On the Impact of Streaming Interface Heuristics on GP Trading Agents: An FX Benchmarking Study

Alexander Loginov[1] and Malcolm I. Heywood[1]

[1]*Faculty of Computer Science, Dalhousie University, Halifax, NS. Canada*

Article originally appears at GECCO under ACM copyright 2013
`http://dl.acm.org/citation.cfm?doid=2463372.2463522`

Abstract

Most research into frameworks for evolving trading agents emphasize aspects associated with the evolution of technical indicators and decision trees / rules. One of the factors that drives the development of such frameworks is the non-stationary, streaming nature of the task. However, it is the heuristics used to interface the evolutionary framework to the streaming data which potentially have most impact on the quality of the resulting trading agents. We demonstrate that including a validation partition has a significant impact on determining the overall success of the trading agents. Moreover, rather than conduct evolution on a continuous basis, only retraining when changes in trading quality are detected also yields significant advantages. Neither of these heuristics are widely recognized by research in evolving trading agent frameworks, although both are relatively easy to add to current frameworks. Benchmarking over a 3 year period of the EURUSD foreign exchange supports these findings.

1 Introduction

As per any application of machine learning in general to a task currently performed by humans, there are potentially a wide range of application specific decisions that play a significant role in determining the success of the resulting application. From the perspective of foreign exchange (FX) currency trading the initial data is described in terms of a sequence of trades. The trades are summarized with respect to a trading interval, typically in terms of the following attributes: opening, closing, highest and lowest price. Traders will construct rules to summarize a trading strategy. FX trading frequently make use of Stop–Loss or "S/L" thresholds in which an order to buy or sell is defined in terms of the market reaching a specific price. Moreover, trading rules are typically designed in terms of a combination of technical indicators (TI). The TI attempts to describe properties of the market that are useful for decision making in general. Such TI are estimated over a historical window of previous data relative to the current decision point, t.

Most research has concentrated on the application of evolutionary computation (EC) to the optimization of a priori technical indicators (TI) and decision trees (DT), Section 2. Conversely, less emphasis is placed on the contribution of parameters associated with interfacing the proposed EC to the application. For example, evolving trading agents in general assumes an online as opposed to offline application of EC. The data describing the trading environment is a continuous stream of a non-stationary nature. It is therefore not realistic to assume that solutions are evolved offline and then deployed. Instead, some heuristic needs to be adopted to either facilitate continuous evolution or detect changes in the stream and then retrain. In the case of continuous evolution a natural tradeoff between exploration and exploitation exists, whereas under change detection a poor choice of metric might result in missing important retraining events.

This research introduces a common framework for symbiotically coevolving TI and DT under representations appropriate for each task: hereafter FXGP (Section 3). A benchmarking study to quantify the significance of parameterizations pertinent to interfacing the FXGP trading agents to the stream represents the central contribution of this research. In particular, we are interested in the relative contribution of continuous coevolution versus change detection in supporting the development of effective trading agents. Secondary parameters are also considered such as support for a validation partition *during* application to the stream and the significance of evolving thresholds specific to the FX task (Section 4). We conclude that including a validation partition is of paramount importance. Change detection combined with population reinitialization is recommended to explicitly retrain trading agents as this appears to be more effective at avoiding carrying over unwanted model bias when a stream is non-stationary. Conversely, the current norm is to continuously evolve a population of trading agents without reinitialization. This is shown to result in significant bias, thus lack the capacity to react appropriately to the non-stationary nature of the task.

2 Background

Previous research has been reported with respect to the evolution of technical indicators (TI) alone e.g., [5]. 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 [14]. More recently separate TI were evolved for buying and selling [6]. Several authors have evolved trading rules or decision trees (DT) relative to a set of a priori selected TI e.g., [3], [7]. One of the most well known instances of GP in this context is 'EDDIE' [12]; 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., [8]. 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 coevolved in two independent populations [11].

In general, authors have placed a lot of effort on designing fitness functions to penalize losses as well as reward profit [2], [8]. 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 [13]. 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 [8] 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 using the content of the current population. Questions potentially arise regarding the number of generations to perform [2]. This is related to the degree of non-stationary (or appropriateness of any model bias) associated with the transition between training and trading periods. Indeed other authors have adopted training at *every* sliding window location [15]. In this work we compare reinitialization and retraining w.r.t. training criteria versus the typically assumed approach of continuous evolution.

