ON THE UTILITY OF EVOLVING FOREX MARKET TRADING AGENTS WITH CRITERIA BASED RETRAINING

by

Alexander Loginov

Submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
March 2013

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "ON THE UTILITY OF EVOLVING FOREX MARKET TRADING AGENTS WITH CRITERIA BASED RETRAINING" by Alexander Loginov in partial fulfillment of the requirements for the degree of Master of Computer Science.

Dated: March 25, 2013

Supervisor:

Dr. Malcolm I. Heywood

Readers:

Dr. Garnett Wilson

Dr. Vlado Keselj

DALHOUSIE UNIVERSITY

DATE: March 25, 2013

AUTHOR: Alexander Loginov

TITLE: ON THE UTILITY OF EVOLVING FOREX MARKET TRADING AGENTS WITH CRITERIA BASED RETRAINING

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: M.C.Sc. CONVOCATION: May YEAR: 2013

Signature of Author

Table of Contents

List of Tables

List of Figures

Abstract

This research investigates the ability of genetic programming to build profitable trading strategies for the Foreign Exchange Market (FX) of one major currency pair (EURUSD) using one hour prices from July 1, 2009 to November 30, 2012. We recognize that such environments are likely to be non-stationary and we do not expect that a single training partition, used to train a trading agent, represents all likely future behaviours. The proposed adaptive retraining algorithm – hereafter FXGP – detects poor trading behaviours and trains a new trading agent. This represents a significant departure from current practice which assumes some form of continuous evolution. Extensive benchmarking is performed against the widely used EURUSD currency pair. The non-stationary nature of the task is shown to result in a preference for exploration over exploitation. Moreover, adopting a behavioural approach to detecting retraining events is more effective than assuming incremental adaptation on a continuous basis. From the application perspective, we demonstrate that use of a validation partition and Stop-Loss (S/L) orders significantly improves the performance of a trading agent. In addition the task of co-evolving of technical indicators (TI) and the decision trees (DT) for deploying trading agent is explicitly addressed. The results of 27 experiments of 100 simulations each demonstrate that FXGP significantly outperforms existing approaches and generates profitable solutions with a high probability.

List of Abbreviations and Symbols Used

Hereafter the following terminology will be assumed [16]:

Ask or Offer	Price at which broker/dealer is willing to sell
Balance	The value of your account not including unrealized gains or losses on open positions
Bear Market	An extended period of general price decline in an individual security, an asset, or a market
Bid	Price at which broker/dealer is willing to buy
Bull Market	A market which is on a consistent upward trend
Drawdown	The magnitude of a decline in account value, either in percentage or dollar terms, as measured from peak to subsequent trough
Equities	Ownership interest in a corporation in the form of common stock or preferred stock
Fundamental Analysis	Macro or strategic assessment of where a currency should be trading on any criteria but the price action itself. The criteria often include the economic condition of the country that the currency represents, monetary policy, and other 'fundamental' elements
Futures	An obligation to exchange a good or instrument at a set price on a future date
Leverage	The use of various financial instruments or borrowed capital, such as margin, to increase the potential return of an investment

Limit Order	An order placed with a brokerage to buy or sell a set number of shares at a specified price or better. Limit orders also allow an investor to limit the length of time an order can be outstanding before being canceled
Market Noise	Price and volume fluctuations in the market that can confuse one's interpretation of market direction. Used in the context of equities, it is stock market activity caused by program trading, dividend payments or other phenomena that is not reflective of overall market sentiment. In general, the shorter the time frame, the more difficult it is to separate the meaningful market movements from the noise
Money Management	The process of budgeting, saving, investing, spending or otherwise in overseeing the cash usage of an individual or group. The predominant use of the phrase in financial markets is that of an investment professional making investment decisions for large pools of funds, such as mutual funds or pension plans. Also referred to as 'investment management' and/or 'portfolio management'
Pip	The smallest change of the last decimal point the currency pair is priced to. Often referred to as 'ticks' in the future markets. For example, in EURUSD, a move of 0.0001 is one pip
Spot	Buying and selling forex with the current date's price for valuation, but where settlement usually takes place in two days
Spread	The distance, usually in pips, between the Bid and Ask prices
Stop Loss or S/L	A limit order to close a position when a given limit is reached. When long, the stop loss order is placed below the current market price. When short, the stop loss order is placed above the current market price

Technical Analysis	Analysis applied to the price behaviour of the market to develop trading decision, irrespective of fundamental factors
Trading Curb	A temporary restriction on program trading in a particular security or market, usually to reduce dramatic price movements. Also known as a collar or circuit breaker

Chapter 1

Introduction

The Foreign Exchange (FX) Market is the world biggest financial market which produces 1/3 of all financial transactions in the world [28]. The average daily turnover of FX was almost $4 trillion in 2010 and it was 20% higher in April 2010 than in April 2007 [33]. An FX market consists of currency pairs which are weighted by economic conditions for that specific denomination versus any or all others in the marketplace. Thus, the price value of a currency is nothing more than a reflection of where that denomination's economy ranks as weak or strong relative to others. An FX market is technically 'purer' than a stock market, i.e. a currency price behaviour reacts more strongly to resistance and support levels than equity markets do [30] but this is not the only difference.

1.1 Forex, Futures and Equities Markets

Historically, the Forex market[1] was the exclusive playground of banks, hedge funds, corporations and financial institutions [2] due to regulation, capital requirements, and technology. One of the main reasons why the Forex market was traditionally the market of choice for large players is because the risk that a trader takes is fully customizable. A trader could use a hundred times leverage or do not use it at all[2]. But this situation has changed. Many firms have opened the Forex market to retail traders, providing leveraged trading, free instantaneous execution trading platforms and real-time news. The popularity of Forex has started to grow and many equity and futures traders started to trade currencies or even switched to Forex exclusively. The traders have realized that Forex has attractive attributes compared to equities or futures [21].

[1] The "forex market" is mostly referred to the Spot Forex Market [2].

[2] With a 1:100 leverage FX traders are allowed to execute trades of up to $100,000 with an initial deposit of only $1000. But while a high degree of leverage allows traders to maximize their profit potential, the potential for loss is equally large [2]

The short summary of the main attributes of the Forex comparing to Equities and Future markets are listed below [21].

Forex market:

- Forex is the largest market in the world and has growing liquidity.
- Forex operates 24-hour a day and 5 days a week.
- Traders can profit in bull and bear markets (see Glossary for definition).
- There are no trading curbs[3,4].
- Instant executable trading platform minimizes slippage and errors.
- Many traders consider Forex more profitable.

Equities market:

- Equity market liquidity mainly depends on the stock daily volume.
- The market is open for trading from 9:30 a.m. to 4:00 p.m. New York time with limited after-hours trading.
- The existence of exchange fees results in higher costs and commissions.
- There is an uptick rule to short stocks, which many day traders find frustrating. Trading curbs may be frustrating for day traders as well.
- The number of steps involved in completing a trade increases slippage and error.

Future markets:

- The liquidity is limited and depends on the month a contract is traded.

[3]Unlike the equities market, there is never a time in the FX market when trading curbs would take effect and trading would be halted, only to gap when reopened. This eliminates missed profits due to exchange regulations. In the FX market, traders would be able to place trades 24 hours a day with virtually no disruptions [21].

[4]When the "curbs are in" at the NYSE, it means that certain types of trading are restricted to prevent volatility. Depending on the situation, this can mean that either all trading is halted or that certain sales can be executed only on an uptick. This kind of rule was implemented after the crash of 1987 (Black Monday), as program trading was thought to be a primary cause of the drop [17]

- The exchange fees result in more costs and commissions.
- Market hours are much shorter and depend on the product traded.
- The leverage is higher than for equities but less than for Forex.
- There tend to be prolonged bear markets.
- Pit trading structure increases error and slippage.

1.2 The Ways To Analyze Financial Markets

There are two major ways to analyze financial markets [21]: fundamental analysis and technical analysis. Fundamental analysis is based on the underlying economic conditions and is not applicable to this research, while technical analysis uses historical prices in an effort to predict future movements [21].

Technical analysis works well for the Forex market. Charts and TIs are used by all professional FX traders and are available in most charting packages. In addition, the most commonly used TIs – such as Fibonacci Retracements, Stochastic Oscillator, Moving Average Convergence/Divergence (MACD), Moving Averages (MA), Relative Strength Index (RSI), and support/resistance levels – have proven valid in many instances [21].

In effect traders are using information from technical indicators in order to draw out trends which enable them to make predictions regarding the future behaviour of the market. Such predictions provide the basis for trading actions e.g., buy, sell or hold. The combination of a TI with a decision rule provides the basis for a "trading strategy". This work is based on technical analysis and investigates the ability of genetic programming to build profitable trading strategies for the Forex market. The thesis is organized as follows:

- Chapter 2 overviews previous work that are related to our research.
- Chapter 3 gives the detailed description of the proposed algorithm; hereafter FXGP.
- Chapter 4 describes the experimental setup and the result of 27 experiments.

- Chapter 5 summarizes the conclusions and discusses the topics for future work.

Chapter 2

Background

The introduction summarized the case for constructing automated trading agents using machine learning in general. It was also established that such an automatic trading agent needed to combine the activities of feature construction — or technical indicators (TI) – with the design of an appropriate decision tree – or under what conditions to apply one of a predefined set of market specific actions. The research of this thesis adopts an approach to machine learning taken from genetic programming in particular and concentrates on:

1. Establishing the mechanism for supporting the co-design of TI and DT, and;
2. Establishing how to interface the trading agent to the data stream representing the market.

In the following summary of related research we begin by introducing the basic architecture for the canonical forms of linear genetic programming, where this is the representation assumed in this research (Section 2.1). Section 2.2 will review some of the properties associated with markets and trading and their potential impact on assuming a genetic programming framework. Finally, Section 2.3 summarizes recent approaches to applying genetic programming to the task of automated trading agents.

2.1 Canonical Linear Genetic Programming

In this work we will use some biological terms as denoted in Genetic Algorithms (GA) and Genetic Programming (GP) books by Mitchell [27] and Koza [19].

All cells in a living organism contain the same set of *chromosomes* – strings of DNA – a 'blueprint' of the organism. A chromosome can be divided into functional blocks – *genes*, each gene encodes a *trait* of the organism (e.g.eye colour) and possible 'settings' for a trait (e.g. blue) and are called *alleles*. All the chromosomes of the

organism collectively constitute the *genome* and the set of the particular genes in a genome is called *genotype*. The *phenotype* is a set of the traits and behaviour characteristics which reflect the influence of the environment. A living organism behaviour can be *diploid* (paired chromosomes) or *haploid* (unpaired chromosomes). Most sexually reproducing organisms are diploid. During sexual reproduction two main genetic operations occur: *recombination* or *crossover* – genes exchange, and *mutation* – when some *nucleodites* (elementary parts of DNA) are changed (often as a result of errors introduced during reproduction). The *fitness* typically reflects the probability that the organism will survive to reproduce or as a function of the number of the offsprings of the organism [27].

A GA is a radical abstraction of its biological model [1] and in short can be summarized [19, 31] as follows (Algorithm 1):

Algorithm 1 A short summary of a GA as a radical abstraction of its biological model

1: Initialize a population, size $\mathbf{P}$, with multiple candidate solutions
2: **while** !*stop_criteria* **do**
3: Evaluate fitness of all individuals in population $\mathbf{P}$
4: **for** $j < |P|/2$ **do**
5: parent(a,b) = Selection($\mathbf{P}$)
6: crossover(a,b,p_c)
7: mutation(a,p_m)
8: mutation(b,p_m)
9: $\mathbf{P^1} \leftarrow$ insert(a,b)
10: **end for**
11: $\mathbf{P} \leftarrow$ Replacement($\mathbf{P}$,$\mathbf{P^1}$)
12: **end while**

Step 1 initializes a population with *multiple* candidate solutions following the definition of an appropriate representation. Step 2 merely indicates that execution continues until either a task specific or computational stop criterion is encountered. Step 3 implies that each candidate solution should be evaluated on the task, and a measure of performance given to each individual in the population. This is usually task specific. Step 4 creates a population of children, P^1. To do so, pairs of parents are

selected relative to the content of the current population, P. Two variation operators are then used to introduce diversity. Crossover trades (genetic) material between parents to create children (e.g., see Figure 2.1), hence is limited to material which currently exists in the population. Mutation is a gene-wise operator and 'flips' a gene between any of its alleles. Finally, replacement (Step 11) merges the parent and child populations to create a new parent population of size |P| i.e., there is a competition for 'survival'.

