0.2336	0.0284	17.0	3.2	3.2	17.0
[20,35]	Mean	0.1900	0.6423	0.4602	0.9891	0.8715	69.4	16.3	50.7	124.6
	STD	0.0516	0.0849	0.0540	0.1252	0.0451	16.5	6.1	6.1	16.5
[35,50]	Mean	0.1959	0.5716	0.4958	0.9625	0.8578	83.1	20.7	46.3	110.9
	STD	0.0384	0.0513	0.0262	0.1105	0.0403	9.9	6.1	6.1	9.9
[50, 65]	Mean	0.2303	0.3887	0.5682	1.0322	0.8316	118.6	37.3	29.7	75.4
	STD	0.0818	0.0904	0.0787	0.2106	0.0750	17.5	18.1	18.1	17.5

Table 5.7: Summarised mean results of GDTs generated by FGP-2 over three different market periods by using varied constraint Rs. (In one specific period, N_{te} is the total number of cases over test data, O_+ is the actual number of positive positions and f_{te} = O_+/ N_{te} x 100% is the proportion of actual positive positions in percentage. n/a means not available.)

R [C_{min},C_{max}]	Down-trend period	Side-way-trend period	Up-trend period
[5, 10]	0.00%	0.81%	0.84%
[10, 15]	0.95%	6.23%	5.67%
[15, 20]	4.95%	12.78%	16.48%
[20, 35]	6.53%	24.07%	32.84%
[35, 50]	17.25%	34.25%	39.77%
[50, 65]	50.63%	52.20%	59.73%

Table 5.8**:** Comparisons of SFQ for each *R* under the three different market situations.

R [10%, 15%], where the SFQ (6.23%) in the side-way-trend period is slightly bigger than the SFQ (5.67%) in the up-trend period). For example, for the *R* [15%, 20%], the GDTs generate a mean of 19 signals over a total 378 down-trend trading days. Its SFQ is 4.95% (19/378×100%) in comparison with 12.78% (63/496×100%) in the side-way-trend period and 16.48% (43/261×100%) in the up-trend period. For *R* [5%, 10%], the GDTs do not even trigger one signal in the down-trend period, in contrast to a few signals in both the side way (i.e. 4), and the up-trend period (i.e. 2.2).

It is encouraging that a) the results in the three markets are consistent with the overall picture presented in Section 5.4.1, and b) the number of signals generated from a GDT varies reasonably according to the market situation.

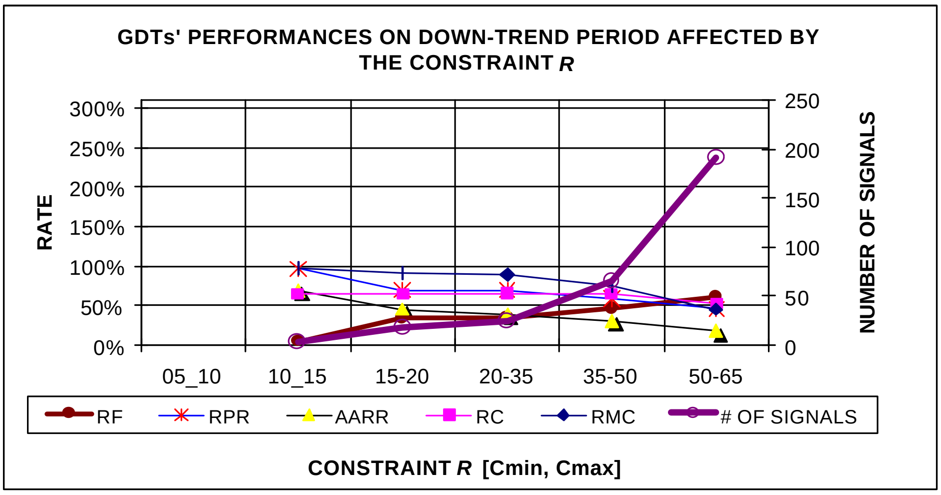


Figure 5.3: GDT mean performances in the down-trend period affected by the constraint *R*.

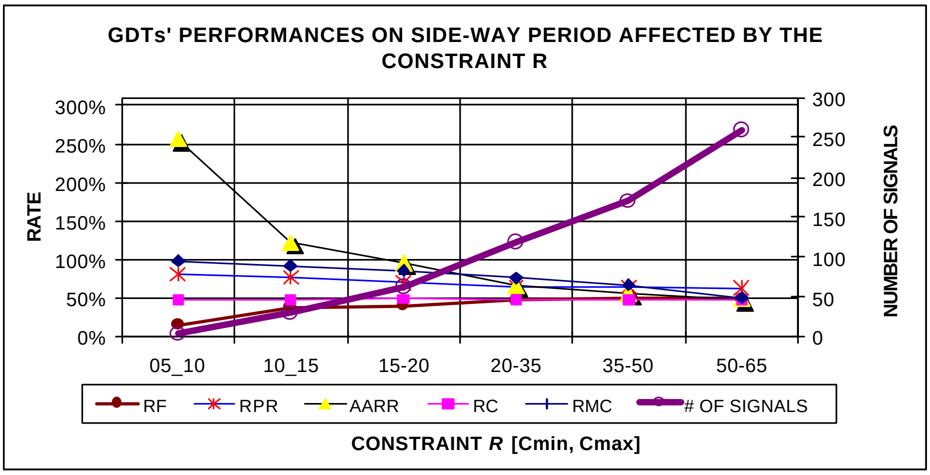


Figure 5.4: GDT mean performances in the side-way-trend affected by the constraint *R*.

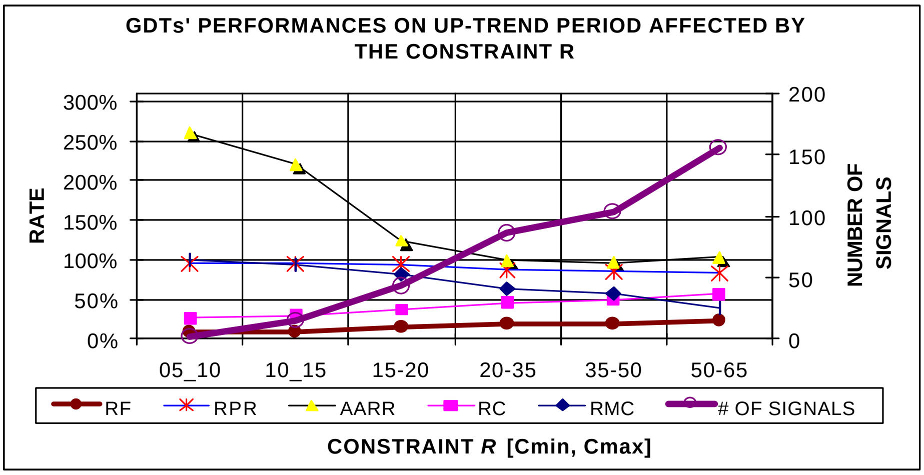


Figure 5.5: GDT mean performances in the up-trend period affected by the constraint *R*.

5.4.2.2 Further Investigation

In Section 5.4.2.1, we investigated the performances of GDTs under the three sub-periods with the three distinctive market trends. It is encouraging that the overall picture for each period is almost the same. In the case studied here, the novel constrained fitness function is effective for achieving a low rate of failure in the three typical market trends, namely, down-trend, side-way-trend, and up-trend. Now, we examine whether the overall picture with regard to GDTs' performances retains if those generated GDTs were applied to another test period over DJIA. To do this, we chose a second test period from 10/04/81 to 29/10/1984, which includes 900 trading days following the first test period (from 12/10/1976 to 09/04/1981). For simplicity, details of values are not presented here. We only visualise the results in Figure 5.6. The illustrated patterns confirm what we expect. The overall picture remains.

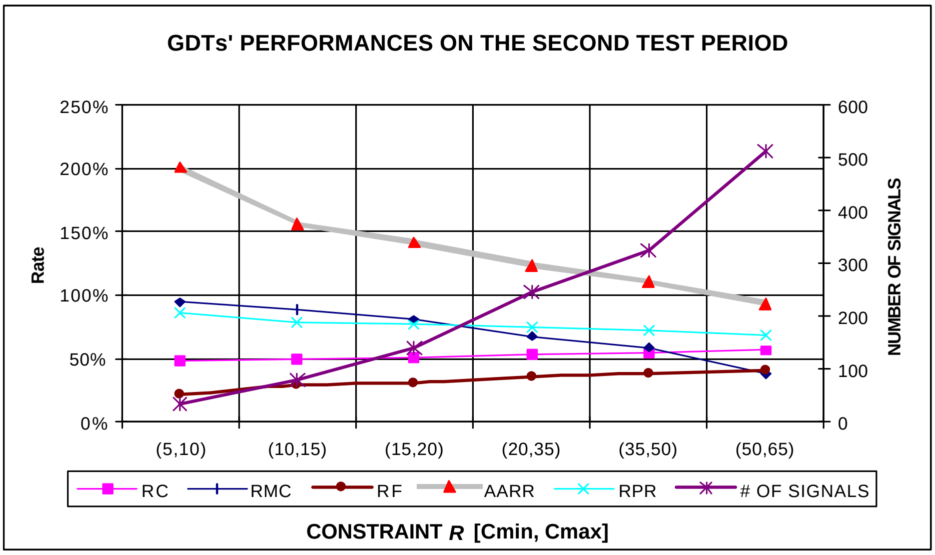


Figure 5.6: GDT mean performances on the test data II (900 trading days: from 12/10/1976 to 09/04/1981) affected by the constraint *R*.

5.5 Comparative Study

Up to this point, we only tested the proposed constrained fitness function on the DJIA index data. Should FGP-2 be effective and applicable to individual stock data? How does FGP-2 compare with other methods?

To partially answer these questions, we refer to Saad et al. (1998) in which three specially developed Neural Networks, (i.e. *Time Delay* (TDNN), *Recurrent* (RNN) and *Probabilistic* (PNN)), and a linear classifier were employed to address a similar prediction problem. They also have the goal of achieving low false alarm. Here, we compare performances based on predictions with r = 2% and n = 22 (i.e. daily predictions on whether a return of 2% or more can be achievable within the next 22 trading days).

5.5.1 The Data

We obtained from Saad the 10 stock data. The 10 stocks cover a larger variety of categories:

- Apple (AAPL, IBM (IBM), Motorola (MOT) and Microsoft (MSFT) represent the technology group which generally has high volatility.
- American Express (AXP) and Well Fargo (WFC) represent the banks.
- Walt Disney Co. (DIS) and McDonald (MCD) represent the consumer stocks.
- Public Svc New Mexico (PNM) and Energras (V.EEG) are cyclical stocks.

All data series ended at 06/03/1997, but with different starting dates. Following (Saad et al. 1998), for each stock, the last 100 days were chosen as the test data.

5.5.2 Experiments

In the experiments, for each data set, we ran FGP-2 10 times. For each run, we took 500 trading data just before 100 test data as the training data, and took a constraint R = (20%, 30%) for most data sets except for AAPL, PNM and V.EEG, for which we took a constraint R = (10%, 20%). The three weights in the constrained fitness function $f_{(2)}$ were still kept with w_rc = w_rf = 1 and w_rmc = 0 as before. The termination condition was 50 generations.

Here, for each stock data set, we report both the mean and the standard deviation of GDTs' results over 10 runs. Results reported here focus on RF and the number of signals as they are only available in (Saad et al. 1998). In order to compare fairly, for each stock data set, we also select the best GDT from the 10 runs in terms of RF and report its results. This strategy follows the one in (Saad et al. 1998). Note that a better method should be capable of achieving a lower RF and meanwhile producing a larger number of positive positions.

5.5.3 Results

Table 5.9 lists all performance results of the three different NNs, the linear classifier and FGP-2 on the 10 stocks with respect to RF and the number of signals. The "Total" column summarises the total number of signals on all 10 stocks for each method. The last column, "Ave." column reports the average rate of failure over the 10 stocks.

Like NNs., FGP-2 out-performs the linear classifier for all the stocks in terms of RF. The 10 best GDTs produced 385 signals totally, which is slightly more than 373, produced by the linear classifier. However, the average RF of the GDTs found, 5.08% is much better than 18.62%, the average RF of the linear classifier.

Results by the best of the GDTs are either as good as or better than those of NNs in terms of the number of "zero" prediction failure over the total 10 stocks. The 10 best GDTs achieved 8 zero-RFs, in contrast, TDNN got 8 zero-RFs as well; PPN, 2 zero-RFs; and RNN, 5 zero-RFs. Though both the best GDTs and TDNN achieve equally 8 zero-RFs, the total number of signals produced by GDTs over all 10 stocks is more than twice as large as that by TDNN (i.e. 385 vs 186). In terms of the average RF, the 10 best GDTs, which achieve a mean RF of 1.29%[6] over the 10 data sets, out-perform each of the three NNs, which achieve the average RFs of 3.05%, 3.61%

[6] The favorable results by FGP-2 may be partly due to the rather bullish market over test period in which over 50% of the positions are positive for all the shares; e.g. 87% of the positions were positive for MSFT and 92% for AXP.

and 7.56% respectively.

Our conclusion, based on Table 5.9, is that FGP-2 performs significantly better than the linear classifier and favourably compares against each of the three NNs. Moreover, FGP-2 has its distinctive advantages: a) it generates the GDTs that the user can interpret, and b) it can generate GDTs with varied RF performances by tuning the constraint in the fitness function. These are not available to the methods compared here.

The results of GDTs presented for comparison are merely based on one set of better solutions that are chosen by us. The solutions we think have a good trade-off between the performance of RF and the quantity of signals. Numerous potential solutions are still available if different constraints were applied to the task. By turning the constraint in the fitness function, either a further lower RF would be available at the price of reducing the number of signals or a further higher RF would be obtained with the consequence of increasing the number of signals. This provides users with more options.

Stocks			AAPL	IBM	MOT	MSFT	AXP	WFC	DIS	MCD	PNM	V.EEG	Total	Ave.
Profit Opp. (r=2%;n=22)			62	72	81	87	92	85	74	73	50	70	746	74.6
PPN	Total N+		51	25	48	49	20	45	19	4	63	14	338	
	RF (%)		7.84	4.00	18.75	4.08	0.00	4.44	0.00	0.00	36.50	0.00		7.56
TDNN	Total N+		10	9	27	61	17	19	7	6	22	8	186	
	RF (%)		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	18.00	12.50		3.05
RNN	Total N+		16	22	33	46	49	29	48	53	35	37	368	
	RF (%)		0.00	0.00	3.03	2.17	0.00	0.00	0.00	5.66	17.14	8.11		3.61
The Linear Classifier	Total N+		82	24	87	17	10	22	2	32	20	76	372	
	RF (%)		31.71	20.83	18.39	0.00	0.00	13.64	0.00	21.88	60.00	19.74		18.62
Mean and STD of 10 GDTs	Total N+	Mean	18.5	68.7	20.7	26.8	38.3	66.6	20.1	40.2	23.4	49.4	373	
		STD	9.9	3.9	5.1	6.2	9.9	11.1	3.1	1.8	5.9	9.6		
	RF (%)	Mean	9.16	10.15	1.33	3.10	3.72	8.20	0.40	0.00	13.07	4.83		5.08
		STD	5.66	1.13	2.82	2.47	3.10	2.33	1.30	0.00	12.30	3.90		
The Best GDT	Total N+		4	70	28	33	39	69	22	43	28	49	385	
	RF (%)		0.00	8.57	0.00	0.00	0.00	4.35	0.00	0.00	0.00	0.00		1.29

Table 5.9: Performance comparisons among three NNs, a linear classifier and FGP-2 in terms of RF and N_+ (the total number of position positions produced).

5.6 Related Work

The novel constrained fitness function that we invent and incorporate into FGP-2, aims to achieve a low rate of failure. In essence, by virtue of the constrained fitness function, FGP-2 is capable of reducing the number of false positive positions (FP) at the expense of increasing the number of false negative positions (FN), while maintaining prediction accuracy. In the context of investment, such a trade-off is valuable. Mistaking an actual negative position for a positive position (a false positive position) is much more costly than the opposite of mistaking an actual positive position for a negative position (a false negative position), because the latter error only means missing a chance, no loss at all.

In this work, our research target is closely related to cost-sensitive learning, which is a subject of a burgeoning literature in machine learning (Turney (1997) provides an online bibliography on this topic). More specifically, the target that the novel constrained fitness function is designed to attack is similar to misclassification-cost classification. Moreover, our research method is also closely related to classification-oriented evolutionary algorithms.

In machine learning, approaches to misclassification-cost classification could be considered to fall into three main categories in terms of stages of processing induction trees.

1) **Pre-processing: re-sampling training data**

One currently available procedure of this type is stratification - changing the frequency of classes in the training data in proportion to their cost (Breiman et al. 1984; Chan & Stolfo 1998; Provost & Buchanan 1995). Underlying such research work is the fact that the training class distribution likely affects the performance of the learned classifier and results in classifiers with varied performances with respect to misclassification cost.

2) **During processing: varieties of biases applied in the process of building decision trees**

- Robert et al. (1995) applied methods in the process of building decision trees by taking misclassification costs into account. In one approach, cost factors are used in the class selection criterion at the leaves of the decision tree. In another approach, in the test selection criterion at the branches of the decision tree.

- Cost-sensitive specialisation (Webb 1996) involves specializing aspects of a classifier associated with high misclassification costs and generalizing those associated with low misclassification costs, aimed at reducing the costs of misclassification errors.
- Bradford et al. (1998) apply both an extended cost-complexity pruning to loss and a Laplace correction based decision pruning to minimizing loss.