3 FXGP Algorithm

Following the preamble established by the Introduction and Background (Sections 1 and 2) we will assume a framework capable of evolving both unique TI and DT. Specifically, the FXGP framework will be followed [11].[1] In the case of the TI, the desire is to support the evolution of properties pertinent to capturing temporal features e.g., parameterization of sliding windows over which basic statistics are estimated [14]; whereas the role of the DT is to combine an arbitrary number of uniquely parameterized TI into a trading rule. Thus, the DT assumes a comparatively straightforward representation: a sequence of conditional statements based on references to the TI population.

Naturally, it is not possible to establish the fitness of the TI population independently of the DT population. Indeed it is only at the DT population that a direct measure of fitness can be established. Hence, FXGP *coevolves* DT and TI populations together under a symbiotic partnership where such schemes have previously been demonstrated for evolving teams of programs in genetic programming [10, 4].

The second aspect of the task domain to be addressed – and focus of this research – is the non-stationary nature of trading. That is to say, a champion individual evolved relative to a historical partition of trading data will only be effective for some duration of the future trading interval. Previous work proposed to employ *trading criteria* to define when a retraining cycle should be called [11]. Thus, a historical period of N_t records is used to construct a TI–DT population of trading agents, and a sequential period of N_v records is used to validate the champion TI–DT individual (Figure 1). The champion individual is allowed to trade until the trading criteria indicates that the champion individual is no longer effective i.e., change detection. The entire content of the TI and DT populations is then replaced with a corresponding new population of randomly initialized individuals. Retraining would then

[1]This was previously demonstrated on three FX pairs [11]. However, no consideration was given to the impact of different schemes for interfacing with the stream, which is the focus of this research.

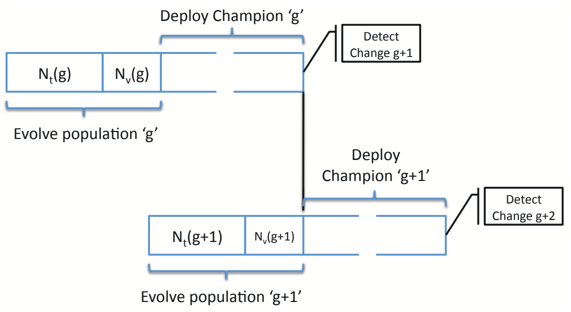


Figure 1: Identifying a new champion trading agent through change detection. g represents the count of independent cycles of evolution. Evolution is conducted relative to the most recent sample of training, $N_t(g)$ and validation data $N_v(g)$ from the stream.

Table 1: TI functions. Note the three forms of division

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	$R[x] \leftarrow R[x] \div R[y]$
Invert	$R[x] \leftarrow 1 \div R[x]$
Divide-by-2	$R[x] \leftarrow R[x] \div 2$

take place with training and validation partitions re-aligned with the point at which the trading criteria identified the failure (Figure 1).

3.1 Training

3.1.1 Representation

Individuals of the **TI population** assume a linear GP representation (e.g., [1]) with instruction set summarized by Table 1. Moreover, each TI has a header defining the basic TI properties: type, scale, period, shift (Table 2). The TI scale serves to detect useless TI and is checked after initialization. The TI initialization is repeated until the TI defines an outcome characterized by the scale type (see last paragraph of this sub-section).

TI programs assume a register level transfer language (Table 1). Thus, $R[x]$ denotes the content of register x, $R[y]$ denotes either: register y content; a price, or; a price m hours back in (relative) time. Register $R[0]$ is assumed to contain the TI result after executing a TI program. The MA type of TI is calculated as (1) whereas the WMA type of TI is calculated as (2), where V_j is a TI price value from the trading data.

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

Table 2: TI parameters. Estimated relative to the current hour t of trading.