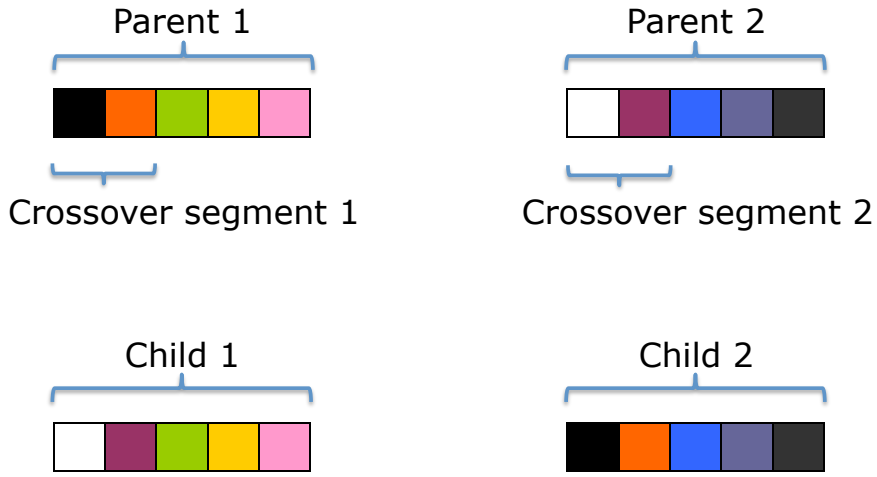


Figure 2.1: Crossover trades (genetic) material between parents to create children.

Naturally, variation operators are representation specific i.e., limited to operating on the genotype. Conversely, selection and replacement are free to make use of performance information, thus operate on the 'search space'. We will later assume a 'breeder' approach, thus children always replace the worst pair of individuals currently in the population. This also implies that the process is elitist with the best individuals never being lost.

The canonical GA of Algorithm 1 is generic to both genetic algorithms (GA) and genetic programming (GP). The principal difference between the two is how a genotype is 'decoded' into its corresponding phenotype. From the perspective of a GA the decoding results in a phenotype representing a scalar or vector of real-valued or integer numbers. In the case of GP – and linear GP in particular – the decoding between genotype and phenotype is synonymous with the "fetch–decode–execution" cycle of a register machine. Thus, genotypes represent (binary) instructions which are associated with fields of a simple register machine. For example, consider a binary

instruction consisting of the tuple $\langle m, t, o, s_1, s_2 \rangle$ or a total of five instruction fields. Let m denote the instruction type (or mode) where instruction types might take the form of register–register, load–store, or register–constant; hence a total of 2 bits. Target and source registers (operands) are identified by t and s_i respectively. Thus, if a total of four registers exist then another 2 bits are necessary (per field). Instruction 'opcode' is denoted by o; thus another 2 bits would be required if the opcodes take the form of the four arithmetic operators. In this particular case, encoding a single instruction takes the form of a sequence of 10 bits or an integer selected over the range $[0, \ldots, 1023]$. A linear GP individual would therefore consist of a 'string' of integers with execution beginning at the first instruction and continuing until the last. The output from the program is generally assumed to be the content of register zero following execution of the program. Depending on the application domain the user determines an appropriate set of opcodes and instruction types or a *register level transfer* language. The necessary decoding can then be designed to ensure that all genotypes result in a valid instruction. Section 3 discusses in detail the representation and instruction opcodes assumed for FXGP.

The genetic programs can be classified by three basic forms of representation [1]: tree, graph and linear representation. This research adopts the tree representation [19] to construct the trading rules (DT) and linear genetic programming to build technical indicators (TIs). This decision was made based on the distinguishing characteristics and main differences of the two approaches that were summarized in the monograph of Brameier and Banzhaf [1] as follows.

The tree programs used in [19] correspond to expressions (syntax trees) from a functional programming language. This approach is also referred to as tree-based GP (TGP). Functions – or rather operations with more than zero arguments – are defined by the inner nodes, while leaves of the tree hold input values or constants, i.e., argument zero operations. In contrast, linear genetic programming (LGP) is a GP variant that evolves sequences of instructions from an imperative or a machine programming language. The set of instructions are restricted to operations including conditional operations, that accept a minimum number of constants or memory variables, called registers, and assign the result to another register (e.g. $r0 := r1 + 1$).

The term linear refers to the structure of the (imperative) program representation,

as opposed to standing for functional genetic programs that are restricted to a linear list of nodes only. Moreover, it does not mean that the method itself can solve linearly separable problems only. Rather, genetic programs often represent highly non-linear solutions due to their inherent power of expression [1].

The GP does not produce parsimonious code and the genetic program contains lots of useless code structures (*introns*) [19]. From the perspective of LGP, Introns are either structural or semantic. Structural introns fall outside the path of code associated with the output register. Conversely, semantic introns represent code that has no functional value – say $r1 := r1+0$ – but are in the path of execution associated with the output register. Tree structured GP assumes the root node as the source of output, thus there is no concept of structural introns. Structural introns are easier to detect and can therefore be efficiently skipped during execution. Conversely, the variation in the form of semantic introns make them more difficult to detect (the degree of difficulty depending on the instructions supported). Introns in themselves are a function of several factors but one of the most prominent factors is the stochastic nature of the variation operators [1]. Specifically, crossover and mutation operators are applied stochastically – see Figure 2.1 – hence intron code may actually evolve to 'protect' the functional code such that the children are as fit as the parents. Introns have a complex relationship with other factors such as gene epistasis that potentially make credit assignment more difficult to resolve. However, without such epistasis it is also the case that the task is linearly separable, thus would not be an appropriate task to solve using evolutionary methods [39]. The way to treat the introns assumed in this research (Section 3.2.1), tries to to keep balance between the effectiveness of the code and the advantages that can be obtained as a result of introns presence in the program.

2.2 On the role of Dynamic or Non-stationary Processes

Trading agents are required to operate in an environment that is described in terms of a 'data stream' which is fundamentally non-stationary or dynamic [3]. Relative to the classical approach as established for, say, the context of supervised learning, then constructing models (e.g., as in classification or function approximation) from an a priori identified training partition with post training evaluation on an independent

test partition is not feasible [8]. At the very least, the model identified relative to a single training partition will only have a lifetime appropriate to a finite future duration after which changes to the underlying process creating the stream will render the model unusable. This means that an 'online' approach needs adopting to constructing a model. Adopting an online approach is relatively easy when a representation takes the form of real-valued free parameters[1]. Thus, for example, in the case of neural networks, the goal of the credit assignment process is to identify the values for 'weights' associated with a given architecture. In this case, offline learning algorithms can be readily modified to the online context of streaming data. For example, the least mean square algorithm associated with the gradient descent family of credit assignment algorithms for MLP style neural networks either averages the error term over all data (offline) or performs updates on an exemplar wise basis (online) [32].

Given that the data takes the form of a non-stationary stream the question then arises regarding how much data should a model receive before it makes a decision. In general offline approaches are stateless. Thus, decisions are made relative to each exemplar and moreover, the decisions are independent of exemplar sequence e.g., the independent and identically distributed assumption (i.i.d.). Conversely, decisions made at any point in a data stream are generally sensitive to the ordering of the stream. This represents a requirement to have a stageful model. Again, several generic schemes could be adopted. The simplest methodology to adopt is to assume that the temporal properties of the stream can be captured by an appropriate form of feature constructor. Examples might include Fourier or Wavelet basis functions [24], finite impulse response (FIR) or Kalman filters [11]. From the machine learning perspective such approaches have the benefit of letting us continue to use the a stateless model (to make the decision) and limiting the search to that of parameterizing the assumed feature constructor. Conversely, instead of separating feature construction from decision making the model could be responsible for both. Thus, adopting a neural network representation that included recurrent connectivity or assuming a linear genetic program in which register values were not reset after each exemplar would both result in a stateful model. The penalty for this is a much more deceptive process of credit assignment. As established in the introduction, in the research reported by

[1]Free parameters are the components of a model that the machine learning algorithm adapts through credit assignment.

this thesis we will adopt a stateless representation for the decision tree (DT) describing trading actions and coevolve the DT with a population of technical indicators (feature constructors).

In the case of genetic programming, the monograph of Dempsey et al., [3] recognize five properties for the successful application of evolutionary computation to dynamic environments: memory, diversity, multi-populations, problem decomposition, and plasticity. The relative role of each is summarized as follows, with additional comments being made as to the degree of significance provided to them by the approach adopted by this thesis:

Memory: is associated with the desirability to provide some mechanism by which previously useful solutions are archived. The underlying assumption in this is that the process generating the stream is periodic in some way. In the research conducted in this thesis we take the view that support for periodic properties is not particularly important in the currency trading domain. Dempsey et al. [3] appear to also down play the overall significance for supporting memory mechanisms.

Diversity: is associated with the exploration–exploitation tradeoff. Canonical GP as summarized in Section 2.1 will converge as the number of generations increases. Thus, at initialization maximum diversity exists, thereafter the fitter individuals will receive a greater opportunity to reproduce, resulting in a gradual dominance of exploitation over exploration. This is to be expected in canonical GP as the data set – defining the environment – is assumed to conform to that of offline learning. However, when streaming data is involved and the underlying process is non-stationary, then the maintenance of diversity (exploration) becomes more important. At the very least, what was previously considered 'fit' will only be dominant for some period of time, after which it might be necessary to modify or depart from the previously dominant individuals. Mechanisms adopted for diversity maintenance include:

1. A greater frequency of mutation – not particularly effective on account of the number of non-functional children produced;

2. Fitness sharing (or speciation) – discounting fitness relative to how many other individuals also perform well on the same task; and;

3. Age heuristics – an additional bias towards the promotion of solutions of a

particular age [9].

In the approach adopted by this thesis we will assume that population content is completely reset with entirely new genotypic material when a retraining event is encountered.

Multi-population: represents a more explicit form of speciation or diversity maintenance. In keeping entirely independent populations there is direct support for peripatric speciation e.g., Chapter 9, [26]. From a purely biological perspective this results in the development of founder populations that follow an independent and distinct path of evolution from an original source population, potentially resulting in a new species. However, from the perspective of explicitly utilizing such a concept for evolutionary computation it is difficult to make specific recommendations, even with respect to the context of offline data sets.

Problem decomposition: potentially impacts the rate at which solutions can be discovered and useful 'building blocks' promoted between different candidate solutions. Herbert Simon expressed the case for problem decomposition and therefore 'reuse' through several parables of which that of the two watchmaker's is the most pertinent to online learning [34]. Specifically, Simon describes two watchmakers, one who constructs watches using a modular approach of independent subassemblies and one who constructs each watch independently. Naturally, the modular approach promotes assembly even under very 'noisy' conditions whereas any interruption of the second watchmaker results in the entire watchmaking process having to be reinitiated. Under dynamic environments constructing models from modules may then result in faster evolution overall. The approach adopted in the research of this thesis is to assume a symbiotic approach to the coevolution of TI and DT. Thus, TI that are useful in one DT are easily promoted between multiple DT with minimal epistasis (deceptive gene-wise interaction). Previous research in symbiotic bid-based genetic programming has illustrated the potential for symbiosis relative to the evolution of monolithic code structures [20].

Plasticity: or 'evolvability' is associated with how changes to the underlying representation (care of variation operators) are able to promote 'useful' variation in the phenotype. Naturally, given the many-to-one relationship between genotype and fitness (many individuals may have the same fitness) significant ambiguity can

exist. Thus, evolutionary computation is often associated with various properties that potentially make credit assignment less than transparent e.g., neutral genetic content and introns [1]. In this research the emphasis on modularity care of symbiosis represents the principal approach for supporting the plasticity of the evolutionary process. That said, we will also augment this with ideas from machine learning. In particular we attempt to be as objective as possible about the conditions under which training events are re-triggered and introduce validation data in a manner consistent with the online streaming context.

2.3 Evolving trading agents

Genetic Algorithms in the automated trading systems are typically used to select a subset of TIs which then are used to discover hidden regularities in the historical rates [14], to optimize the parameters [6] of the selected TIs and to build a trading rule (a decision tree) [29] that employs these TIs.

As described in the previous work [22], some previous research has been reported with respect to the evolution of technical indicators (TI) alone e.g., [7]. One of the goals of evolving TI independently is to provide parameterizations for basic statistics such as moving averages, that are particularly appropriate for the data in question [38]. More recently separate TI were evolved for buying and selling [12]. Several authors have evolved trading rules or decision trees (DT) relative to a set of a priori selected TI e.g., [4, 13]. One of the most well known instances of GP in this context is 'EDDIE' [36]; the resulting IF–THEN rules are sampled by a human trader to determine which to use. Most recently, systems have been proposed to first evolve parameters for a fixed set of TI with the identification of DT e.g., [15]. To do so, a representation is assumed in which all of the information for the TI and DT appear in the same genome. We consider this problematic as both components need to be correct for an individual to be successful. Instead we assume a recent framework in which TI and DT are explicitly co-evolved in two independent populations [23]. Moreover, with respect to the facts that:

- TIs were designed to help a human trader to understand the markets behaviour and not necessarily are optimal for the automated trading systems.

- Some very popular TIs are just a combination of the two or more other TIs (e.g MACD is a combination of two EMAs and SMA)[2].