3) **Post-processing: adjustment of threshold or ordering rules generated**

- Fawcett and Provost (1997) consider non-uniform cost per error in their cellular phone detection task and exhaustively searched (with a fixed increment) for the linear Threshold Unit's threshold that minimize the total cost.
- Pazzani et al. (1994) present a method, called RCO (Reduced Cost Ordering) algorithms, which select and order the rules generated by any rule learner such as C4.5, FOCL, to minimize misclassification costs.

As work in *boosting* and *bagging* has become more convincing (Breiman 1994; Freund & Schapire 1996; Quinlan 1996a), recently, methods based on these techniques have been developed to address classification cost. Such methods involve re-sampling training data and altering the empirical biases of the learning system in the process of decision tree inductions (see, e.g., Ting & Zheng 1998; Fan et al. 1999; Domingos 1999).

The above approaches in machine learning demonstrated that they were superior to the related learning systems that did not account for misclassification cost. However, none of them can provide mechanisms to reduce the concerned misclassification errors (e.g. the rate of failure) gradually with some control to some extent. In contrast, the constraint embedded in the fitness function $f_{(2)}$, provides a useful handle for tuning the misclassification error (e.g., the rate of failure in FGP-2). In our cases, the tighter the constraint is, the lower rate of failure is achievable at the cost of missing more chances.

Research work that applies evolutionary algorithms to classification problems is also related to FGP. Frey and Slate (1991) applied a genetic algorithm (in particular, a learning classifier system (LCS)) to letter recognition. DeJong et al. (1993) and Janikow (1993) presented more successful work. There are also several papers that address classification problems using genetic programming (e.g., Ngan et al. 1998; Nikolaev & Slavov 1997; Bojarczuk et al. 1999). However, none of the above work has the capability of attacking cost-sensitive classification problems.

To our best knowledge, ICET (Turney 1995) is the only system that not only takes misclassification costs into account but also involves genetic algorithms. However, unlike FGP, in which genetic programming straightforward plays a main role, ICET uses genetic algorithm as a supplementary means of finding a set of better parameters for a decision tree induction algorithm. The fittest tree is constructed directly through decision tree induction algorithms, rather than genetic algorithms. The novel constrained fitness function that we invent makes it possible for genetic programming to act as a main framework to approach cost-sensitive classification problems. Besides, like other cost-sensitive methods in learning system, ICET cannot provide the mechanism to find varied potential solutions either.

We argue that such a mechanism is important and desirable. Solutions with varied performances with respect to misclassification costs provide the user with multiple options. In practice, any one of choices may be interesting and valuable to a specific group of users because different users have different preferences. In FGP-2, varying the constraint, embedded in the fitness function, can lead to different GDTs with varied RFs as expected. The investor tends to choose the GDT that is likely to reflect his/her risk preference.

5.7 Summary and Conclusion

5.7.1 Summary

FGP-1, the first version of FGP, has been demonstrated to be useful for improving prediction accuracy in terms of RC. However, in many real-world prediction problems, RC is not the sole concern. Aimed at achieving a low RF prediction, we developed the second version of FGP, FGP-2, in which we use a novel constrained fitness function.

We accomplished the enhancement through modification of the fitness function. The novelty of FGP-2 lies in a crucial constraint embedded in the fitness function. The proposed constrained fitness function is more general and superior to the previous fitness function, i.e. RC, used mainly in FGP-1. Its effectiveness was investigated and demonstrated in a series of experiments using different *Rs* on the DJIA data. In general, varying the constraint results in expected results. A tighter constraint usually results in a lower RF. Meanwhile, RC is not affected significantly, though RMC increases accordingly. This overall picture holds in the three sub-periods with three different market situations, namely, down-trend, side-way-trend, and up-trend, and a further additional test period. Moreover, in our tests, the generated GDTs seem to cope well with different market trends as the quantity of positive positions predicted reasonably varies in accordance with the market properties in our case.

By tuning the constraint *R*, FGP-2 is capable of achieving different levels of RF. This makes FGP-2 attractive. It provides users with a means to their preferences.

To evaluate FGP-2, we compare FGP-2 against three NNs, and a linear classifier reported in (Saad et al. 1998) based on a specific prediction problem over the 10 American stock data. FGP-2 beats the linear classifier and favourably compares against the three NNs in terms of the performance of RF. FGP-2 exhibits its superiority over the three NNs with respect to the quantity of positive positions predicted. The fact that the parameter of constraint *R* in the fitness function is adjustable makes FGP-2 more attractive.

The work in this chapter is closely related to cost-sensitive learning, as well as work using classification-oriented evolutionary algorithms. The work of using a novel constrained fitness function in GAs to attack misclassification-cost classification problems is novel. No similar techniques have been found so far in our review. The applicability of the novel constrained fitness function to other domains is worth further investigation.

5.7.2 Conclusion

In order to achieve the second research goal: to reduce the RF, we developed FGP-2. FGP-2 implements a method for tuning the RF performance to a certain extent. This is achieved by introducing a novel constrained fitness function to FGP.

In this chapter, the experimental results produced by FGP-2 demonstrate that FGP-2 is capable of achieving a lower rate of failure (RF), at the cost of a higher rate of missing chances (RMC), without sacrificing the overall prediction accuracy of the system (RC). By tuning the constraint parameter in the fitness function, users can generate GDTs to suit their preferences with regard to RF and RMC.

In the next chapter, we shall examine whether this overall picture remains if FGP-2 is applied to other similar prediction tasks or other data sets.

Chapter 6

Discussion

6.1 Introduction

In Chapter 5, we described the procedure of developing a constrained fitness function in FGP-2. The target of using FGP-2 is to achieve the second research goal: a low rate of failure. By using the constrained fitness function, FGP-2 is capable of generating GDTs with the low rate of failure in the experiments using the DJIA daily closing prices over an eleven-year period. We analysed the behaviour of FGP-2 by varying the parameter of constraint *R* in the fitness function. A tighter *R* tends to lead to a lower RF without affecting the overall RC significantly, though at the price of a higher RMC. The effectiveness of FGP-2 was further demonstrated by the comparison of FGP-2 with three NNs and a linear classifier over the 10 American stock data.

In this chapter, we would like to further investigate the effectiveness and applicability of FGP-2. We focus our investigation on the utility of the constrained fitness function for achieving a low RF. We would like to know whether the overall picture remains under several different circumstances.

Our investigation here is motivated by several questions that arise in connection with the results of FGP-2 obtained in the preceding chapter. These questions are:

1. Is FGP-2 with the constrained fitness function effective for prediction task P_n^r over a short period?
2. Can FGP-2 achieve the consistent results under the unbalanced cases (see the definitions in Section 6.3.1), where both f_{tr} and f_{te} (see Equation 4.5 and 4.6, page 104) are no longer

around 50% in the data used?

3. Is FGP-2 applicable to a down-trend market?

To answer the above questions, we carried out a series of experiments. We completed three separate experiments.

The first is a study of $P_5^{0.8}$ using DJIA data over a relatively short period (i.e. n = 5 trading days, as opposed to 21 trading days). This is intended to address the first question. We shall present and discuss the results, as opposed to the results of study on $P_{21}^{2.2}$ presented in Chapter 5. The second experiment is actually a set of tests over DJIA data, which involve the prediction tasks of P_{21}^3, P_{21}^4, and P_{21}^5. This attempts to address the second question. The third group of experiments is carried out on a set of foreign exchange data of \$US/£BP (US Dollar against British Pound) for attacking the prediction tasks of P_{21}^1 $P_{21}^{1.5}$, and P_{21}^2. The whole period of the foreign exchange data set that we adopted shows a general down-trend, as displayed in Figure 6.7. This group of tests is intended to address the third question and partially address the second question.

With the three groups of experiments, the hope is that a series of empirical results obtained may help us to understand the applicability of FGP-2, and manifest the strength and weakness possessed by FGP-2 with the novel constrained fitness function. We shall describe the experimental methods used, and then present and discuss the results.

6.2 Testing over a Short Period

We have completed a series of experiments over both the DJIA and the 10 American stock data. However, the prediction tasks were restricted to a period of 21 trading days, which is a middle term from the perspective of investors. However, in finance, predictions over a shorter period are not uncommon. The question arises concerning the applicability of FGP-2 over a short period.

To answer this question, we choose $P_5^{0.8}$ and ask FGP-2 to attack it. The data used are the same as the DJIA data used previously, as shown in Figure 5.1 (see page 120). Note that (f_{tr} , f_{te}) is (50.00%, 51.63%) in the case of $P_5^{0.8}$ and the data used. Both values are around 50%. Such a case is referred to as a balanced case, as opposed to an unbalanced case, where both values of f_{tr} and f_{te} are not around 50% (both cases are to be defined in Section 6.3).

The objective of this experiment is to examine further the capabilities and limitations of FGP-2. In particular, we would like to know whether results obtained over a shorter period by FGP-2 are consistent with what we found over a middle-sized period. In this experiment, we took the same parameters as those used in the previous experiments over the DJIA data (some of which are shown in Table 5.2 (see page 121)), and the same indicators as input to FGP-2. We chose different intervals of R, which are mutually exclusive. For each R, we ran FGP-2 10 times. For simplicity, in Table 6.1 we only report the mean results over 10 runs and their corresponding standard deviations for each R chosen, respectively. All results are visualized in Figure 6.1.

Results in Table 6.1 show the similar overall picture, which has already been manifested in the preceding study over a middle-sized period (see Table 5.6, p128). Varying the constraint in the fitness function can lead to varied results accordingly. A tighter constraint results in a lower RF without affecting the RC much, though at the price of the increased RMC. Consequently, the number of positive position predicted is gradually reduced as the constraint chosen becomes tighter. However, both RPR and AARR increase progressively as the constraint gets tighter, which indicates that investors may benefit more by choosing a tighter constraint R. Figure 6.1 visualises the patterns.

R [C_{min},C_{max}]		RF	RMC	RC	AARR	RPR	# of signals
[5, 10]	Mean	0.1711	0.9838	0.4902	2.4176	0.8352	11.6
	STD	0.1556	0.0073	0.0038	1.0980	0.1589	6.0
[10, 15]	Mean	0.2976	0.9553	0.4955	1.5916	0.7326	39.0
	STD	0.1005	0.0161	0.0068	0.5117	0.0913	19.9
[15, 20]	Mean	0.3655	0.8807	0.5089	1.1661	0.7009	111.2
	STD	0.0450	0.0160	0.0061	0.2132	0.0453	20.4
[20, 35]	Mean	0.3805	0.8514	0.5127	1.0541	0.6763	141.3
	STD	0.0320	0.0333	0.0077	0.1504	0.0335	34.9
[35, 50]	Mean	0.4437	0.7442	0.5101	0.7219	0.6126	269.8
	STD	0.0249	0.0291	0.0110	0.1032	0.0250	34.1

Table 6.1: GDTs' mean performances of over a short period (e.g., 5 trading day) on DJIA.

For example, for *R* [5, 10], which is the tightest constraint chosen, FGP-2 achieved the lowest RF (17.11%), but with the highest RMC (98.39%) and a lowest RC (49.02%), which is around 50%. In contrast, for *R* [35, 50], which is the loosest constraint chosen, the acquired RF is the highest (44.37%), but with the lowest RMC (74.42%), a moderate RC (51.01%), which is around 50% as well.

It is worth noting again that all RC performances obtained are approximately around 50% level regardless of the constraint *R* chosen. However, differences in RF and RMC are significant.

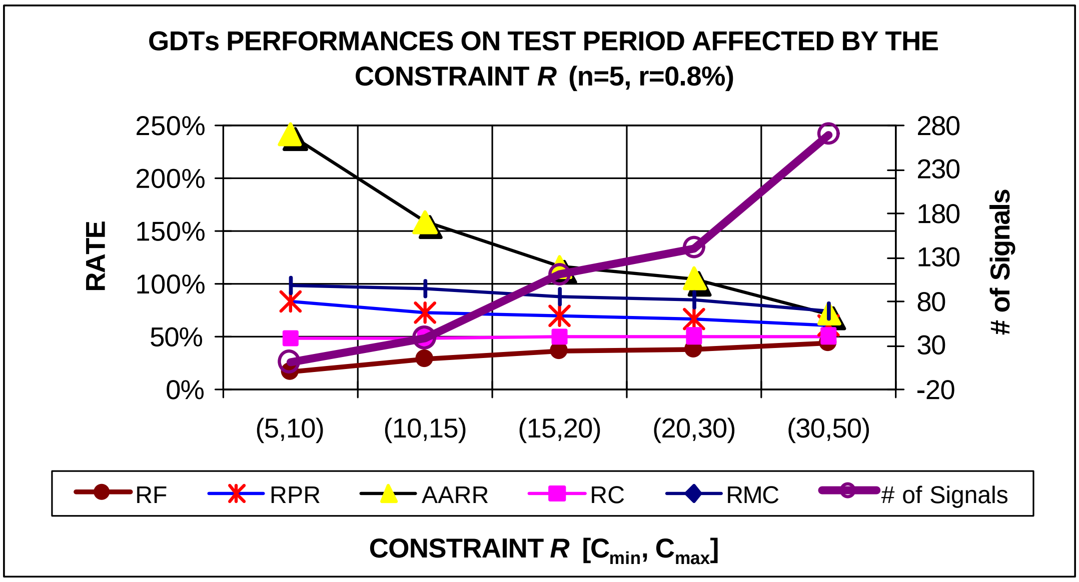


Figure 6.1: GDT performances over a shorter period (e.g. n=5 trading days) affected by the constraint.

For example, the maximum difference in RF is 27.26% (44.37% - 17.11%). The maximum difference in RMC is 23.96% (98.38% -74.42%). This suggests again that the reduced RF is at the cost of increased RMC, but not RC.

6.3 Testing in Unbalanced Cases

So far, the prediction task P_n^r that have been attacked by FGP-2 is restricted to a kind of case where the number of actual positive positions is roughly 50% of whole data cases over both a training and a testing period respectively. We would like to refer to such cases as *balanced cases* (a more formal definition will be given in the next section). Though balanced cases are common in the study of machine learning, unbalanced cases are ubiquitous in the realistic world. In the case the prediction tasks P_n^r , unbalanced cases can be easily formed due to varieties of combinations of *n* and *r*.

In this section, we shall present the study on the effectiveness of FGP-2 in the unbalanced cases. We shall discuss the empirical results. Meanwhile, we shall point out the weaknesses of FGP-2 found in these experiments.

6.3.1 Definitions of Balanced Cases and Unbalanced Cases

Before presenting our experiments, we would like to give our definitions regarding balanced cases and unbalanced cases with respect to P_n^r .

There are a variety of combinations between the size of *r* and the length of *n* for P_n^r . The choices of two parameters rely on the preference of the user. Given a fixed *r* and a fixed *n*, however, the P_n^r can only possibly categorized into either a balanced case or an unbalanced case, which are defined as follows based on a pair of values (f_{tr} , f_{te}) (see its definition at p104).

- **Balanced case**: the prediction task in which both “$45 \leq f_{\text{tr}} \leq 55$” and “$45 \leq f_{\text{te}} \leq 55$” are met. In other words, a prediction situation that the number of actual positive positions and number of actual negative positions are roughly around 50% both on the training data and on the test data.

- **Unbalanced case**: the prediction task that does not belong to the balanced case.

Obviously, there are two different sets of unbalanced cases. One is a set of unbalanced cases where the number of actual positive positions is less than the number of actual negative positions. Another is an opposite set of unbalanced cases where the number of actual positive positions is more than the number of actual negative positions. We call the former a negative unbalanced case, and the latter a positive unbalanced case. A positive unbalanced case is not particularly of interest to us, as it is relatively easier for FGP-2 to achieve a low RF (for example, see the results obtained over the 10 American stocks in Chapter 5).

Negative unbalanced cases attract more attention from us (in what follows, we focus on this kind of the unbalanced case, as opposed to the positive unbalanced case). We use three different terms, namely, a *slightly unbalanced* case, a *moderately unbalanced* case or a *severely unbalanced* case, to distinguish the extent of the difference between the number of actual positive positions and the actual negative positions in data. The three definitions are below:

- **A slightly unbalanced case**: the prediction task in which both “$35 \le f_{\text{tr}} < 45$” and “$35 \le f_{\text{te}} < 45$” are met.
- **A moderately unbalanced case**: the prediction task in which both “$25\le f_{\text{tr}} <35$” and “$25 \le f_{\text{te}} < 35$” are met.
- **A severely unbalanced case**: the prediction task in which both “$0 < f_{\text{tr}} < 25$” and “$0 < f_{\text{te}} < 25$” are met.

Note that in the last chapter, FGP-2 was mainly investigated and analysed in a balanced case, which is the prediction task $P_{21}^{2.2}$ over the DJIA data. In the following experiments, FGP-2 is investigated into three types of the above unbalanced cases, which are formed over the DJIA data as well.

6.3.2 Experiments

In the following series of experiments, we ran FGP-2 using the same DJIA data and GP parameters that were used for addressing $P_{21}^{2.2}$. Three prediction tasks that we selected are P_{21}^{3}, P_{21}^{4}, and P_{21}^{5}, which make predictions over the same middle-term period, i.e. 21 trading days. Due to a different expected return *r* chosen, the potential predictive patterns, which FGP-2 attempts to find, might be different for each prediction. The choice of the above three prediction tasks makes it easy for us to compare them against the balanced case $P_{21}^{2.2}$, whose results were already presented in the last chapter.