Parameter	Description
TI type	Weighted Moving Average (WMA), Moving Average (MA) or Value
TI scale	Defines a TI that crosses 0 or TI that crosses price
Period n	Number n of hours in a price history to calculate MA or WMA
Shift m	Price m hours back

$$WMA_i = \frac{\sum_{j=1}^{n} \frac{V_j}{j+1}}{\sum_{k=1}^{n} \frac{1}{k+1}} \quad (2)$$

The **DT population** is initialized to a fixed size. A DT header declares the following information: DT *score* in pips[2], number of trades and a size of a Stop / Loss (S/L) order in pips. S/L is therefore an example of a task specific threshold which could potentially be evolved. The score and number of trades are initialized with 0; whereas the S/L can be initialized in one of two ways: 1) the S/L is initialized with a randomly selected value between the minimum (s_{min}) and maximum (s_{max}). 2) S/L is initialized with a maximum S/L size. The maximum and minimum S/L are currency pair specific and will be defined in Section 4. Likewise the benchmarking study will consider the degree of utility for each scheme as an experimental parameter.

A DT consists of a variable number of nodes. 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*

where X_i and Y_i can be 0, price or a TI. The *then* and *else* statements reference either: the next node or one of the trading signals: buy, sell or stay. Thus, a DT population is randomly generated with respect to X_i and Y_i scales; albeit under the following constraints:

1. if X_i is 0, then Y_i must also be a TI which crosses 0 and can not be a price or a TI which crosses the price, or;
2. 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.

Note 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 m 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. Thus, the DT population size is fixed whereas the TI population size may vary.

3.1.2 Fitness and Selection

The DT fitness is defined as a score in pips over the training records. 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. The number of trades is used to penalize the DT score for too high or low a frequency of trading. Moreover, if a DT generates only *buy* or *sell* signals its score is discarded and the DT targeted for replacement. The subset of DT individuals with lowest scores are explicitly identified for replacement cf., a breeder model of selection / replacement. Thus, a fixed percentage or *gap* size 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 TI component of a clone can be varied. Following DT selection, any TI individual that no longer receive a DT index are considered ineffective and are also deleted; thus another factor contributing to a variable size TI population.

3.1.3 Mutation

FXGP uses mutation to produce offspring. Only the size of S/L 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 (if enabled) mutates the size of 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}. A DT mutation is performed by applying one of the following actions: 1) Generate a new conditional function; 2) Increment / decrement shift parameter m; 3) Generate new X_i and Y_i; 4) Switch X_i and Y_i; 5) Switch content of *then* and *else* clauses; 6) Insert new *then* clause content, or; 7) 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: 1) TI type; 2) Period (n); 3) Shift (m); 4) Generate a new function; 5) Delete a function, or; 6) Insert a function, where function insertion or deletion is performed with respect to the

[2]'pips' or 'ticks' are the smallest price movement in a currency. In EURUSD, a move of 0.0001 is one pip.

TI program size limits (Table 3). Training stops when the specified number of generations (T_{max}) was reached or when the best score in the DT population plateaus for a fixed number of generations, t_{flt} (Table 3). Thereafter a validation cycle is initiated.

3.2 Validation

The validation process 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 $\alpha_v \times$*best score* it tests the DT. If DT is active the tested tree's counter is incremented. When all DT have been evaluated, FXGP checks the number of the tested DT. If it is greater than $P_{size} - P_{gap}$ then the DT (and referenced TI) with best score on validation assume the role of 'champion' and employed 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.

3.3 Trading and Retraining criteria

During FX trading, the following three trading quality criteria are monitored and used to trigger retraining from an entirely new population pair of TI–DT populations:

- The magnitude of a decline in account value as measured from peak to subsequent trough. This is generally referred to as the *drawdown* and characterized in terms of a minimum (d_{min}) and maximum (d_{max}) number of pips;
- A maximum number of consecutive *losses* (n_{loss}); and
- A maximum number of consecutive hours *without trading* activity (n_{nt}).

The basic goal is to characterize when the trading conditions are deemed to represent a 'significant' difference from that of the previous training period. When any quality criteria are exceeded, FXGP stops trading, reinitializes the TI and DT populations and restarts the training-and-validation cycle (Figure 1). Content of the training and validation partitions is taken relative to the point 't' at which the trading quality criteria interrupted trading. Once a new TI–DT champion is identified trading resumes at the point trading was interrupted.