We adopt the linear GP [1], [40] to build unique TIs rather than just optimize the parameters of the existing TIs.

In general, authors have placed a lot of effort on designing fitness functions to penalize losses as well as reward profit [3, 15]. This also has implications for the potential robustness of the resulting trading agent as a whole. An alternative / complementary approach is to make use of verification data to curtail a training cycle, or early stopping [37]. Also of widespread utility is the use of some form of continuous training, where this is in recognition of the non-stationary nature of the task. For example, a 'rolling window' approach might be adopted [15] in which a sequence of data (or window) is used for training (N_t) and the champion individual deployed as the trading agent for the next δ data points. On reaching the end of the trading period the training window of N_t is realigned with the end of trading and training recommences. Questions potentially arise regarding the number of generations to perform [3]. This is related to the degree of non-stationary (or appropriateness of any model bias) associated with the transition between training and trading periods. In this work we compare training re-triggered based on trading criteria versus the typically assumed approach of continuous evolution.

Finally, we investigate the effectiveness of Stop Loss orders (hereafter S/L) which are widely used by traders [2] but is not a norma in known researches, in the automated trading systems.

[2]http://ta.mql4.com/indicators/oscillators/macd

Chapter 3

Proposed Algorithm

3.1 The FXGP Algorithm Overview

As established in the first chapters FXGP produces a DT–TI combination, hereafter trading agent, that is used for trading (Algorithm 2). A cycle of evolution is initially completed against the first N_t records (lines 5...7) and validated against the next N_v records (lines 8...19). The purpose of validation is to identify a single 'champion' TI–DT individual (with respect to current FXGP content) for performing trading. Trading (lines 21...24) then lasts until some failure criteria is satisfied (section 3.4) at which point the $N_t + N_v$ records leading up to the failure are used to evolve a new trading rule, and the process repeats (Figure 3.1). The trading rule takes the form of a decision tree (DT population) and corresponding set of *co-evolved* technical indicators (TI population). Explicitly evolving the TI provides the opportunity to capture properties pertinent to specific nodes of a decision tree defining the overall trading rule, where the characterization of such temporal properties is known to be significant for a wide range of time series tasks [38]. Four prices – Open, High, Low and Close – are used as inputs to the TI population. Members of the DT population define the trading rule, and it is only with respect to the DT individuals that fitness is evaluated i.e., the TI are simultaneously co-evolved with the DT with fitness only ever being defined or a symbiotic relationship [5]. FXGP parameters used in the description are summarized by Table 3.1.

Algorithm 2 The core FXGP algorithm (Table 3.1 for parameters description). *Retrain* refers to one of the retrain criterion (Section 3.4)

Input: The historical rates of EURUSD pair (1 hour resolution).

Output: Trading rule (DT with linked TIs).

$t = StartFrom$ {define start date/time of trading}
repeat
  $best = NULL$ (no DT–TI agent)
  while $best == NULL$ **do**
    initialize TI population
    initialize DT population
    evolve DT and TI populations over (N_t)
    $n = 0$ {DT–TI agents count}
    for $i = 0$ to t_{size} **do**
      if (validate DT_i over N_v) == TRUE **then**
        $n++$
      end if
    end for
    if $n >= (t_{size} - t_{gap})$ **then**
      for $i = 0$ to n **do**
        find $(\text{DT–TI})_i$ agent with the highest score
      end for
      $best = i$ {best DT–TI agent for deployment}
    end if
  end while
  while $Retrain == False$ **do**
    trade
    $t--$
  end while
until $t > 0$

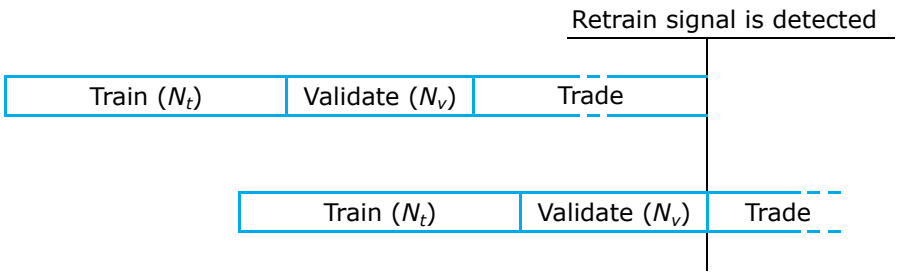


Figure 3.1: The Train-Validate-Trade cycle.

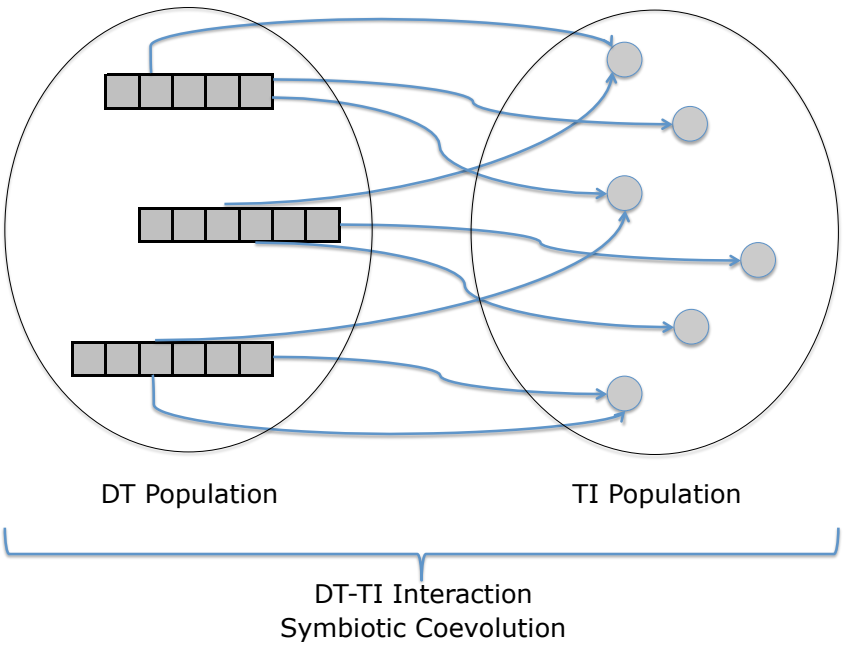


Figure 3.2: DT–TI Interaction.

Parameter	Description
t_{size}	Maximum number of nodes in the DT
p_{size}	Maximum size of the TI program
n_{regs}	Number of registers in the TI program
n_{ma}	Number of records back in a price history to calculate MA or WMA
n_{shift}	Number of records back in a price history to select a price
sl_{min}	Minimum SL size in pips
sl_{max}	Maximum SL size in pips
s	Spread size in pips
Parameters specific for Training procedure only (Section 3.2)	
N_t	The number of records in the training subset
n_{gnr}	Maximum number of generation to train
n_{flt}	Plateau size
t_{psize}	DT population size
t_{pgap}	Number of DT to replace in each generation
r	Trading signals ratio
p_{ti}	Probability of the TI mutation
p_{sl}	Probability of the SL mutation
Parameters specific for Validation procedure only (Section 3.3)	
N_v	The number of records in the validation subset
l	DT score limit
Retrain criteria (Section 3.4)	
d_{min}	Minimum drawdown size in pips
d_{max}	Maximum drawdown size in pips
n_{nt}	Maximum number of consecutive records in a price history without of trading activity
n_{row}	Maximum number of consecutive losses

Table 3.1: FXGP parameters

3.2 Training

3.2.1 Initialization

As per line 5, Algorithm 2, the *TI population* is randomly initialized. The size of the *TI population* population is not fixed but the initial size is defined as (3.1).

$$\frac{t_{psize} \times t_{size}}{4} \tag{3.1}$$

where t_{psize} and t_{size} as described in the Table 3.1.

Each TI has a header that defines the basic TI properties: reference counter, type, scale, period, shift and TI program's length (Table 3.2). Each of these properties will be described further. All TI individuals assume a linear GP representation (e.g., [1]) with instruction set summarized by Table 3.3.

Header field	Description
Reference counter	Shows how many DT are using the TI
TI type	0 - Moving Average (MA), 1 - Weighted Moving Average (WMA), 2 - Value
TI scale	0 - TI that crosses 0, 1 - TI that crosses price (Appendix A)
Period n	Number n of hours in a price history to calculate MA or WMA
Shift m	Price m hours back in a history
Length	The length of the TI program

Table 3.2: TI properties

Function	Definition
Addition	$R[x] \leftarrow R[x] + R[y]$
Subtraction	$R[x] \leftarrow R[x] - R[y]$
Multiplication	$R[x] \leftarrow R[x] \times R[y]$
Square root	$R[x] \leftarrow \sqrt{R}[y]$
Division	$R[x] \leftarrow R[x] \div R[y]$
Division	$R[x] \leftarrow 1 \div R[x]$
Division	$R[x] \leftarrow R[x] \div 2$

Table 3.3: TI functions

TI programs assume a register level transfer language (Section 2.1) with n_{regs} registers (Table 3.3) where $R[x]$ denotes the content of register x and $R[y]$ denotes either: register y content, a price or a price n_{shift} hours back in (relative) time.

Register $R[0]$ contains a TI value after executing a TI program. The program counter corresponding to where $R[0]$ appears as a target register for the last time is stored in the header as the *Length* of the TI program. The program steps between the *Length* and p_{size} are considered introns and ignored[1]. The MA type of TI is calculated as (3.2) and the WMA type of TI is calculated as (3.3), where V_i is a TI value.

$$MA_i = \frac{\sum_{j=1}^{n} V_j}{n} \tag{3.2}$$

$$WMA_i = \frac{\sum_{j=1}^{n} \frac{V_j}{j+1}}{\sum_{k=1}^{n} \frac{1}{k+1}} \tag{3.3}$$

The *Reference counter* is initialized with 0, the *TI type, Period* and *Shift* are initialized with randomly selected non-negative values within a range specified by the user. Thus, *TI type* is initialized with 0, 1 or 2. *Period* and *Shift* are initialized with non-negative values less then n_{ma} or n_{shift} respectively. The *TI scale* serves to detect useless TI and is checked after the initialization. The TI initialization is repeated until the *TI scale* is valid (Appendix A).

When the *TI population* is ready, the *DT population* is initialized. The *DT population* size t_{prize} is fixed and defined by the user. A DT header includes the following information: DT *score* in pips, number of trades and a size of a S/L order (see Glossary for S/L definition) in pips. The score and number of trades are used to determine the fitness of a DT–TI agent and initialized with 0 whereas the S/L can be initialized in two ways: if the probability of the S/L mutation p_{sl} is greater than 0, the S/L is initialized with a randomly selected value between the minimum sl_{min} and maximum sl_{max} (defined by user) S/L sizes (hereafter *Floating S/L*) or, if the probability of the S/L mutation p_{sl} is 0, S/L is initialized with a sl_{max} (hereafter *Fixed S/L*). A DT consists of a variable number of nodes (> 1 and $<= t_{size}$). Each node consists of a conditional statement with either single or dual antecedent tests of the form:

- *if*$(X_i > Y_i)$ *then else*
- *if*$((X_i > Y_i)$ *and* $(X_{i+m} < Y_{i+m}))$ *then else*

[1]This is not meant to in any way reflect the true number of introns, which has empirically been shown to account for 70% to 80% of instructions (e.g., [1]). It does however provide a simple mechanism for ignoring irrelevant code.

where X_i and Y_i can be 0, price or a TI. If X_i and/or Y_i represent a TI, the *Reference counter* of the linked TI(s) is incremented. The *then* and *else* statements reference either: the next node or one of the trading actions: buy, sell or stay. Thus, a DT population is randomly generated with respect to X_i and Y_i scales[2] and under the following constraints:

- if X_i is 0, then Y_i can be any TI which crosses 0 and can not be a price or a TI which crosses the price, or;
- if X_i is price, then Y_i can be also a price or a TI which crosses the price, and can not be 0 or a TI which crosses 0.

In the case of a dual antecedent clause, X_{i+m} and Y_{i+m} represent the value of X_i or Y_i respectively, albeit n_{shift} samples back in (relative) time. FXGP can generate additional TI during DT population initialization if the TI population does not have a TI capable of satisfying the DT initialization constraints.