The three prediction tasks: P_{21}^{3}, P_{21}^{4}, and P_{21}^{5}, belong to unbalanced cases. Note that as the *r* increases, the prediction task P_{n}^{r} gets unbalanced to a greater extent. As the involved (f_{tr}, f_{te}) is (42.95%, 41.32%), P_{21}^{3} belongs to a slightly unbalanced case. Similarly, P_{21}^{4} and P_{21}^{5} are categorised into a moderately unbalanced case with a (f_{tr}, f_{te}) = (30.37%, 26.52%) and a severely unbalanced case with a (f_{tr}, f_{te}) = (21.58%,16.12%) respectively.

6.3.3 Results and Discussion

To address the slightly unbalanced case P_{21}^{3}, we took five mutually exclusive *R*s as the constraints in the fitness function respectively. Note that the C_{max} in the loosest constraint *R* [30%, 40%] reported here is limited at 40%, which is less than the f_{tr} (42.95%). As discussed in the case of $P_{21}^{2.2}$, taking a constraint that is larger than the f_{tr} would possibly make the constrained fitness function not work for achieving a low RF (see Section 5.4.1). This is also true in this slightly unbalanced case (our experimental results confirm this, though results are not shown here). Similarly, we considered the fact when we chose the constraints for approaching both P_{21}^{4} and P_{21}^{5} .

For each constraint chosen, we ran FGP-2 10 times. Table 6.2 shows the experimental mean results of the 10 GDTs obtained. Figure 6.2 depicts an overall picture in which performances of GDTs are affected by the constraint chosen.

R [C$_{min}$,C$_{max}$]		RF	RMC	RC	AARR	RPR	# of signals
[5, 10]	Mean	0.0998	0.9885	0.5903	1.3922	0.9384	6.8
	STD	0.1925	0.0079	0.0029	0.9132	0.1299	6.0
[10, 15]	Mean	0.2490	0.9237	0.6068	1.3388	0.8676	48.9
	STD	0.1233	0.0131	0.0077	0.3104	0.0588	14.0
[15, 20]	Mean	0.4043	0.8825	0.6015	0.8573	0.7700	93.5
	STD	0.0740	0.0100	0.0110	0.0937	0.0642	16.0
[20, 30]	Mean	0.5005	0.8360	0.5856	0.5899	0.7022	155.2
	STD	0.0290	0.0192	0.0082	0.0855	0.0606	27.0
[30, 40]	Mean	0.5417	0.7424	0.5641	0.4276	0.6206	267.3
	STD	0.0382	0.0570	0.0256	0.0668	0.0452	77.0

Table 6.2: GDT performances affected by the constraint in the unbalanced case: P_{21}^{3} with (f_{tr} = 42.95%, f_{te}= 41.32%).

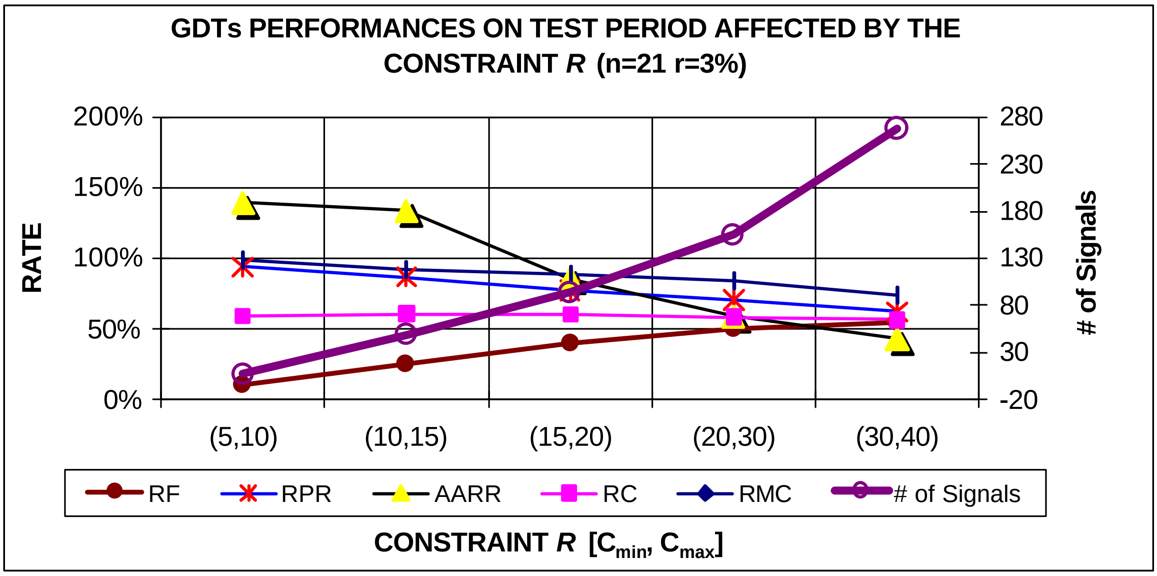


Figure 6.2: GDT performances affected by the constraint in the unbalanced case: P_{21}^{3} .

The overall picture associated with this slightly unbalanced case, i.e. P_{21}^{3}, displays a very similar pattern as the one in the balanced case, $P_{21}^{2.2}$. A tighter constraint leads to a lower RF at the price of a higher RMC, while maintaining a relatively stable RC. Experimental results show that in the slightly unbalanced case, P_{21}^{3}, FGP-2 works as well as in the balanced case, $P_{21}^{2.2}$.

R [C_{min},C_{max}]		RF	RMC	RC	AARR	RPR	# of signals
[5, 10]	Mean	0.2268	0.9864	0.7371	1.4834	0.9746	5.6
	STD	0.1670	0.0060	0.0013	1.2485	0.0541	2.1
[10, 15]	Mean	0.4689	0.9269	0.7360	1.0409	0.8123	42.6
	STD	0.0853	0.0199	0.0065	0.2822	0.0484	10.7
[15, 20]	Mean	0.5834	0.8787	0.7211	0.6419	0.7648	88.6
	STD	0.0609	0.0184	0.0098	0.0822	0.0596	10.9
[20, 30]	Mean	0.6594	0.8183	0.6889	0.4356	0.6905	161.5
	STD	0.0374	0.0229	0.0136	0.0575	0.0693	9.4

Table 6.3: GDT performances affected by the constraint in the unbalanced case: P_{21}^{4} with (f_{tr} = 30.37%, f_{te}= 26.52%).

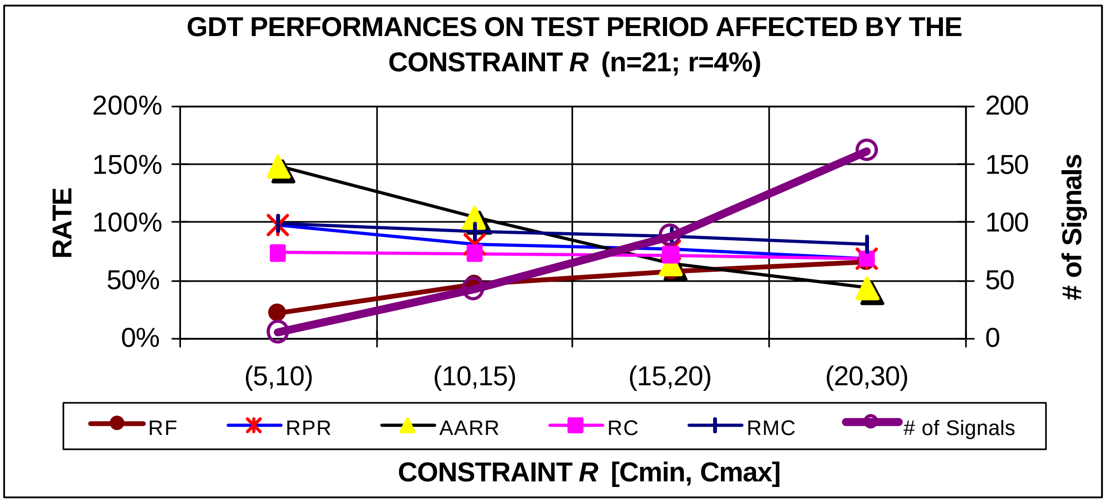


Figure 6.3: GDT performances affected by the constraint in the unbalanced case: P_{21}^{4}.

Similarly, our experimental results of applying FGP-2 to both P_{21}^{4} and P_{21}^{5} are reported in Table 6.3 and Table 6.4 respectively, as well as virtualised in Figure 6.3 and Figure 6.4.

Four different constraints (see the first column in the Table 6.3) were used to approach

P_{21}^{4}. The C$_{\text{max}}$ (30%) in the loosest constraint chosen is restricted to be less than f_{tr} (30.37%). For this moderately unbalanced case, the GDTs generated are still able to achieve satisfactory lower RFs. For example, taking the loosest constraint *R* [5%, 10%] into the constrained fitness function, FGP-2 achieved a mean of RF (22.68%). Moreover, the overall picture retains as does in the slightly unbalanced case of P_{21}^{3}. These experimental results show that FGP-2 works well and achieves acceptable lower RFs.

R [C$_{\text{min}}$,C$_{\text{max}}$]		RF	RMC	RC	AARR	RPR	# of signals
[5, 10]	Mean	0.4533	0.9809	0.8391	0.8614	0.9133	6.6
	STD	0.2289	0.0116	0.0022	0.6244	0.2218	3.6
[10, 15]	Mean	0.7106	0.9219	0.8189	0.5123	0.8297	51.1
	STD	0.0474	0.0103	0.0084	0.0921	0.0456	12.5
[15, 20]	Mean	0.7783	0.8995	0.7947	0.3652	0.7733	86.8
	STD	0.0484	0.0094	0.0180	0.0747	0.0816	21.7

Table 6.4: GDTs performance affected by the constraint in the unbalanced case: P_{21}^{5} with (f_{tr} = 21.58%, f_{te}= 16.12%).

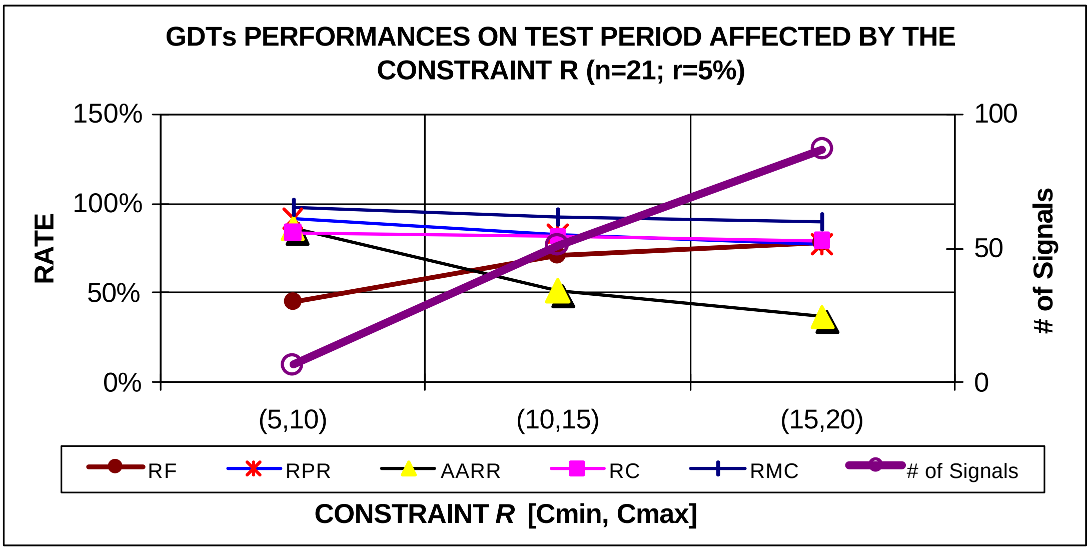


Figure 6.4: GDT performances affected by the constraint in the unbalanced case: P_{21}^{5}.

Three different constraints (see the first column in the Table 6.4) were used with the C$_{\text{max}}$ (20%) in the loosest constraint (*R* [15%, 20%]) less than f_{tr} (21.58%). The overall picture seems to remain. However, unlike the cases of P_{21}^{3} and P_{21}^{4}, all three means of RFs obtained are not acceptable. Even the lowest mean of RFs is 45.33%, which is nearly two times higher than RF

(22.68%) with P_{21}^{4}, and much higher than RF (9.98%) with P_{21}^{3}. The results show that in the severely unbalanced case, P_{21}^{4}, FGP-2 dose not achieve an acceptable lower RF as happen in the balanced, slightly unbalanced, and moderately unbalanced case. This fact suggests that it is preferable not to ask FGP-2 to address the prediction task P_{n}^{r}, which falls into a severely unbalanced case. FGP-2 may not generate an acceptable low RF as expected.

In terms of our experimental results with regard to three unbalanced cases, namely, P_{21}^{3}, P_{21}^{4}, and P_{21}^{5}, together with results with respect to the balanced case, $P_{21}^{2.2}$, we emphasis some points as follows.

1. The overall picture which results from using the constrained fitness function remains in the four types of cases studied here, namely, the balanced case, the slightly unbalanced case, the moderately unbalanced case, and the severely unbalanced case. A tighter constraint leads to a lower RF at price of a higher RMC without affecting the overall RC. A reduction in RF benefits RPR and AARR.

2. FGP-2 works reasonably well for achieving a lower RF in the balanced case, the slightly unbalanced case, and the moderately unbalanced case. However, it does not achieve an acceptable lower RF in the severely unbalanced case. This implies that it is preferable not to ask FGP-2 to address a prediction task P_n^r, which falls into the severely unbalanced case.

3. In this study, it is interesting to note that given a fixed constraint, though the achieved RF is different in different cases, the RPR almost always has a similar performance. Figure 6.5 visualises this phenomena. For example, given a constraint *R* [5%, 10%], the achieved RFs are 13.48%, 9.98%, 22.68% and 45.33% with respect to $P_{21}^{2.2}$, P_{21}^{3}, P_{21}^{4}, and P_{21}^{5} respectively. However, the achieved corresponding RPRs are 92.22%, 93.84%, 97.46%, and 91.33% respectively, which do not show much difference. Similarly, this situation exists for both *R* [10%, 15%] and *R* [15%, 20%].

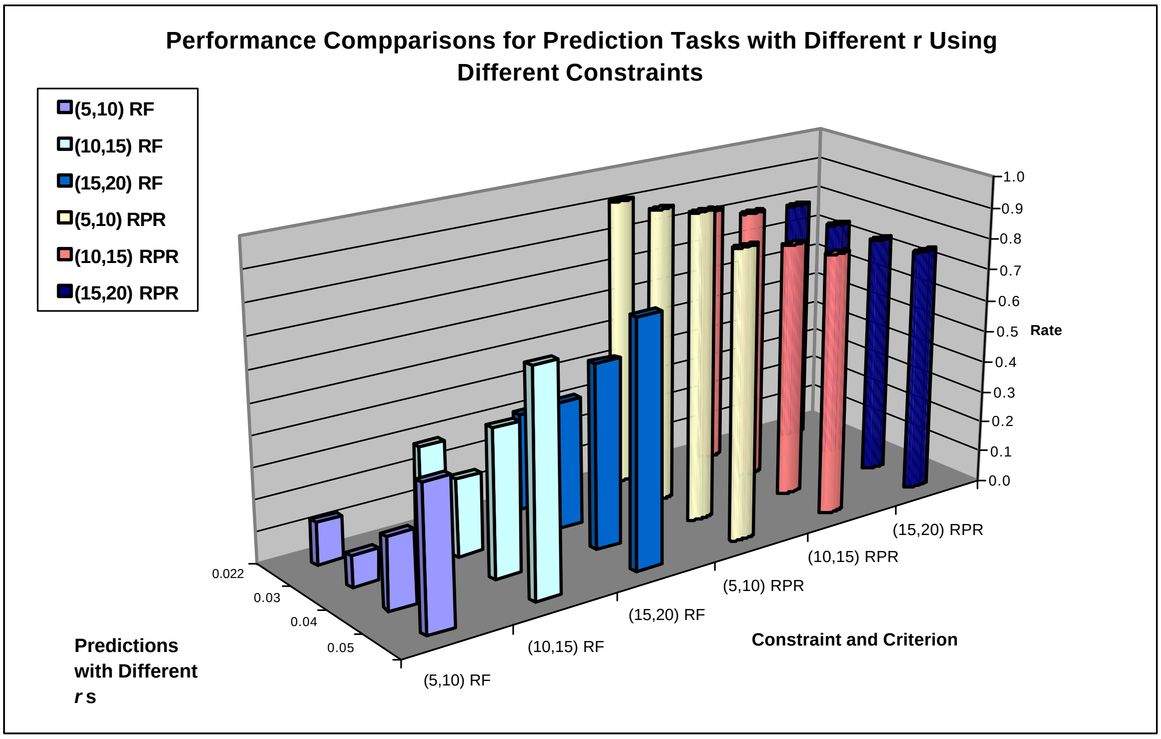


Figure 6.5: Performance comparisons for $P_{21}^{2.2}$, P_{21}^{3}, P_{21}^{4}, and P_{21}^{5} with respect to RF and RPR using different constraints.

4. The number of positive positions generated over the test period may not be affected significantly by the actual number of positive positions (opportunities) held in training data, but mainly affected by the constraint *R* supplied. Figure 6.6 illustrates the fact. Although there are different numbers of opportunities in the four different prediction cases over the test period, the numbers of signals (positive positions predicted) generated by FGP-2 in the different cases are roughly the same as long as the same constraint is applied. (One exception is that the number of positive positions (125) with respect to $P_{21}^{2.2}$ and constraint *R* [15%, 20%] is higher than others).