4 Results

The following empirical study assumes FXGP as described in Section 3 as the underlying framework for coevolving TI–DT trading agents. Our interest is in the relative contribution of:

- Criteria based re-triggering of the retraining event (Section 3.3) versus the widely assumed case of continuously evolved trading agents (Section 4.1).
- Significance of including a validation partition. FXGP assumes that the champion individual is identified by a validation partition. However, this also implies that the most recent data is not available for training. We will consider the inclusion and non-inclusion of the validation partition.
- Degree of support for evolving market specific thresholds. In this case we consider two different forms of S/L thresholds (Section 3.1.1). Either a fixed threshold ($s_l = 0$) or thresholds evolved over a market specific spread interval ($s_l = 0.2$).

In the following we develop the framework for continuous re-evolution and then establish the common parameterization assumed throughout the benchmarking study. The specifics of the data source are then characterized. The results of the benchmarking study are then detailed. In all cases we are interested in performance across multiple runs; thus we report the distribution of results rather than a single best case. This latter point is particularly important because it is not known at the start of trading how effective a strategy might be.

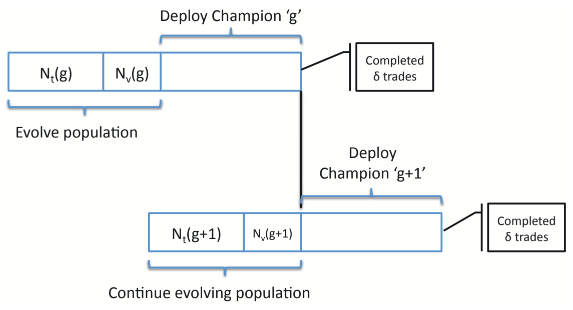


Figure 2: Identifying a new champion trading agent through rolling window. g represents the count of cycles of evolution relative to population content at cycle $g-1$. Trading is performed for a fixed number of cycles (δ) before a new cycle of evolution is performed.

4.1 Framework for continuous re-evolution

One of the unique characteristics of FXGP is that retraining is only ever performed when a negative change in trading behaviour is detected (Section 3.3). Conversely, the norm is to perform some form of continuous evolution e.g., [2], [8]. With this in mind, we assume the following framework for a 'rolling window' approach to evolution [8]. The TI–DT individuals are evolved as per FXGP with respect to training and validation partitions N_t and N_v respectively. Trading is then performed for a fixed duration (δ) using the champion trading agent (Figure 2). The caveat to this is that in the case of an open trade by the end of the period, trading is allowed to continue until all open trades are closed, resulting in potential variation in the trading period. After the δ trading period has elapsed, then the training and validation partitions are realigned with the end of the trading period (Figure 2) and the TI–DT populations are either:

1. Utilized as if evolution progressed from the last generation. Any new content assumes the bias established by the pervious cycle of evolution. This scheme reflects the intent of recent practice (e.g., [2], [8]) and hereafter will be referred 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 3.2 to deploy a validation partition. Determining the specific source of this remains a topic for future research;

2. 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 and is similar to the approach adopted in [15]. 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.

4.2 Source Data

The historical rates for EURUSD (1 hour resolution) were obtained from the Forex Historical Data repository.[3] Each 24 hour period therefore typically consists of 24 samples. Sampling at the rate of one hour intervals is conducted to reduce the impact of "trading noise" [9]. We concentrate on the most widely traded currency pair EURO-to-USD for the three year period July 2009 to November 2012 (or $\approx 17,860$ hrs). Data is described in terms of the following fields: Pair, Date, Time, bid price Open (Open), bid price Low (Low), bid price High (High) and bid price Close (Close).

4.3 Parameterization

Parameters for FXGP are separated into three sets: 1) generic parameters for sizing population, generations and frequency of operator application, 2) those for validation (Section 3.2), and 3) those for characterizing change detection (Section 3.3). All three sets of parameters are summarized in Table 3. Naturally, no claims are made