3.2.2 Fitness and Selection

The financial reward of a Forex trader is measured in pips which are then converted into dollars based on the a user's preferred scheme of money management [25]. Thus, we define the DT fitness as a DT score in pips over the training records as a 'pure' measure that is not dependent upon the scheme assumed for pips-to-dollars conversion. When TI and DT populations are initialized, FXGP simulates the trading activity for each DT over the training records and stores the score and the number of trades in the DT header. Moreover, FXGP calculates the number of actions of each type (*stay*, *buy* or *sell*) and if a DT does not generate actions of all three types, its score is discarded and the DT targeted for replacement. To prevent a DT from generating too frequent *buy* or *sell* actions the trading signals ratio r is used (Table 3.1). If r is less than $(stay \div (buy + sell))$, the DT score is multiplied by coefficient k, where k is calculates as (3.4).

$$k = \frac{stay}{(buy + sell) \times r} \tag{3.4}$$

[2]Distinguishes between a TI associated with a zero crossing versus price (Table 3.2)

The subset of DT individuals with lowest scores is targeted for replacement and the *Reference counter* of the linked TIs are decremented. Thus, a t_{pgap} individuals of the DT population is replaced per generation. All variation is asexual, thus following parent selection, cloning and variation takes place where either the DT or linked TI component of a clone can be varied. Following DT selection, any TI individual that, is not linked to any DT (TI's reference counter value is 0) is considered ineffective and deleted (resulting in a variable size TI population).

3.2.3 Mutation

FXGP uses mutation to produce offspring. Only the S/L parameterization and one DT node or one linked TI can be mutated in each cloned DT. FXGP randomly selects a parent DT, clones it and then mutates the size of Floating S/L of the offspring with probability p_{sl}. After that the target for mutation (TI or node) is selected based on the probability of mutation p_{ti} (Table 3.1). A *DT Mutation* is performed by applying one of the following operators:

- Generate a new conditional function;
- Generate a new shift parameter m;
- Generate new X_i and Y_i;
- Interchange X_i and Y_i within the same individual;
- Switch content of *then* and *else* clauses;
- Insert new *then* clause content.
- Insert new *else* clause content.

Likewise a *mutated TI* is first cloned to avoid interfering with other DT employing the same TI. The following parameters and functions of a TI can be mutated:

- TI type;
- Period (n);
- Shift (m);

- Generate a new function;
- Delete a function;
- Insert a function.

where function insertion or deletion is performed with respect to the TI program size limits p_{size}.

Training stops when the specified number of generations n_{gnr} was reached or when the best score in the DT population plateaus for a n_{flt} number of generations (Table 3.1). Thereafter a validation cycle is initiated.

3.3 Validation

The validation process (Algorithm 2, lines 8...19) is used to check the quality of the DT–TI population and to select the best DT–TI agent for trading. To do so, FXGP checks the score of every tree in a DT population and if the DT score is greater than $l \times$ *best score* it tests the DT and if DT is active the tested DT–TI agents counter is incremented (Algorithm 2, line 11). When all DT are evaluated, FXGP checks the number of the tested DT and if it is greater than $t_{prize} - t_{gap}$ then the DT with best score on validation (and referenced TIs) are selected for trading, otherwise the TI and DT populations are reinitialized and the entire training procedure is restarted i.e., rather than invest more generations in a population which fails under validation we choose to reinitialize and restart the training cycle.

3.4 Trading and Retraining criteria

During FX trading, the following three trading quality criteria are monitored and used to identify when the current DT–TI agent should be replaced (Algorithm 2, lines 21...24):

- Drawdown (d_{min} and d_{max}). The drawdown value is different for each trainig cycle and depends on the result of the previous cycle. If the result of the previous cycle was positive (profit), then the d_{max} value is used, and d_{min} otherwise. The first training-validation-trading cycle uses the d_{max} value.

- The number of consecutive losses (n_{row}).
- The number of consecutive hours without trading activity (n_{nt}).

This approach makes FXGP more flexible and forces it to retrain the DT population only when the market situation is deemed to differ from that of the last training period. Specific values for the quality criteria are established by the user. When the quality criteria are exceeded, FXGP stops trading, re-initializes the TI and DT populations and restarts the training-and-validation cycle. Content of the training and validation partitions is taken relative to the point 't' (Algorithm 2, line 23) at which point the trading quality criteria interrupted trading. Once a new DT–TIs agent is identified trading resumes at the point trading was interrupted.

Chapter 4

Evaluation

In this empirical study we assume the FXGP framework as described in Chapter 3 and we investigate the relative effectiveness of:

- Criteria based retraining (FXGP) versus the typically assumed continuous evolution of trading agents [3] [15].
- Selection of the DT–TI trading agent for deployment based on the results of the validation procedure (Section 3.3). This implies that the most recent data are used for validation but not for the training, versus deploying the best DT–TI trading agent (the agent with the highest score) just after the training (Section 3.2).
- The use of S/L orders in automated trading and the relative effectiveness of the Fixed versus Floating S/L orders (Section 3.2).

Continuous evolution is normally used in the area of GA–GP applicable to Forex and stock markets [3] [15]. Thus, we assume the framework for continuos evolution ('moving window' approach as described in [3], or likewise 'rolling window' approach [15]). The DT–TI agents are evolved and the best DT–TI agent is selected for deployment in the same way as for FXGP, but the trading period has a fixed length δ as shown in Figure 4.1.

After the δ hours the trading is stopped[1] and the Train-Validate cycle is restarted and the DT and TI populations are either [22]:

- Utilized as if evolution progressed from the last generation. Any new content assumes the bias established by the previous cycle of evolution. This scheme reflects the intent of recent practice (e.g., [3], [15]) and hereafter will be referred

[1] In the case of an open trading order by the end of trading period, the trading continues until the open order is closed. This potentially can lead to a some variation in the length of trading periods.

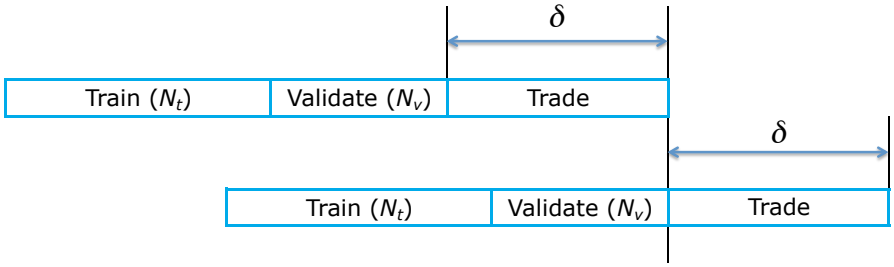


Figure 4.1: The Train-Validate-Trade cycle (trading period of a fixed length δ).

to as continuous evolution or *ContEv*. Such a scheme naturally reflects a bias towards exploiting previously evolved policies. We also note that current practice is not to employ a separate validation partition to choose a champion trading agent for deployment. Indeed, under continuous evolution, benchmarking revealed that it was not possible to employ the validation criteria from Section 3.3 to deploy a validation partition. Part of this might be due to a lack of diversity in the population of trading agents, however, this remains a topic for future research;

- Completely reinitialized before a new training–validation cycle begins. This means that any previously evolved individuals are replaced with an entirely new population as per the initial cycle of evolution. This represents a natural alternate case that emphasizes exploration. Hereafter this will be referred to as stepwise evolution or *StepEv*. Unlike continuous evolution, benchmarking did not reveal any problems with the inclusion of a validation partition for stepwise evolution.

4.1 Source Data

The previous work [23] shows that FXGP demonstrates profitable trades at least for the three major currency pairs (EURUSD, USDCHF and EURCHF) over a 3 year trading period. In this work we limit ourselves to the major currency pair of EURUSD (27 experiments of 100 runs each). The principle motivation is to investigate the significance of the mechanism used to interface FXGP to the market. As established in the introduction, this issue is generally overlooked by current research in trading

agents. The historical rates of EURUSD pair (1 hour resolution) were downloaded from the FXHISTORICALDATA.COM Forex Historical Data repository[2]. The one hour intervals is selected to reduce the impact of "market noise" (see Glossary for definition) [18]. The dataset includes 2009-2012 historical rates (20974 records) and consists of following fields: Pair, Date, Time, bid price Open (Open), bid price Low (Low), bid price High (High) and bid price Close (Close). The trading simulations begin on January 2, 2010, 12am and stop on November 30, 2012, 11pm.

4.2 Experimental setup

Performance was evaluated for the following cases:

1. DT–TI trading agents as evolved from a single initial cycle of evolution, hereafter *Static*. This establishes the baseline against which the degree to which the underlying process is considered to be non-stationary. Thus, if a trading agent was profitable for three years following an initial training period of three months, then the underlying process must be stationary. Under such a case any form of retraining would be unnecessary.

2. Continuous retraining [3, 15] of DT–TI agents with fixed retrain intervals (120, 500 and 1500 hours), hereafter *ContEv 120, ContEv 500, ContEv 1500* respectively. The use of three different retrain intervals enables us to vary the degree of overlap with the (fixed) training period N_t. As noted in [3], depending on the amount of non-stationary behaviour in the data, more or less frequent updates to the trading agents might be necessary.

3. Retraining of DT–TI agents with fixed retrain intervals (120, 500 and 1500 hours) and DT–TI populations re-initialization in the beginning of each train cycle, hereafter *StepEv 120, StepEv 500, StepEv 1500* respectively. Relative to case 2 this means that rather than carry over the current population content between retraining intervals, the entire population is first reinitialized. Such a scheme is emphasizing exploration over exploitation.

[2]http://www.fxhistoricaldata.com/EURUSD/

4. FXGP (retraining DT-TI trading agents interactively as established by the trading quality criterion). As emphasized in Section 3, FXGP uses retraining criteria to trigger retraining events as opposed to conducting retraining at a fixed rate regardless of the quality of trades. Moreover, when retraining is triggered, the TI–DT populations are entirely reinitialized, as per case 3.

100 runs were performed in each case. Each run simulates trading activity for the selected currency pair over the trading period from January 2, 2010 to November 30, 2012. The currency pair is popular during all trading sessions i.e., a trading day typically consists of 24 samples. The trading conditions were assumed as following [10]:

- Minimum price fluctuation: 0.00001.
- Leverage: 1:100.
- Average spread: 0.2.

The parameterization is described in the Table 4.1. The values of the parameters were determined[3] via experiments performed in previous works [23] and [22]. All runs were repeated for p_{sl} 0 (fixed S/L 100 pips) and 0.2 (S/L is evolved during the training) and for N_v 0 (no validation) and 500 (with validation) (Table 4.2) with the exception of ContEv case which can be run with N_v 0. This limitation arises from the proposed validation algorithm (Section 3.3) that can result in no DT–TI agent being selected for deployment, thus, requiring the re-initialization of the DT and TI populations.

After that, four winning solutions (one from each case) were evaluated without S/L orders Figure B.11 (marked with ‡ in Table 4.2). And finally, the median scores were converted into final account balance in USD assuming an initial balance of 100,000 USD, leverage 1:100 and that only 2000 USD (two percent of the initial balance) may be reinvested at the next round of trading. Thus, the pip value is defined as[4] $2000 \times 100 \times 0.0001$ and is equal to 20 USD. The investment of 100000 USD in RBC Canadian Equity Income (RBF762) mutual funds over the same period of time would result in 116732 USD (5.45% rate of return) of ending portfolio value[5]. The

[3]No claims regarding optimality of the parameters are made.
[4]http://fxtrade.oanda.ca/learn/intro-to-currency-trading/conventions/pips
[5]https://services.rbcgam.com/portfolio-tools/public/investment-performance/

Parameter	Value	Parameter	Value
t_{size}	6	p_{size}	6
n_{regs}	2	n_{ma}	98
n_{shift}	8	sl_{min}	5
sl_{max}	100	s	0.2
Training procedure (Section 3.2)			
N_t	1000	p_{ti}	0.5
n_{gnr}	1000	n_{flt}	200
t_{prize}	100	t_{pgap}	25
r	10	p_{sl}	0 or 0.2
Validation procedure (Section 3.3)			
N_v	0 or 500	l	0.95
Retrain criteria (Section 3.4)			
n_{row}	3	n_{nt}	72
d_{max}	400	d_{min}	200

Table 4.1: FXGP parameterization. d_{min}, d_{max}, n_{nt} and n_{row} are ignored in Static, ContEv and StepEv cases

mutual fund was randomly selected from "2012 Mutual Fund Recommended List" [35] and initial deposit of $100000 was evenly distributed among five fund names: RBC Canadian Short-Term Income Fund (Series ADV), RBC Bond Fund (Series ADV), RBC Advisor Canadian Bond Fund (Series ADV), RBC Global Bond Fund (Series ADV) and RBC Corporate Bond Fund (Series ADV).

4.3 Results

The results of all experiments are summarized in the Table 4.2. The results are sorted by median scores in pips over 100 runs in each case.