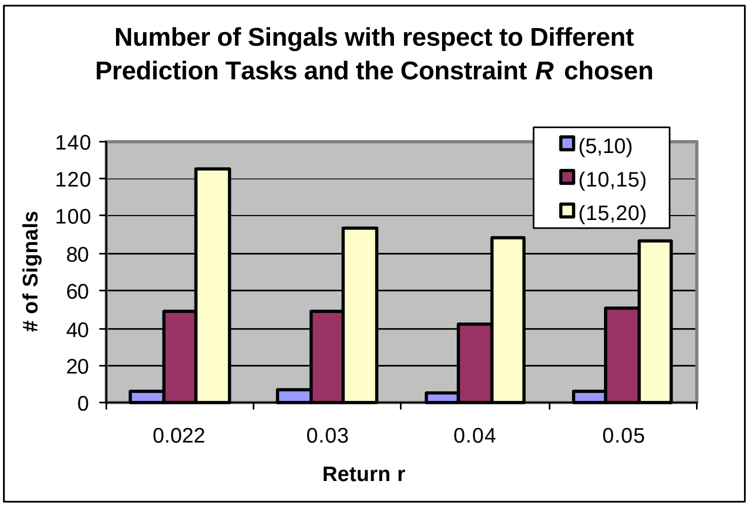


Figure 6.6: The number of signals with respect to different prediction tasks and the constraint *R* chosen.

6.4 Testing on a Market with Down-trend

So far, we have tested FGP-2 on a number of data, including the DJIA index closing prices, individual stock closing prices data such as Microsoft, IBM and McDonald's, etc. However, all the experimental data are gathered from the same kind of resource, i.e. stock markets. Common sense concerning the property of this kind of market is that a general up-trend usually holds from a long historical viewpoint. Question arises as to whether FGP-2 works properly in a market with a general down-trend rather than with a general up-trend.

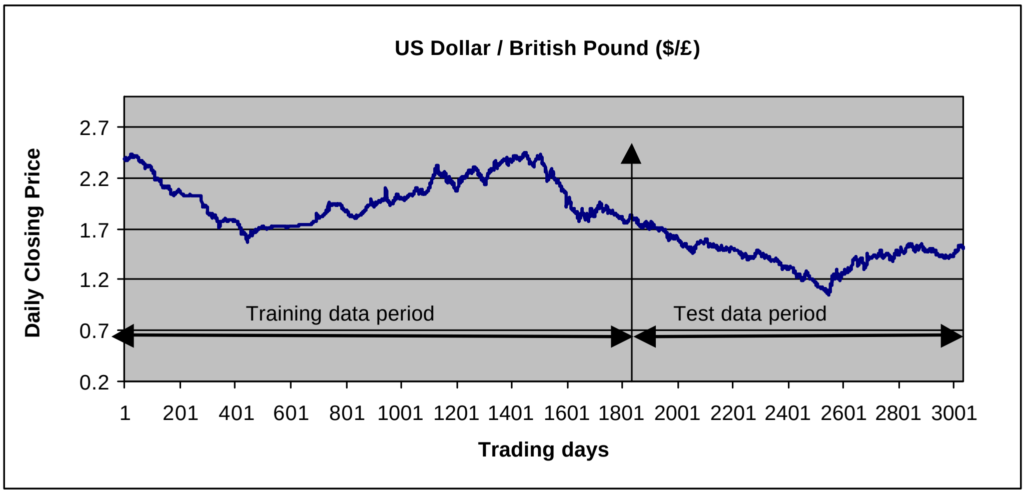


Figure 6.7: The foreign exchange closing prices (total 3035 trading days from 24/01/75 to 11/02/87, including training data: 1900 cases (from 24/01/75 to 16/08/82), and 1135 cases (from 17/08/82 to 11/02/87)).

Unlike many stock prices, prices in foreign exchange markets do not hold the up-trend property in general. Some of them do display the down-trend within a finite long historical period. In order to answer the above question, we purposely choose a foreign exchange data of $US/£BP, which shows a generally down-trend. The data are shown in Figure 6.7 (note that the data may show a up-trend in some sub-periods). We shall use the data to investigate the effectiveness of FGP-2 for achieving the lower RF in a market, in particular, with the property of a generally down-trend.

6.4.1 Experiments

Settings in these experiments are the same as those adopted in the experiments in Section 6.3, including parameters for running GP, 6 indicators input to FGP-2, etc. The prediction tasks set up for investigation are P_{21}^{1}, $P_{21}^{1.5}$, and P_{21}^{2}. They can be categorised into the balanced case, the slightly unbalanced case, and the moderately unbalanced case respectively. As before, we took several mutually exclusive constraints for the constrained fitness function individually for running FGP-2. For each constraint chosen, we ran 10 times and reported only the mean results.

6.4.2 Results and Discussion

Table 6.5 shows the mean results of GDT performances over the test data with respect to P_{21}^{1} (a balanced case). These results are visualised in Figure 6.8. In this foreign exchange market, which shows a generally down-trend, FGP-2 still achieved the similar results as those obtained in the stock markets. The overall picture remains. A tighter constraint leads to a lower RF at the price of a higher RMC without affecting RC much.

Similarly, for $P_{21}^{1.5}$ (a slightly unbalanced case) and P_{21}^{2} (a moderately unbalanced case), we can still observe the overall picture. Both results are reported in Table 6.6 with the graph in Figure 6.9, and in Table 6.7 with the graph in Figure 6.10 respectively. In both unbalanced cases, FGP-2 works properly. This is consistent with what we found in the experiments on the DJIA data. This fact may also partially answer the second question we intend to address in this chapter.

Based on the empirical results here, we may state that the effectiveness of FGP-2 may not be affected significantly by the trend of the market to which it is applied.

R [C_{min},C_{max}]		RF	RMC	RC	AARR	RPR	# of signals
[5, 10]	Mean	0.0998	0.9885	0.5903	1.3922	0.9384	6.8
	STD	0.1925	0.0079	0.0029	0.9132	0.1299	6.0
[10, 15]	Mean	0.2490	0.9237	0.6068	1.3388	0.8676	48.9
	STD	0.1233	0.0131	0.0077	0.3104	0.0588	14.0
[15, 20]	Mean	0.4043	0.8825	0.6015	0.8573	0.7700	93.5
	STD	0.0740	0.0100	0.0110	0.0937	0.0642	16.0
[20,30]	Mean	0.5005	0.8360	0.5856	0.5899	0.7022	155.2
	STD	0.0290	0.0192	0.0082	0.0855	0.0606	27.0
[30,40]	Mean	0.5417	0.7424	0.5641	0.4276	0.6206	267.3
	STD	0.0382	0.0570	0.0256	0.0668	0.0452	77.0

Table 6.5: GDT performances affected by the constraint over the foreign exchange data: \$US/£BP with respect to P_{21}^{1} .

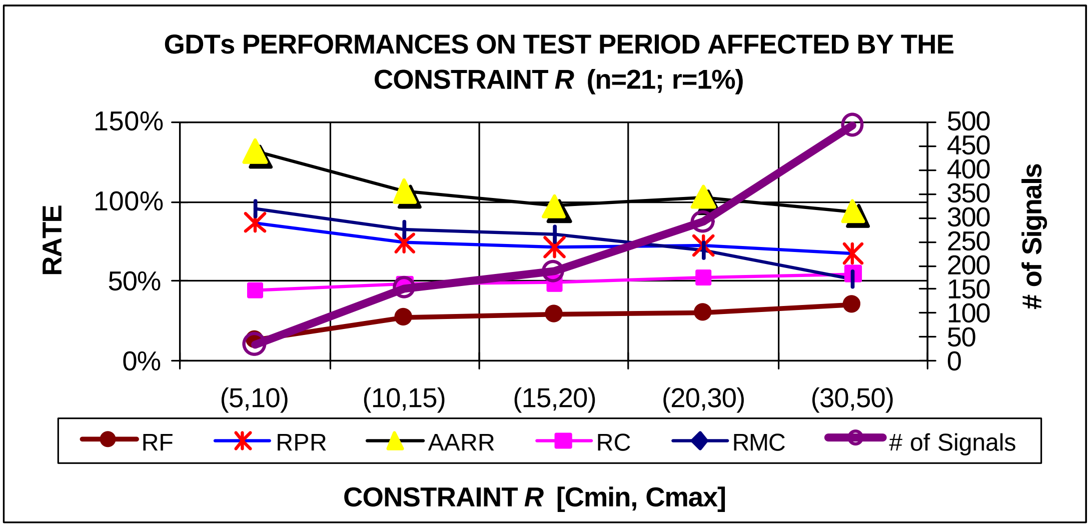


Figure 6.8: GDT performances affected by the constraint over the foreign exchange data: \$US/£BP with respect to P_{21}^{1} .

R [C_{min},C_{max}]		RF	RMC	RC	AARR	RPR	# of signals
[5, 10]	Mean	0.1684	0.9754	0.5630	1.9898	0.8342	15.8
	STD	0.1013	0.0160	0.0045	1.1548	0.0984	8.3
[10, 15]	Mean	0.3028	0.9111	0.5752	0.8349	0.7420	67.0
	STD	0.1034	0.0349	0.0080	0.3606	0.0958	32.1
[15, 20]	Mean	0.3587	0.8307	0.5863	0.8433	0.6852	135.6
	STD	0.0687	0.0487	0.0126	0.1594	0.0707	45.3
[20, 40]	Mean	0.3635	0.7648	0.5982	0.8327	0.6943	188.6
	STD	0.0749	0.0679	0.0258	0.1509	0.0840	59.6

Table 6.6: GDT performances affected by the constraint over the foreign exchange data: $US/£BP with respect to $P_{21}^{1.5}$.

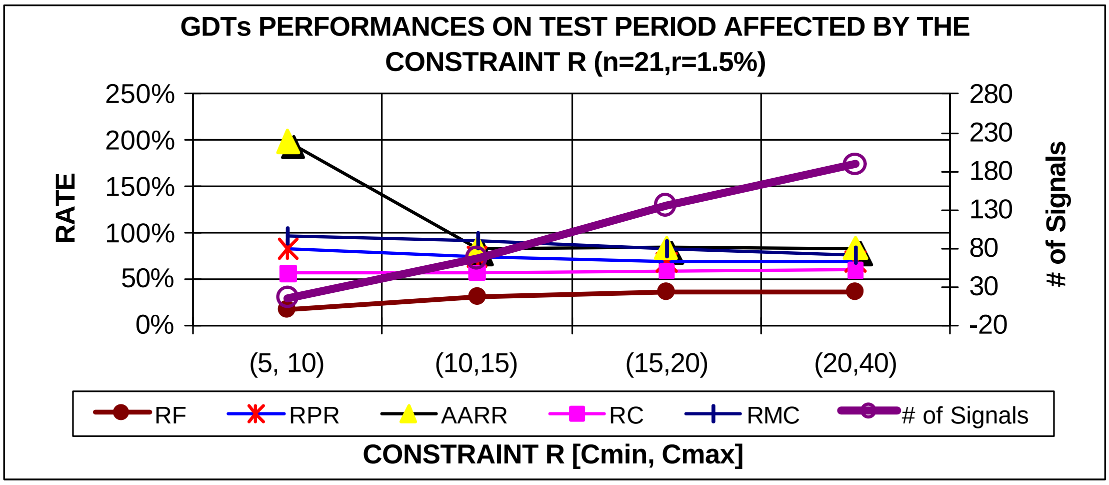


Figure 6.9: GDT performances affected by the constraint over the foreign exchange data: $US/£BP with respect to $P_{21}^{1.5}$.

6.5 Conclusions

In the preceding chapter, we described the procedure of developing a constrained fitness function in FGP-2, and demonstrated its effectiveness for achieving a low rate of RF on the DJIA data and several American stock data. In this chapter, we have further investigated the effectiveness of FGP-2 with the constrained fitness function over a variety of prediction tasks and data sets. Our investigation has been carried out by targeting the prediction over a short period, a number of prediction tasks in unbalanced cases, and predictions over data with the property of a general down-trend respectively.

In terms of the empirical results we have obtained, we draw our conclusions as follows.

R [C_{min},C_{max}]		RF	RMC	RC	AARR	RPR	# of signals
[5, 10]	Mean	0.3190	0.8991	0.6442	0.9177	0.7055	55.7
	STD	0.0582	0.0333	0.0072	0.3263	0.0659	28.4
[10, 15]	Mean	0.3581	0.8387	0.6493	0.7438	0.6874	109.5
	STD	0.0651	0.0534	0.0089	0.1531	0.0611	42.7
[15, 20]	Mean	0.4177	0.7646	0.6448	0.7377	0.6456	177.7
	STD	0.0612	0.0717	0.0138	0.0686	0.0567	65.8
[20, 30]	Mean	0.3962	0.7711	0.6523	0.7091	0.6476	163.6
	STD	0.0652	0.0366	0.0168	0.1086	0.0675	37.2

Table 6.7: GDTs performance affected by the constraint over the foreign exchange data: $US/£BP with respect to P_{21}^2 .

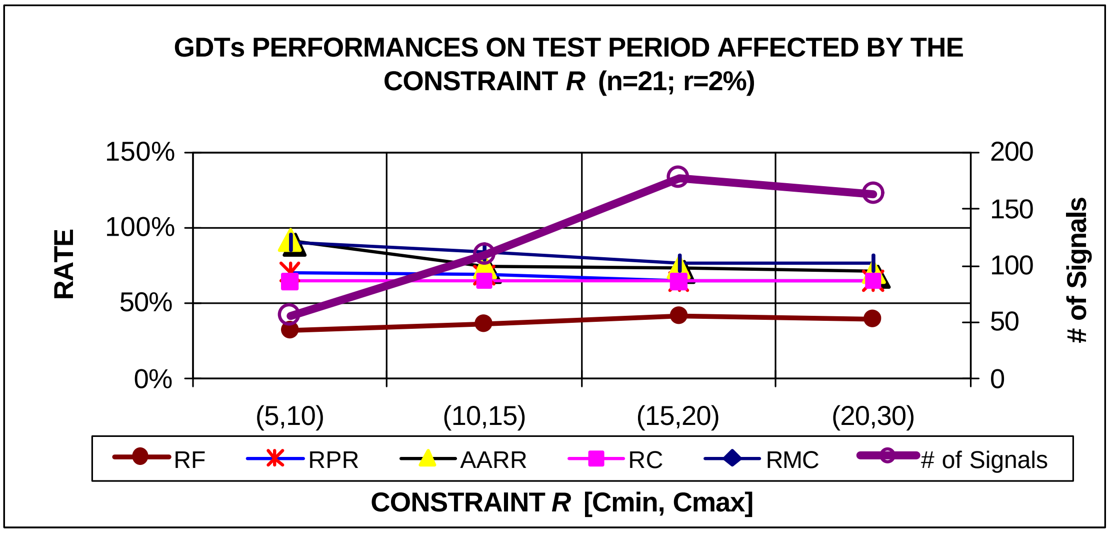


Figure 6.10: GDTs performance affected by the constraint over the foreign exchange data: $US/£BP with respect to P_{21}^2 .

1. FGP-2 works properly over a shorter period in the tests conducted. The overall picture retains over a shorter period. A tighter constraint leads to a lower RF at the cost of a higher RMC, without affecting RC much (see Section 6.2).

2. FGP-2 works well in the unbalanced cases tested. It is capable of generating acceptable lower RFs in slightly and moderately unbalanced cases (see Section 6.3 and Section 6.4). However, in the severely unbalanced case, it fails to achieve acceptable lower RFs (see Section 6.3). Nevertheless, the overall picture remains in any kind of unbalanced cases that we studied here, namely, the lightly, the moderately, or the severely unbalanced cases.

3. The effectiveness of FGP-2 may not be affected by the general trend observed in the data to which it is applied. Like predictions on the data with a general up-trend, the RF achieved by FGP-2 is acceptable in the data with a general down-trend property and the overall picture holds.

Caution should be exercised here, though we have completed substantial experiments on investigation of the effectiveness of FGP-2 with the novel constrained fitness function. First, like any other machine learning tools, FGP-2 is only capable of finding the pattern if it exists. We do not wish to give the false impression that FGP-2 can find the pattern in every data series. In fact, FGP-2 failed to find patterns in certain share prices that we tested on, e.g., HSBC and BT from 1995 to 2000. This may indicate that the business nature of the company have changed, or the behaviour by the investors in these shares have changed over the period tested. Second, we would like to emphasize that FGP-2 is only a tool, not a replacement for human experts. By tuning the constraint in the fitness function, FGP-2 can provide the user with multiple options concerning RF performances. Success of FGP-2 depends on the user’s choice of indicators. Besides, the users are given the responsibility to verify the rules that FGP-2 generates.

Chapter 7

Conclusions

This chapter concludes this thesis, summarises its essential contributions and makes recommendations for future improvements in our current research work.

7.1 Research Summary

The overall research in this thesis is to develop a genetic programming based machine learning tool. We demonstrate the effectiveness of the tool by targeting financial forecasting. Two crucial goals in the tool that we propose are: 1) to improve prediction accuracy and 2) to achieve a low rate of failure.