The following observations can be made based on the obtained results:

- There is a good correlation between the number of profitable runs and median score.
- FXGP, which dynamically changes retrain intervals, demonstrates the highest efficiency among the all tested cases with the biggest number of profitable runs (82%) and the highest median score (1523 pips)[6]. Assuming a typical (median)

[6] N_v 500, p_{sl} 0.2

Algorithm	N_v	p_{sl}	Prof. runs, %	Score, pips					USD
				min	1st	median	3rd	max	
FXGP†	500	0.2	82	-3088	383	1523	2640	5299	130450
FXGP	500	0	79	-2924	296	1447	2581	5086	128940
StepEv 500†	500	0	73	-3879	-235	1118	1947	5232	122366
StepEv 500	500	0.2	73	-3201	-34	923	2168	4902	119917
StepEv 120	500	0.2	73	-2723	-89	870	1863	4403	118455
FXGP‡	500	n/a	66	-4333	-670	779	2040	5141	115572
FXGP	0	0.2	60	-4378	-746	671	1792	4683	113428
StepEv 120	0	0.2	63	-3659	-718	580	1663	3960	111607
StepEv‡ 500	500	n/a	60	-4233	-881	567	1632	5153	111343
StepEv 120	500	0	66	-3438	-433	550	1630	5796	111009
ContEv 1500†	0	0.2	61	-3921	-1021	535	1852	4116	110691
StepEv 1500	0	0.2	58	-3167	-853	366	1460	3624	107311
StepEv 1500	0	0	55	-4018	-1373	303	1482	3716	106053
StepEv 500	0	0	53	-4558	-1151	169	1275	4454	103380
ContEv 500	0	0.2	53	-3206	-740	122	1137	3554	102449
FXGP	0	0	53	-3842	-1030	67	1123	4764	101333
StepEv 500	0	0.2	52	-3856	-939	59	1417	4695	101184
StepEv 1500	500	0	50	-4206	-1206	55	1019	4050	101107
ContEv 120	0	0.2	52	-4136	-1100	15	1373	3908	100306
StepEv 120	0	0	49	-3052	-1195	-48	1014	4043	99036
StepEv 1500	500	0.2	45	-4491	-1202	-69	1130	4087	98621
Static†	0	0.2	38	-4961	-1358	-421	468	2602	91575
ContEv‡ 1500	0	n/a	40	-5990	-1625	-501	983	4211	89985
Static	0	0	37	-3776	-1532	-668	781	3631	86634
Static	500	0.2	23	-5734	-2031	-805	-144	3167	83909
Static	500	0	23	-5911	-2347	-1162	-110	3844	76766
Static‡	500	n/a	16	-6586	-2876	-1680	-257	2673	66379

Table 4.2: Results sorted by median score (over 100 runs). Where: N_v and p_{sl} as described in the Table 3.1, Prof. runs — percent of profitable runs, min — minimum score, 1st — first quartile, 3rd — third quartile, max — maximum score, USD — final account balance in USD recalculated based on median score in pips.

score, FXGP would outperform the investment in the mutual fund by 82%.

- The use of the validation partition significantly improves the results of the two best cases: FXGP (number of profitable runs is better by 37% and median score is better by 127%) and StepEv 500 (number of profitable runs is better by 40% and median score is better by 895%). At the same time the use of validation partition decreases the results in the Static case: the number of profitable runs by 39% and the median score by 91%.

- The use of S/L orders improves the performance in all four cases. The Fixed S/L orders (100 pips) increase the number of profitable trades of FXGP by 20% and the median score by 86%. The use of Floating S/L orders additionally improves the FXGP results: the number of profitable runs by 4% and the median score by 5%.

- All twenty seven configurations can be profitable and all twenty seven have a chance to outperform the investment in a mutual fund (see 'max' column of Table 4.2) but only five configurations would outperform the investment in a mutual fund in a typical case (see 'median' column of Table 4.2).

- FXGP with validation (N_v 500) is the only algorithm with non-negative 1st, median and 3rd quartiles.

- The chance of loss that exists in all cases (the probability of loss is vary from 0.18 to 0.84 for different configurations) and the fact that it is not a priori possible to determine which run will be a priori profitable, shows that it is very important to use multiple runs to evaluate performance.

- The worst performance in the Static case (trains DT–TI populations only once and uses it over the long trading period) among all the tested cases and distribution of retrain intervals of the winning FXGP configuration (Figure 4.3) illustrates the non-stationary nature of trading and confirms that use of a single DT–TI combination is very likely to result in loss over the long trading period.

Figure 4.2 shows the pips scores distribution (over 100 runs) for the four best configurations (marked by † in Table 4.2) representing each case. Specifically, the

100 runs associated with each of the best performing parameterizations for FXGP, StepEv, ContEv and Static are expressed as a combined violin–box plot. The violin establishes the nature of the distribution whereas the box plot establishes the 1st, 2nd (median) and 3rd quartile (whiskers denote the last data point within a multiple of the nearest quartile). It is apparent that all distributions are essentially unimodal conforming to a Gaussian distribution, with the ContEv case returning a much wider variance.

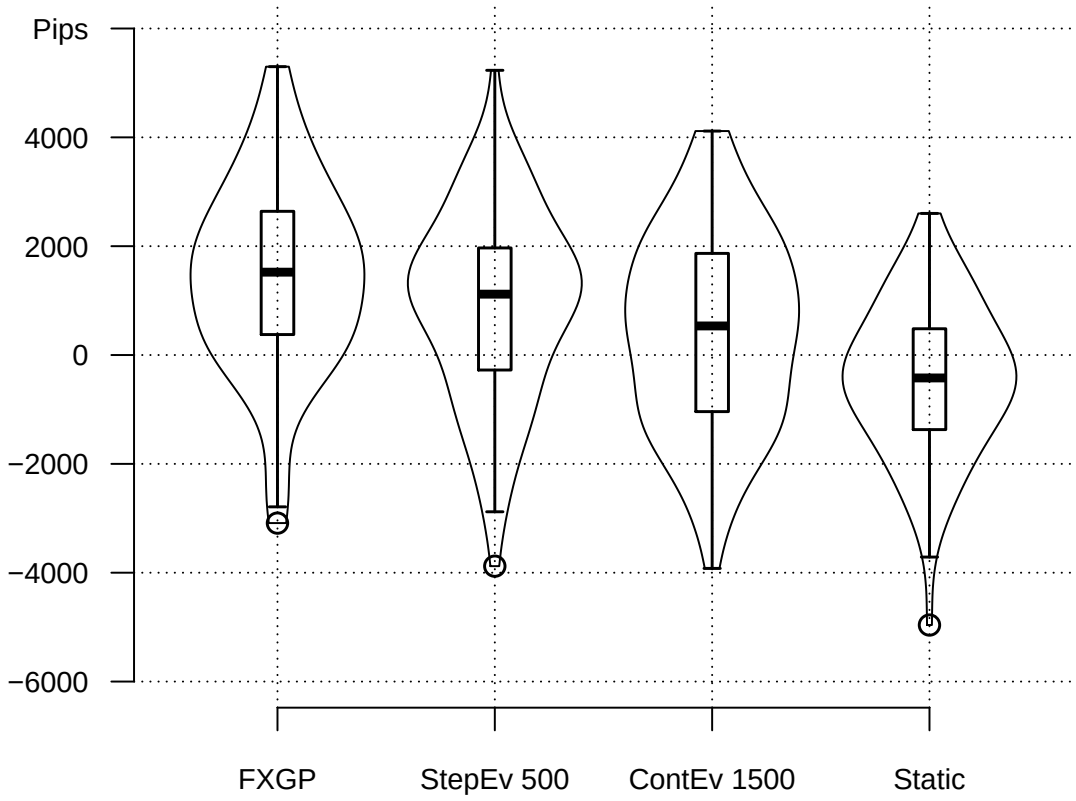


Figure 4.2: FXGP versus StepEv 500 versus ContEv 1500 versus Static (marked by † in Table 4.2).

A Student's T-test (Table 4.3) confirms the independence of the distributions.

FXGP vs. StepEv	FXGP vs. ContEv	FXGP vs. Static
0.0332	6.112×10^{-5}	4.448×10^{-13}

Table 4.3: *p-values* for pairwise Student T-test of FXGP versus StepEv, ContEv and Static (marked by † in Table 4.2).

Figures 4.3, 4.4 and 4.5 show the distribution of retrain intervals, number of retrains and S/L orders size (respectively) of FXGP with validation and floating S/L orders size, hereafter "the winning solution". The box plots show the spread of the 1st, 2nd (median) and the 3rd quartiles that demonstrates adaptation of FXGP to the market changes.

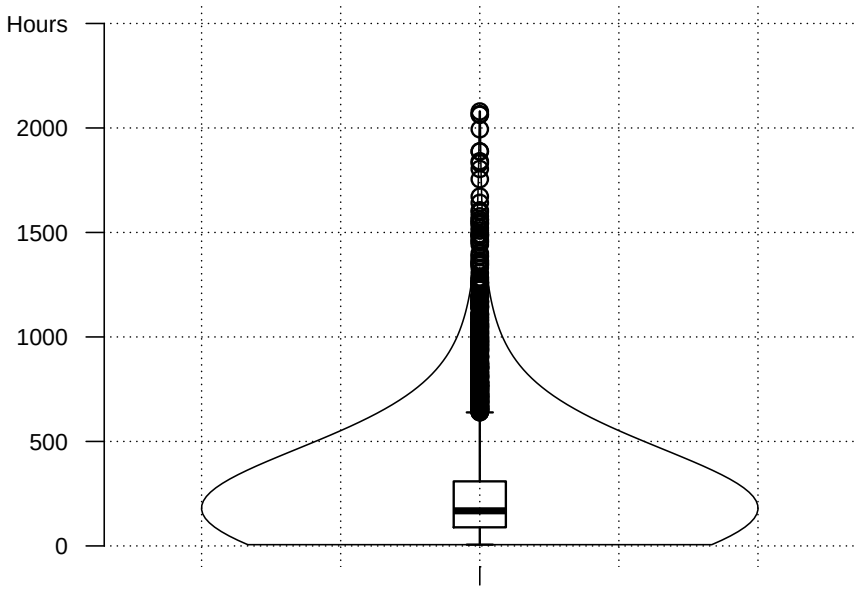
Relative to the distribution of retraining intervals (Figure 4.3), it is apparent that on the one hand the re-triggering is very consistent (subplot (a)), but also that a variation of between 300 to 100 hours appears (subplot (b)). This is equivalent to a spread of between 4 days to 12.5 days. Thus, there is significant degree of application context evident in the re-triggering event.

Figure 4.4 provides another perspective on the impact of the capacity of FXGP to initiate re-triggering by plotting the total number of retraining events for each of the 100 runs. Again the quartile distribution is associated with a very consistent. Given that the deployment covers a 3 year period then it follows that there are typically 25 retraining events per year. Moreover, given that retraining is completed in 30 seconds on an iMac desktop[7], then from the perspective of the 1 hour 'reporting rate' associated with the data stream, it may also be claimed that FXGP operates in real-time.

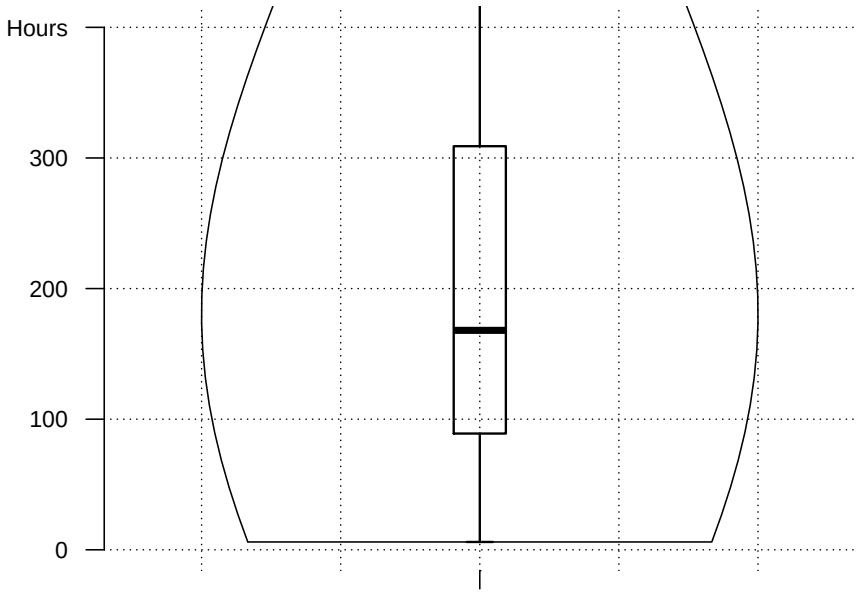
Figure 4.5 summarizes to what degree the S/L orders are customized. Optimization is clearly apparent, with a spread of approximately 50 to 85 pips in the 1st to 3rd quartiles, however, the distribution itself is also clearly unimodal.

Finally, Figure 4.6 and Figure 4.7 show the account balance (pips and USD respectively) in case of the best, typical (median) and worse runs of the FXGP parameterization from Figure 4.2. Note that the 1st to 3rd quartile distributions for this parameterization of FXGP were all profitable (figures illustrate best, median and worst).

[7]Intel Core i7, 2.8 GHz, 16 GB RAM 1333MHz DDR3, OS X 10.7.5



(a) 0. . . 3000 hours



(b) 0. . . 500 hours

Figure 4.3: FXGP retrain intervals (N_v 500, p_{sl} 0.2).

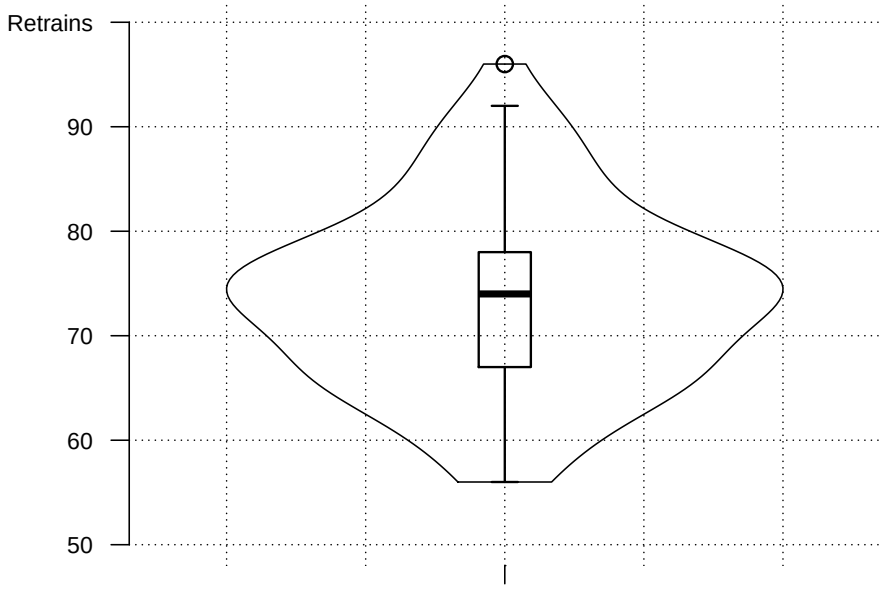


Figure 4.4: FXGP retrains over 100 runs (N_v 0, p_{sl} 0.2).

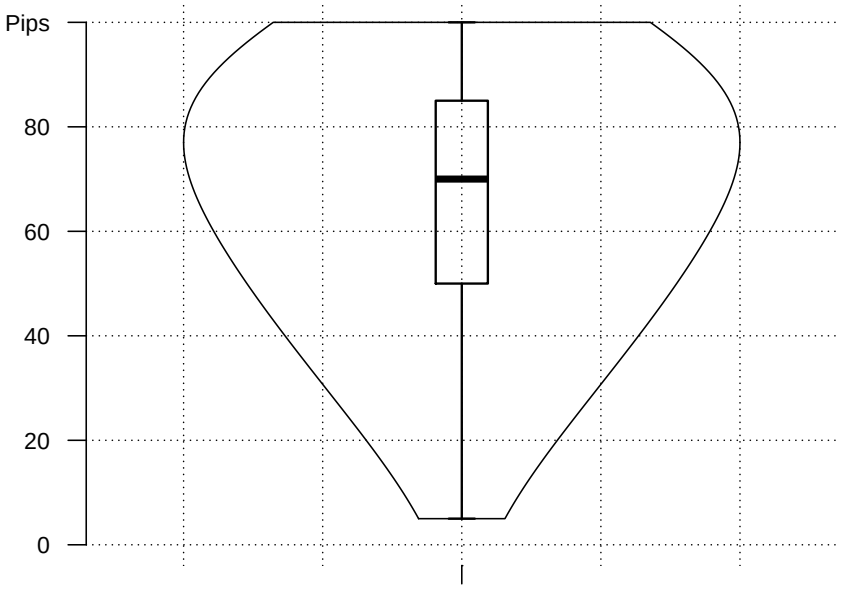


Figure 4.5: FXGP S/L orders in pips (N_v 500, p_{sl} 0.2).

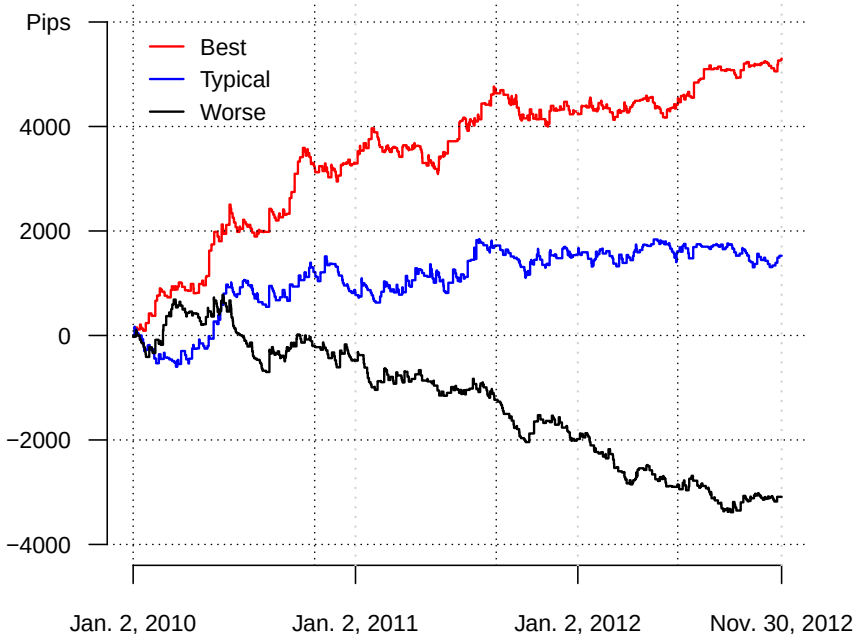


Figure 4.6: FXGP trading profiles (pips) of the best, typical (median) and the worse runs (N_v 500, p_{sl} 0.2).

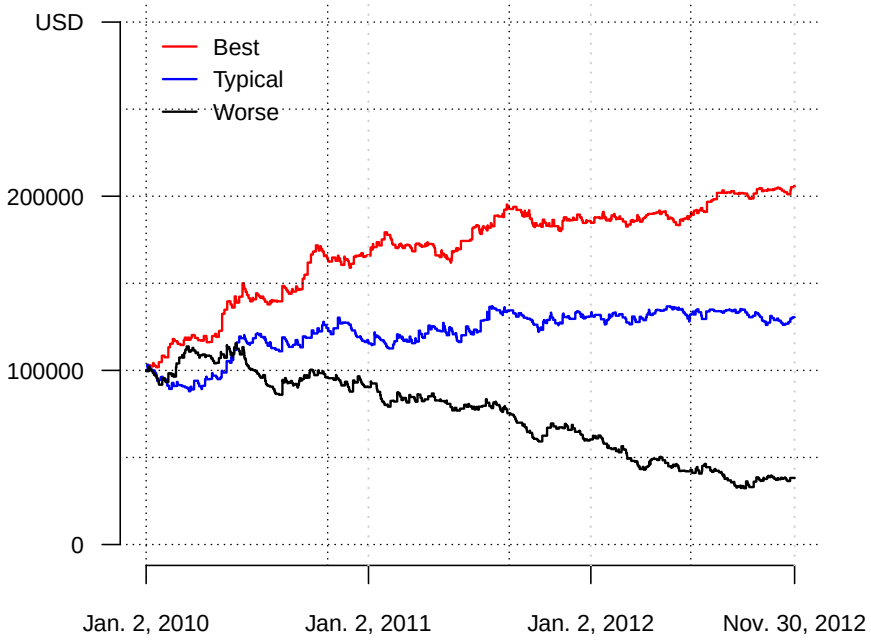


Figure 4.7: FXGP trading profiles (USD) of the best, typical (median) and the worse runs (N_v 500, p_{sl} 0.2).

The results distributions of all tested configurations over 100 runs are shown in Appendix B.

Chapter 5

Conclusion and Future Work

This research was performed to develop an effective algorithm for automated continuous FX trading and to assess the value of the following factors:

- Testing the quality of the DT–TI population and selecting the best trading agent using independent validation data partition.
- Reinitialization and retraining of the DT–TI population to generate a new trading agent.
- Use of a set of criteria to evaluate the trading performance of the trading agent and define the moment to re-trigger the evolution of a new trading agent *during* trading.
- Support the use of S/L orders as a part of trading rule and compare the Fixed and Floating S/L orders.

The use of criteria specifically designed to detect the onset of poor trading behaviour, and therefore trigger the identification of a new trading agent, was demonstrated to be significantly more effective than the use of retraining intervals (with or without continuous evolution) and appears to be the most important factor to increase the results. In addition, the worst performance of the Static case confirms the non-stationary nature of the task.

Reinitialization of the DT–TI population before retraining improves the performance compared to continuous evolution [3, 15]. This illustrates the importance of maintaining the diversity of the DT–TI population. This result runs against current practice in which continuous evolution of a population initialized once is the norm. We are careful to note that changing the type of trading, say to that of a stock market, might result in a different conclusion. However, current practice does not appear to question the need to consider alternatives to continuous evolution.

The use of the validation partition to test the DT–TI population quality and to select the best trading agent (if the population has passed the test) is the second important way to get better results i.e., exploration appears to play a more significant role than exploitation.

The S/L orders also significantly improve the results by reducing the size of losses. The use of Floating S/L is not so important, however it improves the final scores too.

Earlier research has already established that FXGP is capable of discovering profitable strategies with different currency pairs [21]. Future research could revisit the degree of preference for continuous evolution versus criteria based retraining under markets with potentially very different properties, e.g., stock markets or currency markets with different time scales. The use of 'teams' to group independently trained trading agents, the use of separate DT-TI populations to open and close trades and extension of the ability of FXGP to build more types of TI (e.g. TIs similar to EMA or Parabolic SAR) might be considered too.

Appendix A

TI types

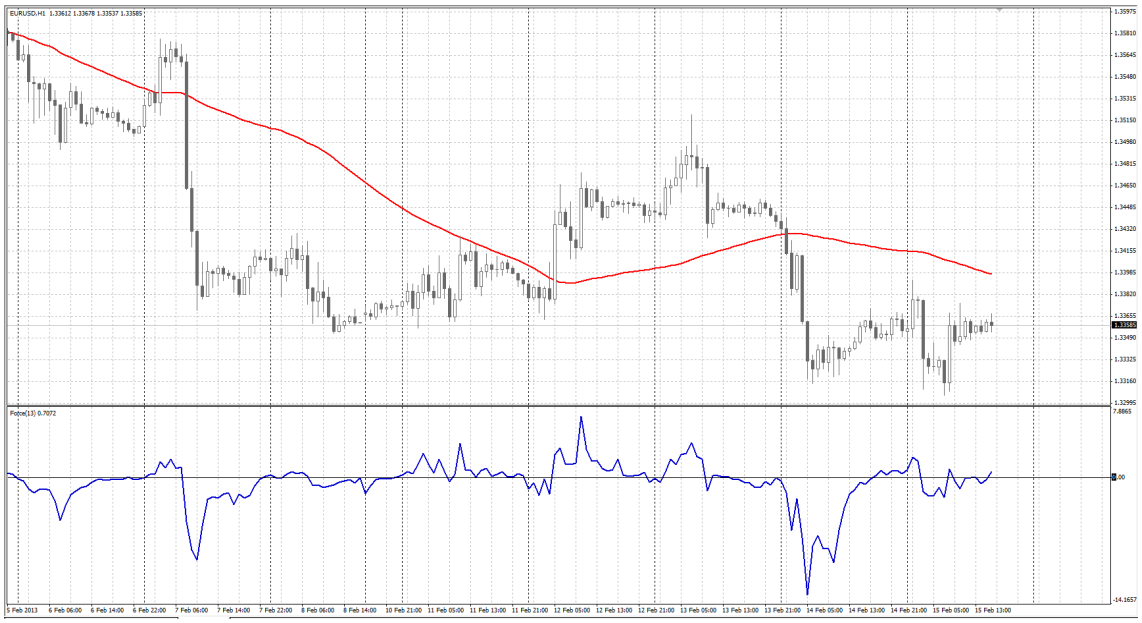


Figure A.1: A trading terminal screenshot with TI of two types. The continuous red line demonstrates the example of a type 1 TI (crosses the price — grey candles), and the continuous blue line that demonstrates an example of a type 0 TI (has positive and negative values - crosses 0)

Appendix B

Scores distributions

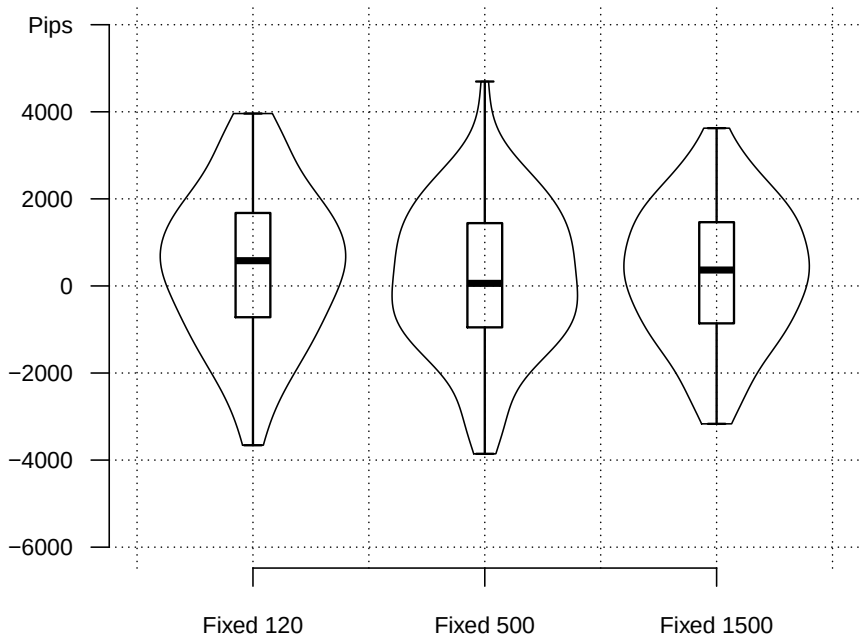


Figure B.1: StepEv 120 versus StepEv 500 versus StepEv 1500 (N_v 0, p_{sl} 0.2).