Since our research focuses on financial forecasting, investigations on predictability of financial markets are necessary to justify that this research is not futile. Based on our review on Efficient Market Hypothesis (EMH), we conclude that financial forecasting is not impossible. Moreover, our review on the study of technical rules in financial literature results in some indicators, which are used as input to our tool. Meanwhile, a literature survey indicates that our research lies in the field of application of GAs in finance. Examples of works in this field are Bauer (1994), Allen & Karjalainen (1995), and Mahfoud & Mani (1996).

Aimed at the first goal of this research, we developed the first version of our program, called FGP-1. FGP-1 is intended to improve prediction accuracy over the given base predictions. In this thesis, FGP-1 is demonstrated to be useful, based on two instances of base predictions. The first is that base predictions consist of ordinal forecasts from experts involved. The second is that base predictions come from non-adaptive technical rules considered in their normal usages.

Based on our experiments, we conclude that FGP is capable of generating more accurate predictions than any of the given expert forecasts, as well as non-adaptive individual technical rules with respect to the prediction tasks that we address.

Aimed at the second goal of this research, we developed the second version of our program, called FGP-2. FGP-2 is intended to achieve a low rate of failure, which is often desirable in financial forecasting. The novelty of FGP-2 lies in the fitness function. A novel constrained fitness function is developed and put into FGP-2. The effectiveness of FGP-2 for achieving a low rate of failure is demonstrated and analysed in a variety of prediction tasks and data sets. We analyse FGP-2, especially its RF performances by tuning the constraint parameter in the fitness function. A tighter constraint leads to a lower RF at the price of a higher RMC without affecting the overall RC much. Result comparisons between FGP-2 and the three NNs and the liner classifier have been carried out over 10 American share data. Based on our review on cost-sensitive learning and data-mining using GAs, we did not find techniques similar to the constrained fitness function for approaching classification problems where misclassification-cost needs to be taken into account.

In summary, in this thesis, we have developed a genetic programming based machine learning tool particularly for attacking financial forecasting problems. Basically, we have achieved the two main research goals by applying FGP-1 and FGP-2 respectively. FGP-1 and FGP-2 are built on top of canonical genetic programming techniques. In particular, we use the grammar-based presentation to generate valid GDTs; we use the hill-climb technique to locally optimise the thresholds in GDTs; furthermore, we develop a constrained fitness function in order to reduce RF. The ideas have been demonstrated to be useful for addressing the financial prediction tasks in this thesis.

7.2 Experiments: Summary and Conclusions

In this thesis, we have applied FGP-1 and FGP-2 to a variety of prediction tasks and data sets. Table 7.1 summarises the experiments we have completed using FGP-1. Table 7.2 summarises the experiments we have completed using FGP-2. In both tables, the claims that each experiment supports are listed (last but one column). The major conclusions are listed as follows.

Prediction task	Type	Data used	Property of the data	Input to FGP-1	Claims related	Section in thesis
Weekly HSI Movement Prediction	Three class classification (Bullish, bearish or sluggish)	Weekly movement of Hang Seng index (103 cases) (25/05/1991-16/10/1993) The whole data set is divided into three mutually exclusive subsets: D1: 34 cases (25/05/1991 - 11/01/1992); D2: 35 cases (18/01/1992 - 5/12/1992); D3: 34 cases (12/12/1992 - 16/10/1993)	Each of these data sets was used as testing data set once, whilst the remaining two sets were employed as the training data set. The mean forecasting accuracy was the overall number of correct forecasts divided by number of cases in the whole data set.	9 expert forecasts	(1.1)	4.2.3.1
S & P 500 P_{63}^{4}	Balanced case; longer period	Training Data (1800:02/04/63-02/07/70) Test Data (900:04/07/70-25/01/74)	(f_{tr}, f_{te}) = (46.56%, 53.78%)	6 direct forecast predicted by 6 technical rules in their normal usages	(1.1)	4.2.3.2
DJIA $P_{21}^{2.2}$	Balanced case; middle period	Training Data (1900:07/04/69-11/10/76) Test Data (900:12/10/76-05/05/80)	(f_{tr}, f_{te}) = (52.47%, 47.11%)	MV_12 (t), MV_50 (t), TRB_5 (t), TRB_50 (t), Filter_5 (t), Filter_63(t)	(1.2); (3)	4.3.3.3
DJIA P_{63}^{4}	Balanced case; longer period	Training Data (1900:07/04/69-11/10/76) Test Data (900:12/10/76-05/05/80)	(f_{tr}, f_{te}) = (52.84%, 49.22%)	MV_12 (t), MV_50 (t), TRB_5 (t), TRB_50 (t), Filter_5 (t), Filter_63(t)	(1.2); (3)	4.3.3.4

Table 7.1: Experiments carried out using FGP-1 and the related claims. (Notes: P_n^r means: to predict whether or not the price will increase a required r% (e.g. 2%) or more within a user-defined period n (e.g., 21 days). As for (f_{tr}, f_{te}), please refer to Equation 4.5 and 4.6, p104).

Prediction task	Type	Data used	Property of the data	*R*s used (C_{min},C_{max})	Claims related	Section in thesis
DJIA $P_{21}^{2.2}$	Balanced case; middle period	Training Data (1900:07/04/69-11/10/76) Test Data (1135:12/10/76-09/04/81) Including: three period below: Down-trend: (378:12/10/76-12/04/78) Side-way-trend: (486: 13/04/78-27/03/80) Up-trend: (261: 28/03/80-09/04/81)	(f_{tr}, f_{te}) = (52.47%, 47.11%)	(5, 10); (10, 15); (15, 20); (20, 35); (35, 50); (50, 65)	(2.1.1)	5.3 and 5.4
DJIA P_{21}^{3}	Slightly unbalance case; middle period		(f_{tr}, f_{te}) = (42.95%, 41.32%)	(5, 10); (10, 15); (15, 20); (20, 30); (30, 40)	(2.1.2)	6.3
DJIA P_{21}^{4}	Moderately unbalanced case; middle period		(f_{tr}, f_{te}) = (30.37%, 26.52%)	(5, 10); (10, 15); (15, 20); (20, 30)	(2.1.2)	6.3
DJIA P_{21}^{5}	Severely unbalanced case; middle period,		(f_{tr}, f_{te}) = (21.58%, 16.12%)	(5, 10); (10, 15); (15, 20)	(2.1.2)	6.3
DJIA $P_{5}^{0.8}$	Balanced case; shorter period		(f_{tr}, f_{te}) = (50.00%, 51.63%)	(5, 10); (10, 15); (15, 20); (20, 35); (35, 50)	(2.2)	6.2
FE P_{21}^{1}	Balanced case; middle period	Foreign Exchange ($/£) Training Data (1900: 24/01/75-17/08/82) Test Data (1135:17/08/82-11/02/87)	(f_{tr}, f_{te}) = (48.42%, 57.44%)	(5, 10); (10, 15); (15, 20); (20, 30); (30, 40)	(2.1.1); (2.3)	6.4
FE $P_{21}^{1.5}$	Slightly unbalance case; middle period		(f_{tr}, f_{te}) = (40.42%, 44.49%)	(5, 10); (10, 15); (15, 20); (20, 40)	(2.1.2); (2.3)	6.4
FE P_{21}^{2}	Moderately unbalanced case; middle period		(f_{tr}, f_{te}) = (32.05%, 37.51%)	(5, 10); (10, 15); (15, 20); (20, 30)	(2.1.2); (2.3)	6.4
10 US-Shares P_{22}^{2}	Mixture of balanced cases and positive unbalanced cases; middle period	10 American Individual Stocks Training Data (Varied) Test Data (100:14/10/96-06/03/97)	*$50<f_{tr}<92$; $50< f_{te}<97$*	(10, 20); (20, 30)	(2.1.2); (4)	5.5

Table 7.2: Experiments carried out using FGP-2 and the related claims. (Notes: P_n^r means: to predict whether or not the price will increase a required r% (e.g. 2%) or more within a user-defined period n (e.g., 21 days). As for (f_{tr}, f_{te}), please refer to Equation 4.5 and 4.6, p104).

1. **FGP-1 can be used to improve prediction accuracy over the given base predictions with respect to the prediction tasks we address.**

1.1 *FGP-1 can be used to combine ordinal individual forecasts and improve forecasting (Weekly HSI Movement Prediction and S&P 500 P_{63}^{4}).*

FGP-1 is fed with ordinal forecasts such as: bullish, bearish or sluggish predictions; buy or not-buy predictions. It can be used to generate more accurate GDTs than each of those input ordinal predictions by combining them (see Chapter 4).

1.2 *FGP-1 can be used to generate more accurate GDTs than each of the six technical rules* *considered in their normal usages with respect to the prediction tasks* P_n^r *(DJIA* $P_{21}^{2.2}$*, DJIA* P_{63}^{4}*).*

FGP-1 is fed with the six indicators as input that are extracted from the six technical rules respectively. FGP-1 can constitute different types of selectors that correspond to types of input indicators. Each selector has the form [*Indicator relation threshold*] where the relation belongs to the set {=, <, >} and the threshold is a real number. The threshold in each selector could possibly be adjusted during evolution. Moreover, FGP-1 looks for the interactive combination structures between those selectors. The way of selector combination is either conjunctive or disjunctive. By doing so, FGP-1 is capable of evolving GDTs that are able to make predictions of higher accuracy than any of the six technical rules considered in their normal usages (see Chapter 4).

2. **FGP-2 allows the user to tune a parameter, i.e. the constraint, in the fitness function in order to reduce RF without affecting the RC significantly, though, at the price of increasing RMC. FGP-2 has achieved consistent results in a variety of data sets and prediction tasks, as explained below** (see Chapter 5 and Chapter 6).

2.1 *FGP-2 on prediction tasks with balanced as well as unbalanced cases.*

2.1.1 In *balanced* cases (see the definition in Section 6.3.1).

In our experiments (DJIA $P_{21}^{2.2}$, $P_{5}^{0.8}$, $/£ P_{21}^{1}), varying the constraint results in varied results as expected. An overall picture emerges, i.e. a tighter constraint results in a lower RF without affecting RC much, though at the cost of a higher RMC.

A further analysis of GDTs' performances is conducted over three sub-periods during the whole test period for the $P_{21}^{2.2}$. The three sub-periods represent three distinct market characteristics, namely, down-trend, side-way-trend, and up-trend. Results show that the overall picture remains over each sub-period. On the other hand, GDTs display an interesting and desirable nature, i.e. they produce a far fewer number of positive positions over the down-trend market situation compared with those over the side-way trend or the up-trend. In other words, FGP-2 has the potential to cope well with varied market situations in the data tested.

2.1.2 In *unbalanced* cases (see the definitions in Section 6.3.1).

In the slightly, moderately and severely unbalanced cases, the overall picture remains. This is verified by experiments of DJIA P_{21}^{3}, DJIA P_{21}^{4} and DJIA P_{21}^{5}; and FE (Foreign Exchange (\$/£))$P_{21}^{1.5}$ and FE P_{21}^{2}. However, in a severely unbalanced case, i.e. DJIA P_{21}^{5}, FGP-2 does not achieve an acceptable low RF as expected. Experimental results of the 10 American shares P_{22}^{2} show that the overall picture also remains in the *positive unbalanced cases* (see the definition in Section 6.3.1, p149).

2.2 *FGP-2 on prediction tasks with long as well as shorter periods.*

Prediction tasks with a shorter period might be more attractive. FGP-2 can deal with the task with a shorter period (e.g. 5 days) well in the data tested. This is demonstrated by the experiments of DJIA $P_{5}^{0.8}$, in which the overall picture holds (see Section 6.2).

2.3 *FGP-2 on data that show general up-trend as well as general down-trend.*

Majorities of our experiments are conducted on stock markets. Data in such markets are generally of up-trend in the long term. To investigate whether the overall picture holds or not on general down-trend markets, a foreign exchange data (\$/£) is chosen to verify this. In both the balanced case (FE P_{21}^{1}) and the unbalanced cases (FE $P_{21}^{1.5}$, FE P_{21}^{2}), empirical results show the overall picture remains on the general down-trend market that we study (see Section 6.4).

3. **FGP-1 outperforms C4.5 with respect to the experiments that we have carried out.**

   C4.5 does not provide mechanisms to approach problems where the misclassification cost is taken into account. Thus, comparisons are only meaningful with FGP-1. Comparisons are completed over prediction tasks with two different periods, namely, 63 and 21 days using the DJIA data (DJIA $P_{21}^{2.2}$, P_{63}^{4}) (see Section 4.3).

4. **FGP-2 favourably compares with the three NNs and beats the linear classifier with respect to the 10 American individual stocks based on** P_{22}^{2} (see Section 5.5).

7.3 Contributions

This thesis contributes to the fields of machine learning, genetic programming and financial forecasting. Two major contributions to the body of knowledge made in this thesis are:

1. We have examined two ways of applying genetic programming to financial forecasting, demonstrated by FGP-1:

   a) If ordinal predictions are given, FGP-1 can potentially combine these predictions to make more accurate predictions;

   b) If indicators are given, FGP-1 can use them to build rules by constructing selectors and searching for thresholds.

   Our experiments support that FGP-1 can combine predictions to make predictions with higher accuracy. It can also generate rules with higher prediction accuracy.

2. A novel constrained fitness function has been proposed. By embedding it into FGP, the resulting algorithm, FGP-2, is capable of achieving a lower rate of failure (RF), without significant effect on the rate of correctness (RC), at the price of a higher rate of missing chances (RMC). This, to a certain extent, allows users to produce GDTs to suit their preferences with regard to RF and RMC.

7.4 Further Research

With respect to FGP system developed in this thesis, the following recommendations may enhance and extend FGP.

1. Understanding the roles of the parameters in the constrained fitness function

The constrained fitness function has been demonstrated to be useful for achieving lower RFs in a variety of data sets and prediction tasks. However, for FGP-2 to work, one must set up appropriate values of the parameters in the fitness function (i.e. three weights: *w_rc*, *w_rmc*, *w_rf* , and the constraint, *R*). Improper settings of these parameters can lead to bad results. It would be worthwhile to research the effects of these parameters on the efficiency of the constrained fitness function.

2. Applicability of FGP to other domains

As discussed in Section 5.7, the idea of putting a constraint into the processes of decision tree generation is potentially be applicable to other algorithms in machine learning. In the context of two-class classification problems, there are two forms of misclassification: false positive and false negative. In many applications, one form of misclassification is more costly than another. The novel constrained fitness function enables the user to reduce one of these two forms of misclassification at the price of the other. We would like to see more use of it in other domains.

Appendix A

The Schema Theorem for Genetic Algorithms

The Schema Theorem of Holland (1975) is based on the concept of *schemas*. A schemas is defined for fixed-length string structures as follows:

> A schema, *H*, is a similarity template describing a subset of strings with similarities at certain string positions (Goldberg 1989).

For a fixed-length binary representation, the alphabet for this language is {0, 1}. Here, to be brief, a schema is any string composed of 0s, 1s and *'s, where each * is interpreted as a "don't care" symbol, which matches either 0 or 1. A schema thereby describes a subset of the potential solutions. For example, the schema 0*00 represents the set of bit strings that includes exactly 0010 and 0110.

Two properties associated with a schema *H* are:

- The Defining Length, $d(H)$ is the number of bits between the index of the first specified position and the index of the last specified position. For example, d(1*****10) = 7 – 1 = 6, while d(1*******) = 1-1 = 0.
- The Schema Order, *o*(*H*) is the number of specified positions (i.e. the number of non-* positions) in H. For example, *o*(1*******) = 1, while *o*(11111111) = 8.

The schema theorem intends to characterize the evolution of the population within a GA in terms of the number of instances representing each schema. Let m(*H*, *t*) denote the number of instances of schema *H* in the population at time *t* (i.e., during the *t*th generation). The schema theorem describes the expected value of m(H, $t+1$) with respect to m(H, t) and other properties of the schema, population, and GA algorithms parameters.

There are three steps in developing the schema theorem. Three steps are selection step, crossover step, and mutation step, each of which has effect on the expected value of m(H, $t+1$) during population evolving in the GA. Considered in the schema theorem are a fitness-proportionate selection strategy, a single-point crossover, and a bit-flipping mutation, which are associated in corresponding steps.

Step 1: Schemata and Fitness-Proportionate Selection

The first step involves the consideration of the effect of that selection has on H from one generation to the next. According to fitness-proportionate selection strategy, the propagation of H will be proportional to the average fitness of the population samples containing H, in relation to the average fitness of the entire population. Thus, given m(H, t), the expected number instances of schema H given in the next generation may be stated as

$$E\,[\mathrm{m}\,(H, t{+}1)] = \mathrm{m}\,(H, t)\,\frac{f(H)}{\overline{f}} \tag{A.1}$$

where, $f(H)$ is the average fitness of the bit strings matching a schema H. $\overline{f}$ is the average fitness of the entire population. This formula states that schemata in the population, with above-average fitness, will receive exponentially increasing representations from generation to the next.

Step 2: Schemata and Single-Point Crossover

The single-point crossover operator is normally applied probabilistically to the population of binary strings. Note that crossover disrupts a schema only when the crossover point occurs within the defining length of the schema. The probability that a schema H survives the disruption of crossover in a string of length of l, denoted with $P_s\,(c)$, is given as follows.

$$P_s\,(c) \geq 1 - P_c\,\frac{d(H)}{l-1} \tag{A 2}$$

where P_c is the probability crossover, and the inequality reflects that fact that crossover may not actually disrupt the schema even when the crossover point is within the defining length.