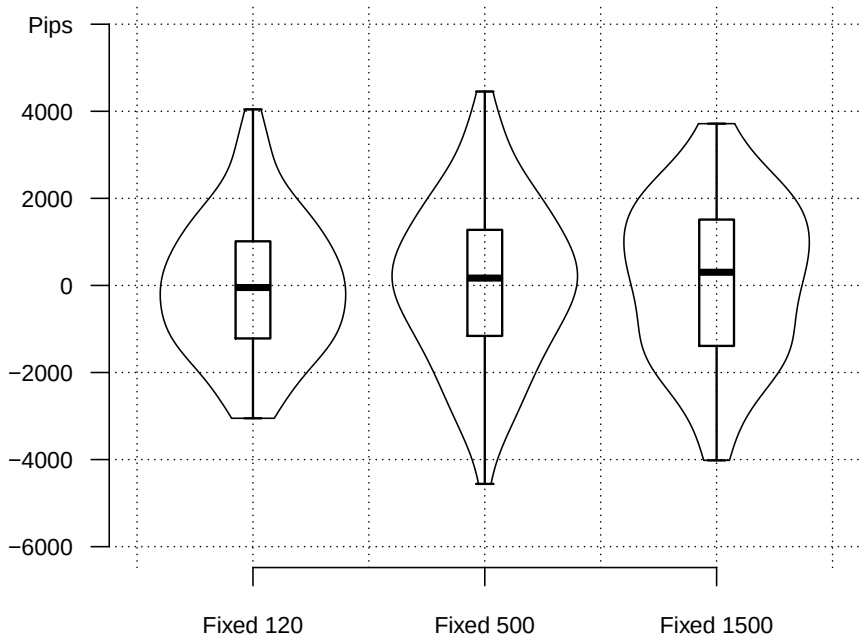


Figure B.2: StepEv 120 versus StepEv 500 versus StepEv 1500 (N_v 0, p_{sl} 0).

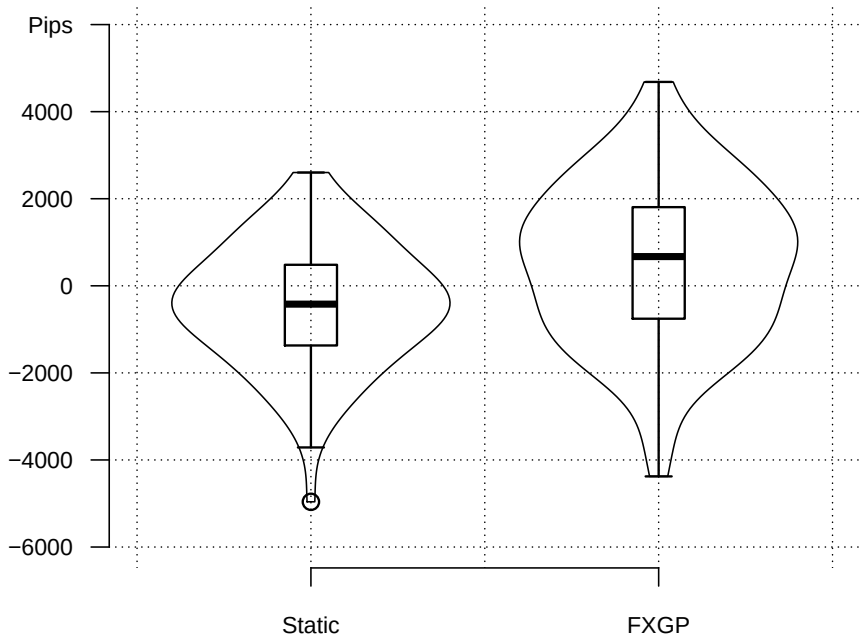


Figure B.3: Static versus FXGP (N_v 0, p_{sl} 0.2).

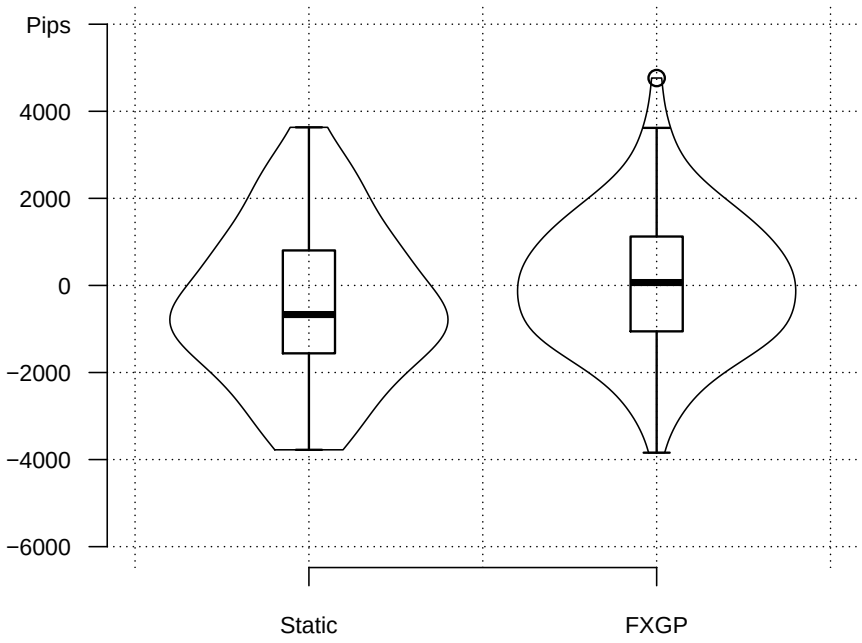


Figure B.4: Static versus FXGP (N_v 0, p_{sl} 0).

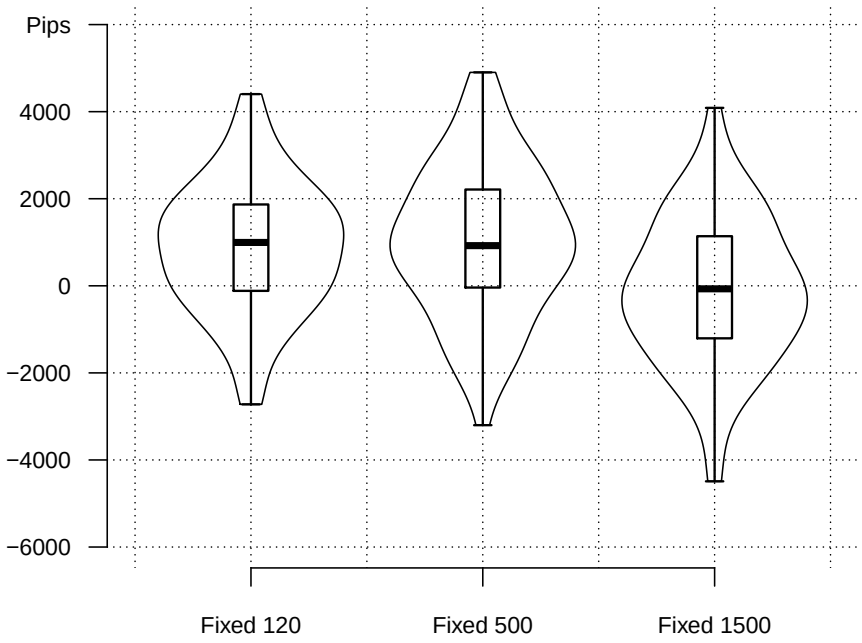


Figure B.5: StepEv 120 versus StepEv 500 versus StepEv 1500 (N_v 500, p_{sl} 0.2).

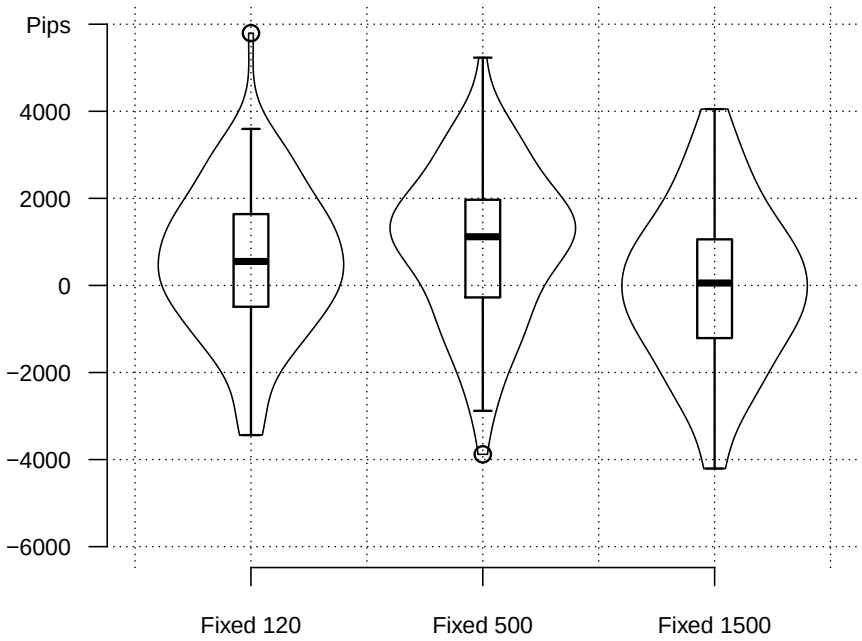


Figure B.6: StepEv 120 versus StepEv 500 versus StepEv 1500 (N_v 500, p_{sl} 0).

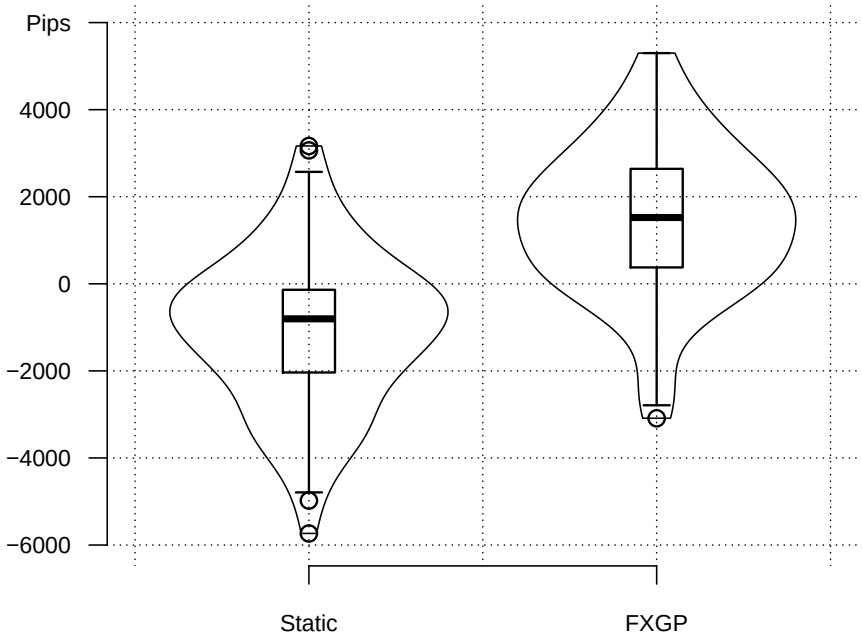


Figure B.7: Static versus FXGP (N_v 500, p_{sl} 0.2).