Step 3: Schemata and Bit-Flipping Mutation

Mutation normally defined as a low-probability operator that randomly flips the bit value for any position in a population. The probability that a schema H survives disruption due to mutation, denoted by $P_s\,(m)$ is then given as follows.

$$P_s\,(m) = (1 - P_m)^{\,o(H)} \qquad \text{(A 3)}$$

Note that the schema theorem only considers the possible negative influence of crossover and mutation operation, i.e. crossover or mutation disrupts H for the next generation, without considering their (presumably) positive effects, i.e. crossover or mutation constructs possible one or two new *Hs* for the next generation. Thus, the full schema theorem for the genetic algorithms with fitness-proportionate selection, single-point crossover and bit-flipping mutation, provides a lower bound on the expected frequency of schema H at the next generation, as follows.

$$E\,[\mathrm{m}\,(H, t+1)] \geq \mathrm{m}\,(H, t)\,\frac{f(H)}{\overline{f}}\,(1 - P_c\,\frac{d(H)}{l-1})\,(1 - P_m)^{\,o(H)} \qquad \text{(A 4)}$$

The Schema Theorem can be interpreted by stating that “short, low-order, above-average schemata receive exponentially increasing trails in subsequent generations” (Holland 1975).

Bibliography

[Alander, 1994] Alander, J. T. (1994). An indexed bibliography of genetic algorithms: Years 1957-1993. Art of CAD Ltd., Vaasa (Finland) (over 3000 GA references).

[Alander, 1995] Allander, J.T. (1995). Indexed bibliography of genetic algorithms papers of 1996. University of Vaasa, Department of Information Technology and Production Economics, Rep.94-1-96.

[Alexander, 1961] Alexander, S. S. (1961). Price movements in speculative markets: Trends or random walks. *Industrial Management Review*, 2 (2). 7-26.

[Alexander, 1964] Alexander, S. S. (1964). Price movement in speculative markets: Trend or random walks, No. 2. In Cootner, P. (ed.). *The Random Character of Stock Market Prices*, MIT Press, Cambridge, MA, 338-372.

[Allen & Karjalainen 1995] Allen, F. & Karjalainen, R. (1995). Using genetic algorithms to find technical trading rules. Working paper at Rodney L. White Center for Financial Research, The Wharton School, University of Pennsylvania. 20-95.

[Allen & Karjalainen, 1999] Allen, F. & Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, Vol. 51, Issue 2, February. 245-271.

[Altenerg, 1995] Altenerg, L. (1995). The schema theorem and price's theorem. In Whitley, L.D. & Vose, M.D. (eds.*). Foundations of Genetic Algorithms 3,* Morgan Kaufmann. 23-49.

[Anderson et al., 1992] Anderson, B.L., McDonnell, J.R. & Page, W.C. (1992). Configuration optimisation of mobile manipulators with equality constraints using evolutionary programming. *Proceedings of First Annual Conference on Evolutionary Programming*. San Diego, CA: Evolutionary Programming Society. 71-79.

[Angeline et al., 1999] Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X. & Zalzala, A. (eds.). (1999). *Proceedings of the Congress on Evolutionary Computation,* (CEC'99). Washington D.C., USA. IEEE Press.

[Angeline & Pollack, 1992] Angeline, P.J. & Pollack, J.B. (1992). The evolutionary induction of subroutines. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*. Bloomington, Indiana, USA. Lawrence Erlbaum Associates.

[Angeline & Kinnear, Jr., 1996] Angeline, P.J. & Kinnear, Jr., K. E. (eds.), (1996). *Advances in genetic programming II*. MIT Press.

[Arifovic, 1994] Arifovic, J. (1994). Genetic algorithms learning and the cobweb model. *Journal of Economic Dynamics and Control*, 18(1), 3-28.

[Arifovic, 1996] Arifovic, J. (1996). The behaviour of the exchange rate in the genetic algorithms and experimental economics. *Journal of Political Economy*, 104, 510-541.

[Arthur, 1992] Arthur, B. (1992). On learning and adaptation in the economy. Santa Fe Institute Working Paper 92-07-38.

[Asch et al. 1984] Asch, P., Malkiel, B. G. & Quandt, R. E. (1984). Market efficiency in racetrack betting. *Journal of Business*, Vol. 57, No. 2, 165-175.

[Bäck, 1996] Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. New York: Oxford University Press.

[Bäck, 1997] Bäck, T. (ed.), (1997). *Proceedings of the seventh international conference on genetic algorithms*. San Francisco, California: Morgan Kaufmann Publishers, Inc., 1997.

[Bäck et al., 1997] Bäck, T., Hammel, U. & Schwefel, H.P. (1997). Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*. Vol. 1 No. 1. 3-17.

[Backus, 1959] Backus, J.W. (1959). The syntax and semantics of the proposed international algebraic language of Zurich. ACM-GAMM conference, ICIP, Paris, June.

[Bachelier, 1900] Bachelier, L. (1900). *Theory of speculation in the random character of stock market prices*. MIT, Cambridge, MA, 1964; Reprint.

[Banzhaf et al., 1998] Banzhaf, W., Nordin, P., Keller, R.E. Francone, F.D. (1998). *Genetic Programming: An introduction on the automatic evolution of computer programs and its applications*. San Francisco, California: Morgan Kaufmann.

[Banzhaf et al., 1999] Banzhaf, W, Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M. & Smith, R.E. (eds). (1999). *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO'99). Orlando, Florida, USA, 13-17 July. Morgan Kaufmann.

[Bauer, 1994] Bauer, R. J. Jr. (1994). *Genetic algorithms and investment strategies*. New York, John Wiley & Sons, Inc.

[Beyer, 1997] Beyer, H.-G. (1997). An alternative explanation for the manner in which genetic algorithms operate. *BioSystems*. 41: 1-15.

[Blume et al., 1994] Blume, L., Easley, D. & O'Hara, M. (1994). Market statistics and technical analysis: the role of volume, *Journal of finance*, 49, 153-181.

[Bojarczuk et al., 1999] Bojarczuk, C.C., Lopes, H.S. & Freitas, A.A. (1999). Discovering comprehensible classification rules by using genetic programming: a case study in a medical domain. In Banzhaf, W, Daida, J., Eiben, A.E. Garzon, M.H., Honavar, V., Jakiela, M. & Smith, R.E. (eds.). *Proceedings of the Genetic and Evolutionary Computation Conference*, Vol. 2, Orlando, Florida, USA, July. Morgan Kaufmann. 953-958,

[Bollerslev, 1986] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31, 307-327.

[Box & Jenkins, 1976] Box, G.E.P. & Jenkins, F.M. (1976). *Time series analysis: Forecasting and control*, 2nd ed. Oakland, CA: Holden-Day.

[Bradford et al., 1998] Bradford, J., Kunz, C., Kohavi, R., Brunk, C. & Brodley, C. (1998). Pruning decision trees with misclassification costs. In *Proc. of the 1998 European Conference on Machine Learning*. 131-136.

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth, Pacific Grove, CA.

[Breiman, 1994] Breiman, L., (1994). Bagging predictions, Technical Report 421, Dept. of Statistics Technical Report 421, University of Claifornia, Berkeley, Claifornia.

[Brock et al., 1992] Brock, W., Lakonishok, J. & LeBaron, B. (1992). Simple technical trading rules and the stochastic properties of stock returns. *Journal of Finance*, 47, 1731-1764.

[Brooker et al., 1989] Brooker, L.B., Goldberg, D.E. & Holland, J.H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence* 40: 235-282.

[Brown & Jennings, 1989] Brown, D., and Jennings, R. (1989). On technical analysis, *Review of Financial Studies*, 2, 527-552.

[Butler, 1997] Butler, J.M. (1997). Eddie beats the market, data mining and decision support through genetic programming. *Developments*, Reuters Limited, (1997), Vol.1.

[Campbell et al., 1997] Campbell, J.Y., Lo, A.W. & MacKinlay, A.C. (1997). *The econometrics of financial markets*, Princeton, N.J.: Princeton University Press.

[Chan & Stolfo, 1998] Chan, P.K. & Stolfo, S.J. (1998). Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. *Proc. 4th International Conference on Knowledge Discovery and Data Mining*. New York, NY, 164-168.

[Chen, 1997] Chen, S.-H. (1998). Evolutionary computation in financial engineering: A road map of GAs and GP. *Financial Engineering News*, Vol. 2, No. 4.

[Chen et al., 1998] Chen, S.-H., Yeh, C.-H. & Lee, W.-C. (1998). Option pricing with genetic programming. In Koza, J.R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H. & Riolo, R. (eds.), *Proceedings of the Third Annual Genetic Programming Conference*, CA: Morgan Kaufmann Publishers. 32-37.

[Chen & Duffy, 1996] Chen, S-H. & Duffy, J. (1996). Genetic programming in the coordination game with a chaotic best-response function. In Fogel, L., Angeline, P. & Bäck, T. (eds.). *Evolutionary Programming V*, MIT Press. 277-286.

[Chen & Lin, 1998] Chen, S.-H. & Lin, W.-Y. (1998). The appeal of evolution: The case of the RGA-based portfolios," in Debnath, N.C. (ed.). *Proceedings of the ISCA 13th International Conference*. 125-130.

[Chen & Lu, 1999] Chen, S.-H. & Lu, C.-F. (1999). Would evolutionary computation help in designs of ANNs in forecasting foreign exchange rates? In Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X. & Zalzala, A. (eds.). *Proceedings of the Congress on Evolutionary Computation*, (CEC'99). Washington D.C., USA. IEEE Press. 267-274.

[Chen & Yeh, 1996a] Chen, S-H. & Yeh, C-H. (1996a). Genetic programming learning and the cobweb model. In Angeline, P. & Kinnear, K.E. (eds.). *Advances in genetic programming 2*, MIT Press, Cambridge, MA. Chapter 22. 443-466.

[Chen & Yeh, 1996b] Chen, S.-H. & Yeh, C.-H. (1996b). Genetic programming and the efficient market hypothesis. In Koza, J.R., Goldberg, D., Fogel, D. & Riolo, R. (eds.). *Procedings of the First Annual Conference on Genetic Programming.* Stanford University, CA, USA, 28-31 July. MIT Press. 45-53.

[Chen & Yeh, 1996c] Chen, S.-H. & Yeh, C.-H. (1996c). Toward a computable approach to the efficient market hypothesis: An application of genetic programming. *Journal of Economic Dynamics and Control*, 21, 1043-1063.

[Chen & Lee, 1997] Chen, S.-H. & Lee, W.-C. (1997). Option pricing with genetic algorithms: the case of european options," in Back, T (ed.), *Proceedings of 1997 International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Francisco. 704-711.

[Chen & Yeh, 1997a] Chen, S.-H. & Yeh, C.-H. (1997). Using genetic programming to model volatility in financial time series. In Koza, J.R., Goldberg, D., Fogel, D. & Riolo, R.

(eds.). *Procedings of the Second Annual Conference on Genetic Programming*. Stanford University, CA, USA, 28-31 July. MIT Press. 58-63.

[Chen & Yeh, 1997b] Chen, S.-H. & Yeh, C.-H. (1998). Genetic programming in the overlapping generations model: An illustration with dynamics of the inflation rate. In Porto, V.W., Saravanan, N. Waagen, D. & Eiben, A.E. (eds.). *Evolutionary Programming VII*, Lecture Notes in Computer Science. 829-838.

[Chen & Kuo, 1999] Chen, S.-H. & Kuo, T.-W. (1999). Towards an agent-based foundation of financial econometrics: An approach based on genetic programming artificial markets. In *Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco, CA: Morgan Kaufmann. 966-973.

[Chan & Stolfo, 1996] Chan, P.K & Stolfo, S. (1996). Scaling learning by meta-learning over disjoint and partially replicated data. *Proc. of Ninth Florida Artificial Intelligence Research Society*. 151-155.

[Chan & Stolfo, 1998] Chan, P.K. & Stolfo, S.J. (1998). Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. *Proc. 4th International Conference on Knowledge Discovery and Data Mining*, New York, NY, 164-168.

[Colin, 1994] Colin, A. (1994). Genetic algorithms for financial modelling, In Deboeck, G. (eds.). *Trading on the edge: Neural, genetic and fuzzy systems for chaotic financial markets*. John Wiley & Sons, NY. 148-173.

[Cootner, 1962] Cootner, P. (1962). Stock prices: random vs systematic changes. *Industrial Management Review,* 3 (2), 24-45.

[Cootner, 1964] Cootner, P. (ed.). (1964). *The Random Character of Stock Market Prices*. MIT Press, Cambridge, MA.

[Cowles, 1933] Cowles, A. (1933). Can stock market forecasters forecast? *Econometrica*, 1, 309-324.

[Chomsky, 1956] Chomsky, N. (1956). Three models for the description of language. *IEEE Transactions on Information Theory*, 2(3): 113-124.

[Daida et al., 1996] Daida, J.M., Bersano-Begey, T.F., Ross, S.J. & Vesecky, J.F. (1996). Computer-assisted design of image classification algorithms: Dynamic and static fitness evaluations in a scaffolded genetic programming environment. In Koza, J.R., Goldberg, D., Fogel, D. & Riolo, R. (eds.). *Procedings of the First Annual Conference on Genetic Programming*. Stanford University, CA, USA, July. MIT Press. 279-284.

[Darwin, 1859] Darwin, C. (1859). *On the origin of species by means of natural selection*. John Murray.

[Davis, 1991] Davis, L. (ed.). (1991). *Handbook of genetic algorithms*. Van Nostrand Reinhold.

[Davis, 1994] Davis, L. (1994). Genetic algorithms and financial applications. In Deboeck, G. (eds.) *Trading on the edge: Neural, genetic and fuzzy systems for chaotic financial markets*. John Wiley & Sons, NY.

[Domingos, 1999] Domingos, P. (1999). MetaCost: A general method for making classifiers cost-sensitive. In *Proc. of the Fifth International Conference on Knowledge Discovery and Data Mining* (KDD-99). San Diego, CA: ACM press.

[Donaldson & Kamstra, 1996] Donaldson, R.G. & Kamstra, M. (1996). Using artificial neural networks to combine financial forecasts. *Journal of Forecasting,* Vol. 15, 49-61.

[Donchian, 1957] Donchian, R.D. (1957). Trends following methods in commodity analysis. *Commodity Year Book.*

[Dorsey & Mayer, 1995] Dorsey, R.E. & Mayer, W.J. (1995). Genetic algorithms for estimation problems with multiple optima, non-differentiability, and other irregular features. *Journal of Business and Economics Statistics,* Vol.13, No. 1, 53-66.

[Deboeck, 1994] Deboeck, G.J. (eds), (1994). *Trading on the edge: neural, genetic, and fuzzy systems for chaotic financial markets.* New York: Wiley.

[DeJong 1975] DeJong, K.A. (1975). *An analysis of the behaviour of a class of generic adaptive systems*. Ph.D dissertation. University of Michigan.

[DeJong et al., 1993] DeJong, K.A., Spears, W.M. & Gordon, D.F. (1993). Using genetic algorithms for concept learning. *Machine Learning*, 13, 161-188.

[de la Maza, 1989] de la Maza, M. (1989). A SEAGUL visits the race track. *Proceedings of the 3rd International Conference on Genetic Algorithms*. Morgan Kaufman, 208-212.

[de la Maza & Yuret, 1995] de la Maza, M. & Yuret, D. (1995). A model of stock market participants. In Biethahn, J. & Nissen, V. (eds). *Evolutionary Algorithms in Management applications,* Springer Verlag, Heidelberg. 290-304.

[Dworman et al., 1996] Dworman, G., Kimbrough, S.O. and Laing, J.D. (1996). Bargaining by artificial agents in two coalition games: A study in genetic programming for electronic commerce. In Koza, J.R., Goldberg, D., Fogel, D. & Riolo, R. (eds.). *Procedings of the First Annual Conference on Genetic Programming.* Stanford University, CA, USA, July. MIT Press. 54-62.

[Economist, 1993] The Economist (1993). Frontiers of finance. 328 (7832). October 9th, a survey.

[Edmonds & Kershaw, 1994] Edmonds, A.N. & Kershaw, P.S. (1994). Genetic programming of Fuzzy logic production rules with application to financial trading. *Proceedings of the IEEE World Conference on Computational Intelligence,* Orlando, Florida.

[Edwards & Magee, 1992] Edwards, R.D. & Magee, J. (1992). *Technical analysis of stock trends*. New York: New York Institute of Finance.

[Engle, 1982] Engle, R.F. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of U.K. inflation. *Econometrica,* 50, 987-1008.

[Evett & Fernandez, 1998] Evett, M. & Fernandez, T. (1998). Numeric mutation improves the discovery of numeric constants in genetic programming. In Koza, J.R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H. & Riolo, R. (eds). (1998). *Proceedings of the Third Annual Conference on Genetic Programming*. University of Wisconsin, Madison, Wisconsin, SA, 22-25 July. Morgan Kaufmann. 66-71.

[Fama, 1965] Fama, E.F. (1965). The behaviour of stock prices. *Journal of Business* 38, 34-105.

[Fama & Blume, 1966] Fama, E.F. & Blume, M.E. (1966). Filter rules and stock-market trading. *Journal of Business*, Vol. 39, 226-241.

[Fama, 1970] Fama, E.F. (1970). Efficient capital markets: A review of theory and empirical work *Journal of Finance* 23, 383-417.

[Fama, 1991] Fama, E.F. (1991). Efficient capital markets: II. *Journal of Finance* 46(5), 1575-1617.