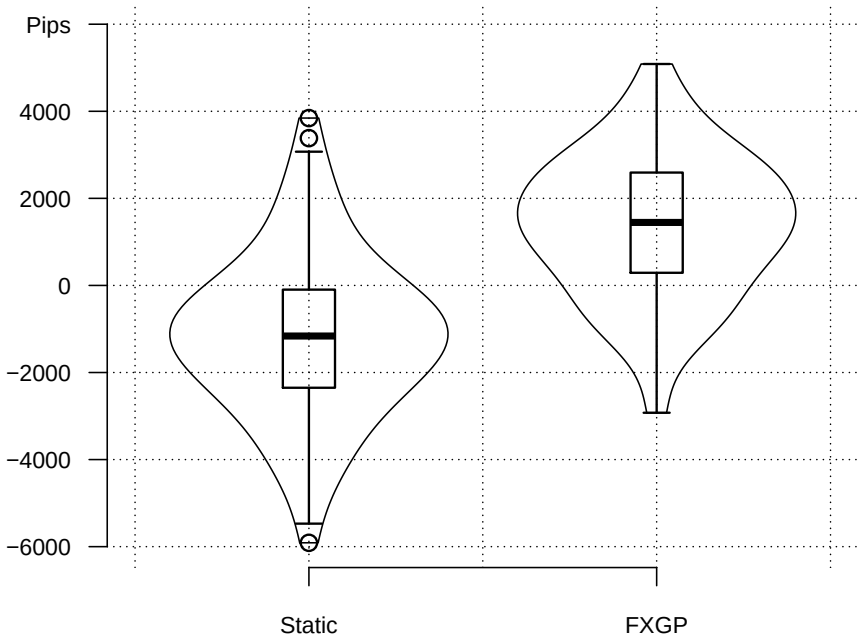


Figure B.8: Static versus FXGP (N_v 500, p_{sl} 0).

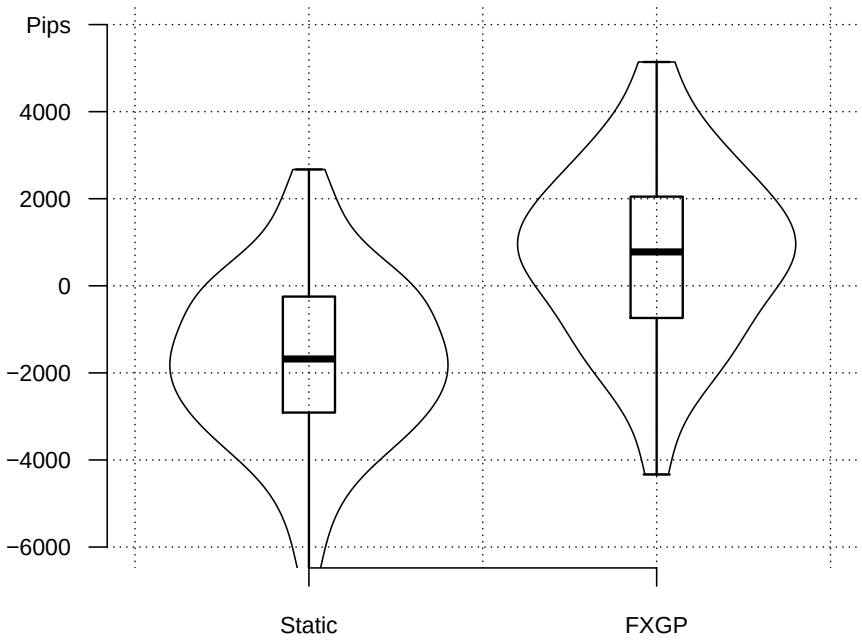


Figure B.9: Static versus FXGP (N_v 500, p_{sl} n/a).

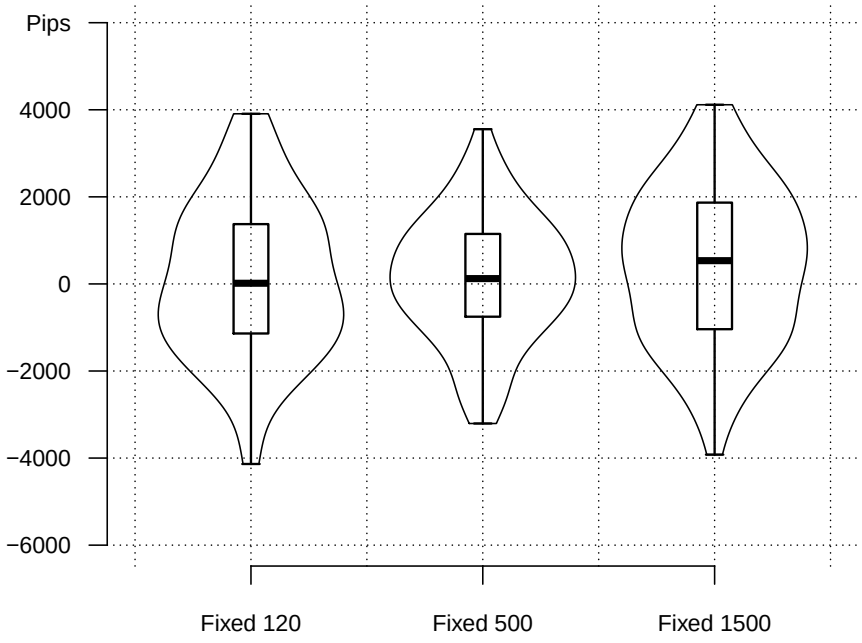


Figure B.10: ContEv 120 versus ContEv 500 versus ContEv 1500 (N_v 0, p_{sl} 0.2).

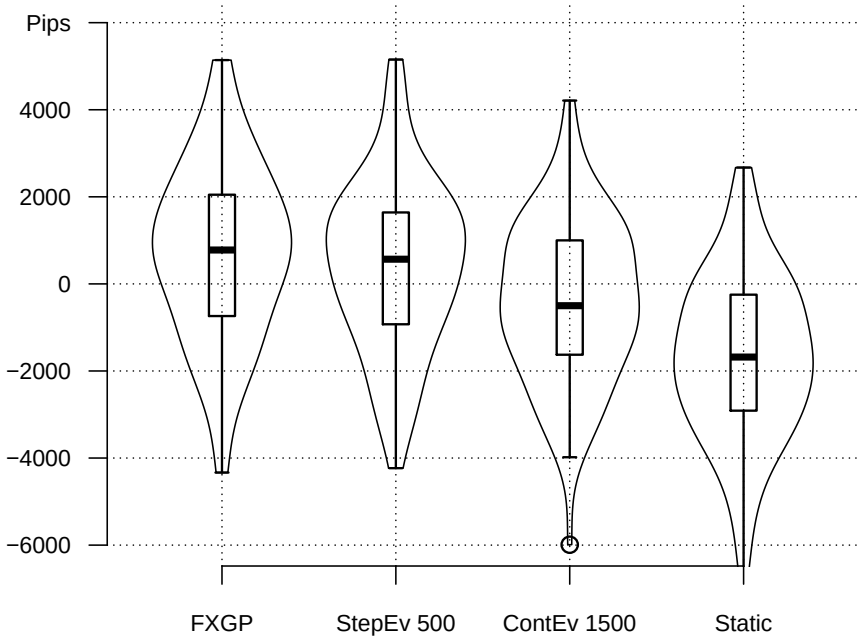


Figure B.11: FXGP versus StepEv 500 versus ContEv 1500 versus Static without S/L orders (marked by ‡ in Table 4.2.)

Bibliography

[1] M. Brameier and W. Banzhaf. *Linear Genetic Programming*. Springer, 2007.

[2] G. Cheng. *7 Winning Strategies for Trading Forex: Real and Actionable Techniques for Profiting from the Currency Markets*. Harriman House Ltd, 2007.

[3] I. Dempsey, M. O'Neill, and Brabazon A. *Foundations in Grammatical Evolution for Dynamic Environments*, volume 194 of *Studies in Computational Intelligence*. Springer, 2009.

[4] M.H. Dempster, T.W. Payne, Y. Romahi, and G.P. Thompson. Computational learning techniques for intraday FX trading using popular technical indicators. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 12(4):744–54, January 2001.

[5] J. A. Doucette, A. R. McIntyre, P. Lichodzijewski, and M. I. Heywood. Symbiotic coevolutionary genetic programming. *Genetic Programming and Evolvable Machines*, 13(1):71–101, 2012.

[6] C. Dunis, A. Harris, S. Leong, and P. Nacaskul. Optimizing intraday trading models with genetic algorithms. *Neural Network World*, (1992):1–22, 1999.

[7] P Fernández-Blanco. Technical market indicators optimization using evolutionary algorithms. *Genetic and Evolutionary Computation Conference - GECCO*, pages 1851–1857, 2008.

[8] J Gama, R Sebastião, and PP Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 2012.

[9] A Ghosh, S Tsutsui, and H Tanaka. Function optimization in nonstationary environment using steady state genetic algorithms with aging of individuals. *International Conference on Evolutionary Computation*, 1998.

[10] FxPro Group, accessed Sept, 2012. `http://www.fxpro.com/trading/cfd/ecn/fx`.

[11] S Haykin. Adaptive filter theory (ise). 2003.

[12] A. Hirabayashi, C. Aranha, and H. Iba. Optimization of the trading rule in foreign exchange using genetic algorithm. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09*, page 1529, 2009.

[13] A. Hryshko and T. Downs. An implementation of genetic algorithms as a basis for a trading system on the foreign exchange market. In *IEEE Congress on Evolutionary Computation*, pages 1695–1701, 2003.

[14] A. Hryshko and T. Downs. System for foreign exchange trading using genetic algorithms and reinforcement learning. *International Journal of Systems Science*, 35(13-14):763–774, October 2004.

[15] H. Iba and C. C. Aranha. *Practical applications of evolutionary computation to financial engineering*, volume 11 of *Adaptation, Learning, and Optimization*. Springer, 2012.

[16] ICM Trade Capital Markets Ltd. Guide to online forex trading. 19 pages.

[17] Investopedia, accessed Sept, 2012. `http://www.investopedia.com/terms/t/two-percent-rule.asp#axzz2710QU8jR`.

[18] Investopedia, accessed Sept, 2012. `http://www.investopedia.com/terms/n/noise.asp#axzz27d0d2rid`.

[19] John R Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection v. 1 (Complex Adaptive Systems)*. MIT Press, 1993.

[20] P. Lichodzijewski and M. I. Heywood. Symbiosis, complexification and simplicity under GP. *ACM Genetic and Evolutionary Computation Conference*, 2010.

[21] K. Lien. *Day trading and swing trading the currency market : technical and fundamental strategies to profit from market moves*. John Wiley and Sons, Inc., 2008.

[22] A. Loginov and M. I. Heywood. On the impact of streaming interface heuristics on GP trading agents: An FX benchmarking study. In *ACM Genetic and Evolutionary Computation Conference, under review*, 2013.

[23] A. Loginov and M. I. Heywood. On the utility of trading criteria based retraining in forex markets. In *Evo Applications*, 2013.

[24] S Mallat. *A wavelet tour of signal processing*. Academic Press, 1999.

[25] Jared F. Martinez. *The 10 Essentials of Forex Trading: The Rules for Turning Patterns into profit*.

[26] E Mayr. *What evolution is*. Basic Books, 2002.

[27] Melanie Mitchell. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. MIT Press, 1998.

[28] I. V. Morozov and R. R. Fatkhullin. Forex: from simple to complex, 2004. Teletrade Ltd.

[29] P.B. Myszkowski and A. Bicz. Evolutionary algorithm in Forex trade strategy generation. *International Multiconference on Computer Science and Information Technology - IMCSIT*, pages 81–88, 2010.

[30] A. Passamonte. Six facts that give forex traders an edge. *Forex Journal*, 2011. http://www.forexjournal.com/fx-education/forex-trading/12125-six-facts-that-give-forex-traders-an-edge.html.

[31] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A Field Guide to Genetic Programming*. Lulu, 2008.

[32] D Saad. *On-line learning in neural networks.* Cambridge University Press, 2009.

[33] Bank For International Settlement. Triennial central bank survey of foreign exchange and OTC derivatives market activity - preliminary global results, April 2010. http://www.bis.org/press/p100901.htm.

[34] HA Simon. *The sciences of the artificial.* The MIT Press, 2 edition, 1989.

[35] The Spiess McGlade Team. 2012 mutual fund recommended list, 2012. http://www.managedmoneyreporter.com.

[36] E.P.K. Tsang and J. Li. Eddie for financial forecasting. *In: Chen, S.H. (ed.) Genetic Algorithms and Genetic Programming in Computational Finance, pp. 161174. Kluwer Academic Publishers*, pages 161–174, 2002.

[37] C. Tuite, A. Agapitos, M. O'Neill, and A. Brabazon. A preliminary investigation of overfitting in evolutionary driven model induction. In *EvoApplications*, volume 6625 of *LNCS*, pages 120–130, 2011.

[38] N. Wagner, Z. Michalewicz, M. Khouja, and R. R. McGregor. Time series forecasting for dynamic environments: The dyfor genetic program model. *IEEE Transactions on Evolutionary Computation*, 11(4):433–452, 2007.

[39] L. D. Whitley. Fundamental Principles of Deception in Genetic Search. *Foundations of genetic algorithms*, 1(1980):221–241, 1991.

[40] Garnett Wilson and Wolfgang Banzhaf. Interday foreign exchange trading using linear genetic programming. *GECCO 10 Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1139–1146, 2010.