[Fama & French, 1988] Fama, E.F. & French, K.R. (1988). Permanent and temporary components of stock prices. *Journal of Political Economy*. 246-273.

[Fama & Blume, 1966] Fama, E.F. & Blume, M.E. (1966). Filter rules and stock-market trading. *Journal of Business* 39 (1), 226-241.

[Fan et al., 1996] Fan, D.K., Lau, K-N. & Leung, P-L. (1996). Combining ordinal forecasting with an application in a financial market. *Journal of Forecasting*, Vol. 15, No.1, Wiley, January, 37-48.

[Fan et al., 1999] Fan, W., Stolfo, S. J., Zhang, J. & Chan, P.K. (1999). AdaCost: Misclassification Cost-sensitive Boosting. In *Proc. of the 1999 International Conference on Machine Learning* (ICML99).

[Fawcett & Provost, 1997] Fawcett, T., & Provost, F.J. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1 (3).

[Fogel et al., 1966] Fogel, L.J., Owens, A. & Walsh, M. (1966). *Artificial intelligence through simulated evolution*. New York: John Wiley & Sons.

[Fogler, 1993] Fogler, H.R. (1993). A modern theory of security analysis. *Journal of Portfolio Management*. Spring, 6-15.

[Fogler, 1995] Fogler, H.R. (1995). Investment analysis and new quantitative tools. *Journal of Portfolio Management*. Summer, 39-48.

[Fogel & Ghozeil, 1998] Fogel, D.B. & Ghozeil, A. (1998). The schema theorem and the misallocation of trials in the presence of stochastic effects. In Porto, V.W. Saravanan, N., Waagen, D.E. & Eiben, A.E. (eds.). *Evolutionary Programming VII: Proceedings of the 7th Annual Conference on Evolutionary Programming*, San Diego, CA. Springer-Verlag. 313-321.

[Foster, 1986] Foster, G. (1986). *Financial statement analysis*. Second Edition, Prentice-Hall.

[Frankel & Froot, 1990] Frankel, J.A. & Froot, K.A. (1990). Chartists, fundamentalists, and trading in the foreign exchange market. *American Economic Review*, 80(2), 181-185.

[Freund & Schapire, 1996] Freund, Y. & Schapire, R.E., (1996). Experiments with a new boosting algorithm, In *Proc. of the Thirteenth International Conference on Machine Learning,* Morgan Kaufmann, 148-156.

[Frey, 1991] Frey, P.W. & Slate, D.J. (1991). Letter recognition using Holland-style adaptive classifiers, *Machine Learning*, 6, 161-182.

[Gencay, 1996] Gencay, R. (1996). Non-linear prediction of security returns with moving average rules. *Journal of Forecasting*, 15, 165-174.

[Gibbons & Hess, 1981] Gibbons, M.R. & Hess, P. (1981). Day of the week effects and asset returns. *Journal of Business*. 579-590.

[Goldberg, 1989] Goldberg, D.E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

[Goldberg & Schulmeister, 1988] Goldberg, M. & Schulmeister, S. (1988). Technical analysis and stock market efficiency. Economic research reports #88-21. C.V. Starr Center for Applied Economics. Department of Economics, New York University.

[Goonatilake & Treleaven, 1995] Goonatilake, S. & Treleaven, P. (eds.), (1995). *Intelligent systems for finance and business*. Wiley, New York.

[Granger, 1972] Granger, C.W.J. (1972). Empirical studies of capital markets: A survey. In

Szego, G.P. and Shell, K. (eds.), *Mathematical Models in Investment and Finance.* Amsterdam:North-Holland. 469-519.

[Granger, 1992] Granger, C.W.J. (1992). Forecasting, in Newman, P., Milgate, M. & Eatwell, J. (eds.). *New palgrave dictionary of money and finance,* Macmillan, London, 142-143.

[Grefenstette, 1992] Grefenstette, J.J. (1992). Deception considered harmful. In *FOGA-92, Foundations of Genetic Algorithms,* (Vail, Colorado), 24-29 July.

[Grefenstette & Baker, 1989] Grefenstette, J.J. & Baker, J.E. (1989). How genetic algorithms work: a critical look at implicit parallelism. In Schaffer, J.D. (ed). *Proceedings of the Third International Conference on Genetic Algorithms*. George Mason University. Morgan Kaufmann. 20-27.

[Grossman & Stiglitz, 1980] Grossman, S. & Stiglitz, J. (1980). On the impossibility of informationally efficient markets. *American Economic Review* 70, 393-408.

[Gritz & Hahn, 1997] Gritz, L. & Hahn, J.K. (1997). Genetic programming evolution of controllers for 3-D character animation. In Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B. Garzon, M., Iba, H & Riolo, R.L. (eds.) *Genetic Programming 1997: Proceedings of the Second Annual Conference.* Stanford University, CA, USA, July. Morgan Kaufmann. 139-146.

[Gruau, 1996] Gruau, F. (1996). On using syntactic constraints with genetic programming. In Angeline, J. & Kinnear, Jr., K.E. (eds.). *Advances in Genetic Programming II,* MIT Press, Camridge, MA, 377-394.

[Gujarati, 1995] Gujarati, D.N. (1995). *Basic econometrics,* 3rd ed. New York: McGraw-Hill.

[Handley, 1994] Handley, S.G. (1994). The automatic generations of plans for a mobile robot via genetic programming with automatically defined functions. *Advances in Genetic Programming*. Kinnear, Jr., K.E.. (ed.). MIT Press, Cambridge, MA, 391-401.

[Harrald & Kamstra, 1997] Harrald, P. G. & Kamstra, M. (1997). Evolving Artificial Neural Networks to Combine Financial Forecasts. *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, 40-52.

[Harvey, 1993] Harvey, A.C. (1993). *Time series models*. Second Edition. Harvester Wheatsheaf.

[Haynes et al., 1995] Haynes, T.D., Wainwright, R., Sen, S. & Schoenefeld, D. (1995). Strongly typed genetic programming in evolving cooperation strategies. In Eshelman, L. (ed.). *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann. 271-278.

[Hausch & Ziemba, 1985] Hausch, D.B. & Ziemba, W.T. (1985). Transactions costs, extent of inefficiencies, entries and multiple wagers in a racetrack betting model. *Management Science*, Vol. 31, No. 4, 381-394.

[Holland, 1962] Holland, J.H. (1962). Outline for a logical theory of adaptive systems. *Journal of Association of Computer Machine*. Vol. 3, 297-314.

[Holland, 1975] Holland, J.H. (1975). *Adaptation in natural and artificial system,* University of Michigan Press.

[Holland, 1986] Holland, J.H. (1986). Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems. In Michalske, R., Carbonell, J. & Mitchell, T. (eds). *Machine learning: An artificial intelligence approach*, Vol. 2. San Mateo, Calif.: Morgan Kaufmann.

[Holland, 1992] Holland, J.H. (1992). *Adaptation in natural and artificial system*. Cambridge , Mass.: MIT Press.

[Hooker, 1995] Hooker, J.N. (1995). Testing heuristics: we have it all wrong. *Journal of Heuristics*, Vol.1, No.1, 33-42.

[Howley, 1996] Howley, B. (1996). Genetic programming of near-minimum-time spacecraft attitude manoeuvres. In Koza, J.R., Goldberg, D., Fogel, D. & Riolo, R. (eds.). *Procedings of the First Annual Conference on Genetic Programming*. Stanford University, CA, USA, July. MIT Press. 98-106.

[Hudson et al., 1996] Hudson, R., Dempsey, M. & Keasey, K. (1996). A note on the weak form efficiency of capital markets: The application of simple technical trading rules to UK stock prices- 1935 to 1994. *Journal of Banking and Finance*, 20, 1121-1132.

[Izumi & Okatsu, 1996] Izumi, K. & Okatsu, T. (1996). An artificial market analysis of exchange rate dynamics. In Fogel, L., Angeline, P. & Bäck, T. (eds.). *Evolutionary Programming V*, MIT Press. 27-36.

[Janikow, 1993] Janilow, C. Z. (1993). a knowledge-intensive genetic algorithms for supervised learning. *Machine Learning*, 13, 189-228.

[Jegadeesh, 1990] Jegadeesh, N. (1990). Evidence of predictable behaviour of security returns. *Journal of Finance*, 45, No 3, 881-898.

[Jegadeesh & Titman, 1993] Jegadeesh, N. & Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *Journal of Finance*, 48, No.1, 65-91.

[Karjalainen, 1994] Karjalainen, R. (1994). *Using genetic algorithms to find technical trading rules in financial markets*. Ph.D. Thesis, University of Pennsylvania.

[Keim, 1983] Keim, D.B. (1983). Size-related anomalies and stock return seasonality: Further empirical evidence. *Journal of Financial Economics*, 13-32.

[Kendall, 1953] Kendall, M.G. (1953). The analysis of economic time series-Part I: Prices. *Journal of the Royal Statistical Society*, Series A: 11-25.

[Kho, 1996] Kho, B.C. (1996). Time-varying risk premia, volatility and technical trading rule profits: Evidence from foreign currency future markets. *Journal of Financial Economics*, 41:249-290.

[Kinnear, Jr., 1994] Kinnear, Jr. K.E. (eds.), (1994). *Advances in genetic programming*, MIT Press.

[Kinnear, Jr., 1994a] Kinnear, Jr., K.E. (1994a). Alternatives in automatic function definition: A comparison of performance In *Advances in Genetic Programming*, Kinnear, Jr., K.E. (ed.). MIT Press, Cambridge, MA, 119-141.

[Kitano, 1990] Kitano, H. (1990). Designing neural network using genetic algorithms with graph generation system. *Complex Systems*, (4). 461-476.

[Koza, 1992] Koza, J.R., (1992). *Genetic Programming: on the programming of computers by means of natural selection*. MIT Press.

[Koza, 1994] Koza, J.R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press.

[Koza, 1994a] Koza, J.R. (1994a). Scalable learning in genetic programming using automatic function definition. *Advances in Genetic Programming*, Kinnear, Jr., K.E. (ed.). MIT

Press, Cambridge, MA, 99-117.

[Koza, 1995] Koza, J.R. (1995). Genetic programming for economic modelling. In Goonatilake, S. & Treleaven, P. (eds.). *Intelligent systems for finance and business*. Wiley, New York. 250-270.

[Koza, 1995a] Koza, J.R. (1995a). Genetic programming for economic modelling. In Goonatilake, S. & Treleaven, P. (eds.), (1995). *Intelligent systems for finance and business*. Wiley, New York. 250-270.

[Koza, 1995b] Koza, J.R. (1995b). Evolving the architecture of a multi-part program in genetic programming using architecture-altering operations. In McDonnell, J.R. Reynolds, R.G. & Fogel, D.B. (eds.). *Evolutionary Programming IV: Proceedings of the Fourth Annual* Conference on Evolutionary Programming. San Diego, CA, 1995. MIT Press. 695-717.

[Koza, 1995c] Koza, J.R. (1995c). Two ways of discovering the size and shape of a computer program to solve a problem. In Eshelman, L.J. (ed.). *Proceedings of the Sixth International conference on Genetic Algorithms*. Morgan Kaufmann Publishers, Inc. San Francisco, California. 287-294.

[Koza & Andre, 1996] Koza, J.R. & Andre, D. (1996). Classifying protein segments as transmembrane domains using architecture-altering operations in genetic programming. In Angeline, P.J. and Kinnear, K.E. (eds.). *Advances in Genetic Programming 2*, chapter 8, pages 155-176. MIT Press, Cambridge, MA, USA.

[Koza et al., 1996] Koza, J.R., Goldberg, D., Fogel, D. & Riolo, R. (eds.). (1996). *Proceedings of the First Annual Conference on Genetic Programming*. Stanford University, CA, USA, July. MIT Press.

[Koza et al., 1997] Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A. & Dunlap, F. (1997). Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation*, 1(2): 109-128.

[Koza et al., 1997] Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M., Iba, H. & Riolo, R.L. (eds). (1997). *Proceedings of the Second Annual Conference on Genetic Programming*. Stanford University, CA, USA, 13-16 July. Morgan Kaufmann.

[Koza et al., 1998] Koza, J.R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H. & Riolo, R (eds). (1998). *Proceedings of the Third Annual Conference on Genetic Programming*. University of Wisconsin, Madison, Wisconsin, SA, 22-25 July. Morgan Kaufmann.

[Koza & Bennett III, 1999] Koza, J.R and Bennett III, F.H. (1999). Automatic synthesis, placement, and routing of electrical circuits by means of genetic programming. In Spector, L., Langdon, W.B., O'Reilly, U.M. & Angeline, P.J. (eds.). *Advances in Genetic Programming 3*, MIT Press, Cambridge, MA, USA 105-134.

[Lau & Tsang, 1997] Lau, T.L. & Tsang, E.P.K. (1997). Solving the processor configuration problem with a mutation-based genetic algorithm. *International Journal on Artificial Intelligence Tools (IJAIT)*, World Scientific, Vol.6, No.4, 567-585.

[LeBaron et al., 1999] LeBaron, B., Abu-Mostafa, Y.S., Lo, A.W. & Weigend, A.S. (eds.). (1999). *Computational Finance 1999*, MIT press.

[LeBaron, 2000] LeBaron, B. (2000). Agent Based Computational Finance: Suggested Readings and Early Research, (forthcoming), *Journal of Economic Dynamics and Control*, 2000.

[Lettau, 1997] Lettau, M. (1997). Explaining the facts with adaptive agents: The case of mutual fund flows. *Journal of Economic Dynamics and Control,* 21, 1117-1148.

[LeBaron, 1998] LeBaron, B. (1998). Technical trading rules and regime shifts in foreign exchange. In Acar, E. & Satchell S. (eds.), *Advanced trading rules,* Butter-worth Heinemann.

[Lederman & Klein, 1995] Lederman, J. & Klein, R. (ed.). (1995). *Virtual Trading*. Chicago: Probus Publishing.

[Lehmann, 1990] Lehmann, B.N. (1990). Fad, martingales, and market efficiency. *Quarterly Journal of Economics*, 105, 1-28.

[Leinweber & Arnott, 1995] Leinweber, D.J. & Arnott, R.D. (1995). Quantitative and computational innovation in investment management. *Journal of Portfolio Management*. Winter, 9-15.

[Levy, 1996] Levy, H. (1996). *Introduction to Investments*. South-Western College Publishing.

[Levich & Thomas, 1993] Levich, R.M. & Thomas, L.R. (1993). The significant of technical trading-rule profits in the foreign exchange market: A bootstrap approach. *Journal of International Money and Finance,* 12, 451-474.

[Li, 1999] Li, J. (1999). FGP: A genetic programming tool for financial prediction. *Proceedings of GECCO-99 PhD Student Workshop*, Orlando, Florida, USA, July 13-19 1999. p374.

[Li & Tsang, 1998] Li, J. & Tsang, E.P.K. (1998). Market efficiency, predictability and genetic algorithms, March 1998. Technical Report CSM-307, University of Essex.

[Li & Tsang, 1999a] Li, J. & Tsang, E.P.K. (1999a). Improving technical analysis predictions: an application of genetic programming. *Proceedings of The 12th International Florida AI Research Society Conference*. Orlando, Florida, May 1-5, 1999, 108-112.

[Li & Tsang, 1999b] Li, J. & Tsang, E.P.K. (1999b). Investment decision making using FGP: a case study. *Proceedings of The Congress on Evolutionary Computation* (CEC'99). Washington DC, USA, July 6-9 1999, 1253-1259.

[Li & Tsang, 2000] Li, J. & Tsang, E.P.K. (2000). Reducing failures in investment recommendations using genetic programming. *Proceedings of 6th International Conference on Computing in Economics and Finance, Society for Computational Economics*, Barcelona, July, 2000. (a revised version was submitted to the *Journal of Computational Economics,* under review).

[Lo & MacKinlay, 1990] Lo, A.W. & MacKinlay, A.C. (1990). When are contrarian profits due to stock market overreaction? *Review of Financial Studies* 3, 175-206.

[Lobo, 1991] Lobo, G. (1991). Alternative methods of combining security analysts' and statistical forecasts of annual corporate earnings, *Journal of Forecasting*, 57-63.

[Lukac et al., 1988] Lukac, L.P., Brorsen, B.W. & Irwin, S.H. (1988). A test of futures market disequilibrium using twelve different technical trading systems. *Applied Economics*, 20, 623-639.

[Lukac & Brorsen, 1990] Lukac, L.P., Brorsen, B.W. (1990). A comprehensive test of futures market disequilibrium. *Financial Review*, 25 (4), 593-622.

[MacDonald & Marsh, 1994] MacDonald, R. & Marsh, I. (1994). Combining exchange rate forecasts: what is the optimal consensus measure? *Journal of Forecasting*, 313-332.

[Mahfoud & Mani, 1996] Mahfoud, S. & Mani, G. (1996). Financial Forecasting Using Genetic Algorithms. *Journal of Applied Artificial Intelligence* Vol.10, Num 6, 543-565.

[Malkiel, 1992] Malkiel, B. (1992). Efficient market Hypothesis, in Newman,P., Milgate, M. and Eatwell, J. (eds.), *New Palgrave Dictionary of Money and Finance*. Macmillan, London, pp739.

[Mandelbrot, 1963] Mandelbrot, B. (1963). The variation of certain speculative prices. *Journal of Business*, Vol. 36, 394-419.

[Manning & Schutze, 1999] Manning, C.D. & Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

[Marengo & Tordjman, 1995] Marengo, L. & Tordjman, H. (1995). Speculation, heterogeneity, and learning: in a rational expectation model. Technical report, WP-95-17, International Institute for Applied Systems Analysis, Vienna, Austria.

[Margarita, 1991] Margarita, S. (1991). Neural network, genetic algorithms and stock trading. *Artificial Neural Networks* 1, 1763-1766.

[Margarita, 1992] Margarita, S. (1992). Genetic neural networks for financial markets: Some results. In Neumann, B. (ed.). *Proceedings of 10th European Conference on Artificial Intelligence*, John Wiley & Sons. 211-213.

[Marimon et al., 1990] Marimon, R., McGrattan, E. & Sargent, T, J. (1990). Money as a medium of exchange in an economy with artificially intelligent agents. *Journal of Economic Dynamics and Control*. 14, 329-373.

[Markowitz & Xu, 1994] Markowitz, H. M. & Xu, G.L. (1994). Data mining corrections, *Journal of Portfolio Management*, Fall, 60-69.

[McDonnell et al., 1992] McDonnell, J.R., Anderson, B.L., Page, W.C. & Pin, F.G. (1992). Mobile manipulator configuration optimisation using evolutionary programming. *Proceedings of First Annual Conference on Evolutionary Programming*. San Diego, CA: Evolutionary Programming Society. 52-62.

[Mehta & Bhattacharyya, 1999] Mehta, K. & Bhattacharyya, S. (1999). Combining rules learnt using genetic algorithms for financial forecasting. *Proceedings of the Congress on Evolutionary Computation* (CEC'99), Washington DC, USA, July 6-9 1999, 1245-1252.

[Miller & Goldberg, 1995] Miller, B.L. & Goldberg, D.E. (1995). Genetic algorithms, tournament selection, and the effects of Noise. IlliGAL Report No. 95006.

[Mitchell, 1996] Mitchell, M. (1996). *An introduction to genetic algorithms*. MIT Press.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill, New York..

[Montana, 1995] Montana, D.J. (1995). Strongly typed genetic programming. *Evolutionary Computation*, Vol. 3:2, 199-230.

[Mozetic, 1985] Mozetic, I. (1985). NEWGEM: Program for learning from examples, program documen-tation and user’s guide. University of Illinois Report Number UIUCDCS-F-85-949, Urbana-Champaign, ILL.

[Neely et al., 1997] Neely, C., Weller, P. & Ditmar, R. (1997). Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *Journal of Financial and Quantitative Analysis*, 32, 405-26.

[Neftci, 1991] Neftci, S.N. (1991). Naï ve trading rules in financial markets and Wiener-Kolmogorov prediction theory: A study of 'technical analysis'. *Journal of Business*, 64, No. 4, 549-571.

[Ngan et al., 1998] Ngan, P.S., Wong, M.L. & Leung, K.S. (1998), Using grammar based genetic programming for data mining of medical knowledge. *Genetic Programming 1998*: *Proc. of 3rd Annual Conference*, Morgan Kaufmann, 254-259.

[Nikolaev & Slavov, 1997] Nikolaev, N.I. & Slavov, V. (1997). Inductive genetic programming with decision trees. *Proc. of 1997 European Conference on Machine Learning* (ECML-97).

[Nordin & Banzhaf, 1997] Nordin, P. & Banzhaf, W. (1997). An on-line method to evolve behaviour and to control a miniature robot in real time with genetic programming. *Adaptive Behaviour*, 5:107-140.

[Nolan et al., 1999] Nolan, F., Wilkiewicz, J., Dasgupta, D. & Franklin, S. (1999). Evolutionary Economic Agents. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99). AAAI Press.*

[Nunez, 1988] Nunez, M., (1988). Economic induction: A case study, *Proc. of the Third European Working Session on Learning, EWSL-88*, California: Morgan Kaufmann, 139-145.

[Nunez, 1991] Nunez, M., (1991). The use of background knowledge in decision tree induction, *Machine Learning*, 6, 231-250.

[O'Reilly, 1995] O'Reilly, U.-M. (1995). *An analysis of genetic programming*. PhD thesis, Carleton University, Ottawa-Carleton Institute for Computer Science, Ottawa, Ontario, Canada, 22 September.

[Oussaidene et al., 1997] Oussaidene, M., Chopard, B., Pictet, O. & Tomassini, M. (1997). Practical aspects and experiences - Parallel genetic programming and its application to trading model induction, *Journal of Parallel Computing* Vol. 23, No. 8, 1183-1198.

[Park & Song, 1997] Park, Y. & Song, M. (1997). Genetic programming approach to sense clustering in natural language processing. In Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B. Garzon, M., Iba, H & Riolo, R.L. (eds.) *Genetic Programming 1997: Proceedings of the Second Annual Conference.* Stanford University, CA, USA, July. Morgan Kaufmann. p261.

[Patell & Wolfson, 1984] Patell, J.M. & Wolfson, M.A. (1984). The intraday speed of adjustment of stock process to earnings and dividend announcements. *Journal of Financial Economics*. June, 223-252.

[Palmer et al., 1994] Palmer, R.G., Arthur, W.B., Holland, J.H., LeBaron, B. & Tayler, P. (1994). Artificial economic life: A simple model of a stock market. *Physica D*, 75, 264-274.

[Pau, 1991] Pau, L. (1991). Technical analysis for portfolio trading by syntactic pattern recognition, *Journal of Economic Dynamics and Control*, 15, 715-730.

[Pazzani et al., 1994] Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. (1994). Reducing misclassification costs. In *Proceedings of the 11th International Conference of Machine Learning,* New Brunswick. Morgan Kaufmann, 217-225.

[Pictet et al., 1995] Pictet, O.V., Dacorogna, M.M., Chopard, B., Oudsaidene, M., Schirru, R. & Tomassini, M. (1995). Using Genetic Algorithm for Robust Optimization in Financial Applications. *Neural Network World* Vol. 5. No. 4. 573-587.

[Poli & Cagnoni, 1997] Poli, R & Cagnoni, S. (1997). Genetic programming with user-driven selection: Experiments on the evolution of algorithms for image enhancement. In Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B. Garzon, M., Iba, H & Riolo, R.L. (eds.) *Genetic Programming 1997: Proceedings of the Second Annual Conference.* Stanford University, CA, USA, July. Morgan Kaufmann. 269-277.

[Poli & Langdon, 1997] Poli, R. & Langdon, W.B. (1998). Schema theory for genetic programming with one-point crossover and point mutation. Evolutionary Computation, Vol. 6, no. 3, 231-252.

[Pring, 1991] Pring, M.J. (1991). *Technical analysis explained* (Second Edition). McGraw-Hill, New York.

[Provost & Buchanan, 1995] Provost, F.J. & Buchanan, B.G. (1995). Inductive policy: The pragmatics of bias selection. *Machine Learning*, 20 (1/2): 35-61.

[Provost et al., 1998] Provost, F.J. Fawcett, T. & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. *In Proc. 15th International conference on Machine Learning*, Madison, WI. 445-453.

[Quinlan, 1986a] Quinlan, J. R. (1986b), Induction of decision trees. *Machine Learning*, 1: 81–106.

[Quinlan, 1986b] Quinlan, J. R. (1986c). Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234.

[Quinlan, 1987] Quinlan, J. R. (1987). Generating production rules from decision trees. In *Proc. IJCAI-1987: International Joint Conference on Artificial Intelligence*, pages 304–307, Los Altos, CA. Morgan Kaufmann.

[Quinlan, 1993] Quinlan, J. R., (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann, Los Altos, CA.

[Quinlan, 1996a] Quinlan, J. R. (1996a). Bagging, boosting, and C4.5. In *Proc. AAAI-1996: Thirteen National Conference on Artificial Intelligence*, pages 725–730, Menlo Park, CA. AAAI Press.

[Quinlan, 1996b] Quinlan, J. R., (1996b). Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4: 77–1990.

[Rabatin, 1998] Rabatin, A. (1998). Adaptive Portfolio trading using genetic algorithms. *Proceedings of the Fifth International Conference on Forecasting of Financial Markets*. London, May.

[Raj & Thurston, 1996] Raj, M. & Thurston, D. (1996). Effectiveness of simple technical trading rules in the Hong Kong futures markets. *Applied Economic Letter*, Vol. 3 33-36.

[Raymer et al., 1996] Raymer, M.L., Punch, W.F., Goodman, E.D. & Kuhn, L.A. (1996). Genetic programming for improved data mining: An application to the biochemistry of protein interactions. In Koza, J.R., Goldberg, D., Fogel, D. & Riolo, R. (eds.). *Procedings of the First Annual Conference on Genetic Programming*. Stanford University, CA, USA, July. MIT Press. 375-380.

[Rechenberg, 1973] Rechenberg, I. (1973). Evolutionsstratrategie: optimierung technischer systeme nach prinzipien der biologischen evolution. Stuttgart: Frommann-Holzboog.

[Roberts et al., 1995] Roberts, H., Denby, M. & Totton, K. (1995). Accounting for misclassification costs in decision tree classifers. In *Intelligent Data Analysis* (*IDA*-95).

[Rosca, 1997] Rosca, J.P. (1997). Analysis of complexity drift in genetic programming. In Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B. Garzon, M., Iba, H & Riolo, R.L. (eds.) *Genetic Programming 1997: Proceedings of the Second Annual Conference.* Stanford University, CA, USA, July. Morgan Kaufmann. 286-194.

[Rose, 1997] Rose, C.P. (1997). *Robust Interactive Dialogue Interpretation*. PhD thesis, Language Technologies Institute, Carnegie Mellon University.

[Ross, 1976] Ross, S.A. (1976). The arbitrage theory of capital asset pricing. *Journal of Economic Thoery*, 13, No. 3, 341-360.

[Routledge, 1994] Routledge, B. R. (1994). Artificial selection: Genetic algorithms and learning in a rational expectations model. Technical report, GSIA, Carnegie Mellon, Pittsburgh, Penn.

[Rumelhart & McClelland, 1986] Rumelhart, D.E. & McClelland, J.L. (eds.). (1986). *Parallel distributed processing: Explorations in the microstructure of cognition: Volumn I: Foundations.* Cambridge, Mass.: MIT Press.

[Rust et al., 1994] Rust, J., Miller, J.H. & Palmer, R. (1994). Characterizing effective trading strategies: Insights from a computerized double action tournament. *Journal of Economic Dynamics and Control*. 18 (1). 61-96.

[Saad et al., 1998] Saad, E., Prokhorov, D., and Wunsch, D., (1998). Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks, *IEEE Transactions on Neural Networks*, vol. 9. 1456-1470.

[Schwefel, 1981] Schwefel, H.-P. (1981). *Numerical optimisation of computer models*, Wiley, Chichester.

[Schulmeister, 1987] Schulmeister, S. (1987). An essay on exchange rate dynamics. Wissenschaftszentrum Berlin fur Sozialforschung, Berlin. IIM/LMP 87-8.

[Schwefel, 1981] Schwefel, H.-P. (1981). *Numerical optimisation of computer models*. Wiley, Chichester.

[Sharpe, 1964] Sharpe, W.F. (1964). Capital asset price: A theory of market equilibrium under condition of risk, *Journal of Finance*, 19, No.3, 425-442.

[Sharpe et al., 1995] Sharpe, W.F., Alexander, G. Bailey, J.V. (1995). *Investments (Fifth Edition)*. Prentice Hall, Inc.

[Silber, 1994] Silber, W. (1994). Technical trading: when it works and when it doesn’t. *Journal of Derivatives*, 1, Spring, 39-44.

[Smith, 1980] Smith, S.F. (1980). *A learning system based on genetic adaptive algorithms*. Doctoral dissertation. University of Pittsburgh, Pittsburgh, Pa.

[Smith, 1983] Smith, S.F. (1983). Flexible learning of problem solving heuristics via adaptive search. In *Proceedings of the eighth international joint conference on artificial intelligence*. 422-425.

[Spector, 1996] Spector, L. (1996). Simultaneous evolution of programs and their control structures. In Angeline, P. & Kinnear, Jr., K.E. (eds.). *Advances in genetic programming 2*, MIT Press, Cambridge, MA. 137-154.

[Salomon, 1998] Salomon, R. (1998). Short notes on the schema theorem and building block hypothesis in genetic algorithms. In Porto, V.W. Saravanan, N., Waagen, D.E. & Eiben, A.E. (eds.). *Evolutionary Programming VII: Proceedings of the 7th Annual Conference on Evolutionary Programming,* San Diego, CA. Springer-Verlag. 113-124.

[Sweeney, 1986] Sweeney, R. J. (1986). Beating the foreign exchange market. *Journal of Finance*, 41 (1), 163-182.

[Sweeney, 1988] Sweeney, R. J. (1988). Some new filter rule test: Methods and results. *Journal* *of Financial and Quantitative Analysis*, 23, 285-300.

[Taylor, 1995] Taylor, P. (1995). Modelling artificial stocks markets using genetic algorithms. In Goonatilake, S. & Treleaven, P. (eds.). *Intelligent systems for finance and business*. 271-288.

[Taylor, 1986] Taylor, S.J. (1986). *Modeling financial time series*. Wiley, New York, NY.

[Taylor, 1992] Taylor, S.J. (1992). Rewards available to currency futures speculators: compensation for risk or evidence of inefficient pricing? *Economic Record*, 68 (Supplement): 105-116.

[Taylor, 1994] Taylor, S.J. (1994). Trading futures using the channel rule: A study of the predictive power of technical analysis with currency examples. *Journal of Futures Markets*, 14(2), 215-235.

[Taylor & Allen, 1992] Taylor, M.P. & Allen, H. (1992). The use of technical analysis in the foreign exchange market. *Journal of International Money and Finance*, 11, 304-314.

[Ting & Zheng, 1998] Ting, K.M. & Zheng, Z. (1998). Boosting trees for cost-sensitive classifications. In *Proc. 10th European Conference on Machine Learning*. Chemnitz, Germany. 191-195.

[Treynor & Ferguson, 1985] Treynor, J., and Ferguson, R. (1985). In defence of technical analysis. *Journal of Finance*, 40, 757-773.

[Tsang, 1993] Tsang, E.P.K. (1993). *Foundations of constraint satisfaction*. Academic Press, London.

[Tsang et al., 1998] Tsang, E.P.K., Li, J. & Butler, J.M. (1998). EDDIE beats the bookies, *International Journal of Software, Practice & Experience*, Wiley, Vol.28 (10), 1033-1043.

[Tsang & Li, 1999] Tsang, E.P.K., Li, J. (1999). A genetic programming tool for financial forecasting. (Submitted to *Journal of forecasting*, under review).

[Tsang & Li, 2000] Tsang, E.P.K. & Li, J. (2000). Combining ordinal financial predictions with genetic programming. *Proceedings of the Second International Conference On Intelligent Data Engineering And Automated Learning*. (IDEAL2000) December 2000. Hong Kong.

[Tsang et al., 2000] Tsang, E.P.K., Li, J., Markose, S., Er, H., Salhi, A., and Iori, G. (2000). EDDIE in financial decision making. (Submitted to *Journal of Finance and Management*).

[Turney, 1995] Turney, P.D. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2, 369-409.

[Turney, 1997] Turney, P.D. (1997). Cost-sensitive learning bibliography. Online bibliography, Institute for Information Technology of the National Research Council of Canada, Ottawa, Cadada, http://ai.iit.nrc.ca/bibliographies/cost-sensitive.html.

[Vacca, 1997] Vacca, L., (1997). Managing options risk with genetic algorithms. *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering* (CIFEr), New York City, March 1997, 29-35.

[Vriend, 1994] Vriend, N.J. (1994). Self-organized markets in a decentralised economy. Santa Fe Institute, Economics Research Program, Working paper 94-03-013.

[Wall & Correia, 1989] Wall, K. & Correia, C. (1989). A preference -based method for forecast combination. *Journal of Forecasting*, 269-192.

[Webb, 1996] Webb, G. I. (1996). Cost-sensitive Specialization. *Proc. of the 1996 Pacific Rim International Conference on Artificial Intelligence*. Cairns, Springer-Verlag, 23-34.

[Werner et al., 1987] Werner, F.M., Bondt, D. & Thaler, R. (1987). Further Evidence on Investor Overreaction and Stock Market Seasonality. *Journal of Finance*, 42, No. 3, July. 557-581.

[Weiss, 1999] Weiss, G. M. (1999). Timeweaver: a Genetic Algorithms for identifying predictive patterns in sequences of events, In *Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco, CA: Morgan Kaufmann. 718-725.

[Weiss & Kulikowski, 1991] Weiss, S.M. & Kulikowski, C.A. (1991). *Computer system that learn*. Morgan Kaufmann.

[White, 1998] White, A.J. (1998). A genetic adaptive neural network approach to pricing options: a simulation analysis. *Journal of Computational Intelligence*, Vol. 6, No. 2, 13-23.

[Whigham, 1996] Whigham, P. A. (1996). *Grammatical Bias for Evolutionary Learning*. PhD thesis, School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy.

[Yao, 1993] Yao, X. (1993). A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*. (8). 539-567.

[Yao and Liu, 1998] Yao, X., & Liu, Y. (1998). Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28, 417-425.

[Yao et al., 1997] Yao, X., Kim, J.-H. & Furuhashi, T. (1997). *Simulated Evolution and Learning*. Lecture Notes in Artificial Intelligence, (eds.), Volume 1285, Springer-Verlag, Heidelberg, Germany.

[Yao, 1999] Yao, X. (1999). *Evolutionary Computation: Theory and Applications*. (ed.), World Scientific Publ. Co., Singapore.