[3] `http://fxhistoricaldata.com/EURUSD/`

Table 3: FXGP parameters

Parameter	Value
Training period (N_t)	1,000
Validation period (N_v)	500
Probability of TI mutation (p_{ti})	0.5
Max. training generations (T_{max})	1,000
DT population size (P_{size})	100
# DT replaced per generation (P_{gap})	25
Max. nodes (instr.) per DT (TI)	6
Max (Min) Stop/Loss (s_{max}, s_{min})	$(5, 100)$
Validation criteria (Section 3.2)	
Proportion with feasible trade (α_v)	0.95
Plateau detection threshold (t_{flt})	200
Retraining criteria (Section 3.3)	
Max (Min) Drawdown in pips	$(200, 400)$
Max. consecutive losses (n_{loss})	3
Max. consecutive inactive trades (n_{nt})	72

regarding their optimization. Several parameters are then associated with the case of continuous and stepwise evolution of trading agents, Section 4.1. Specifically, the size of the fixed length trading period is varied over three independent experiments: $\delta \in \{120, 500, 1500\}$ or approximately 1 week, 1 month, 3 months. This reflects the lack of support for being able to dynamically trigger retraining. Hereafter these will be identified as *ContEv or StepEv xxx*, where *xxx* denotes the trading period duration. Naturally, all trading agents assume common $N_t = 1,000$ and $N_v \in \{0, 500\}$ parameterizations. Finally, we will also have a baseline TI–DT model in which no retraining is performed after the initial cycle of training. Hereafter this will be referred to as *Static*.

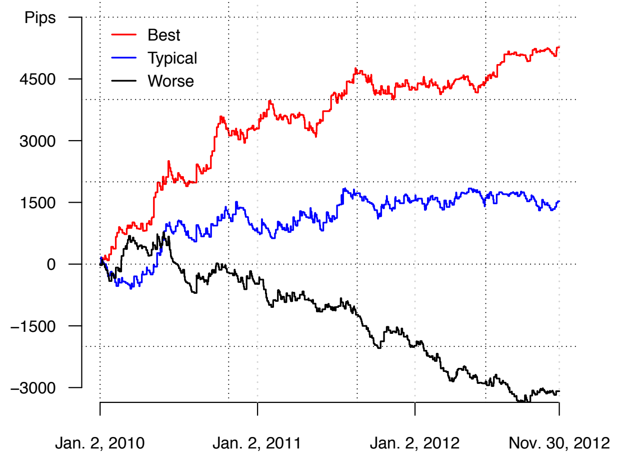
In total there are four frameworks: FXGP, StepEv (stepwise evolution), ContEv (continuous evolution; representing current practice) and Static. FXGP follows the single parameterization of Table 3. StepEv (or ContEv) replace and retrain (or update relative to current population content) after a fixed length trading period, δ, hence the "retraining criteria" of Section 3.3 is replaced by the approach of Section 4.1. As indicated above three different durations for the trading period are considered relative to fixed size training and validation periods. ContEv will only be deployed in a single configuration, on account of the approach to deploying validation partitions from Section 3.2 not selecting champion individuals. However, this also represents the most frequent scenario in which ContEv is deployed e.g., [2], [8]. For FXGP and StepEv we enable and disable validation ($N_v \in \{0, 500\}$) or support evolution of S/L thresholds or not. In the latter case, S/L is either fixed at 100 pips or evolved. This results in 23 experiments, each run over 100 initializations simulating trading activity for the EURUSD currency pair over the *trading period* from January 2, 2010 to November 30, 2012.[4]

4.4 Benchmarking results

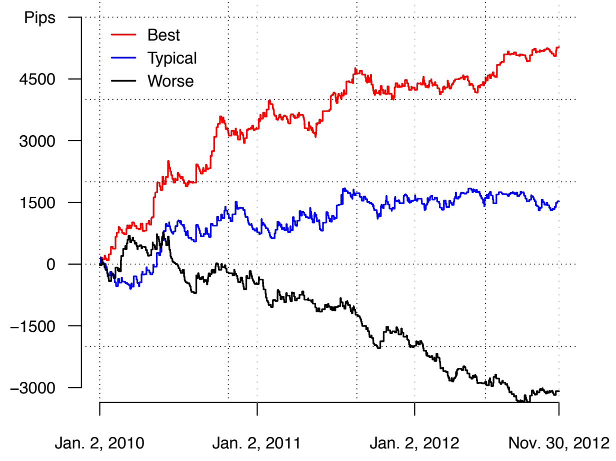
Table 4 provides a high level summary of all 23 experiments. The table order reflects the median score in pips over the trading period as collected over all 100 runs. Naturally, there is a close correlation between median score and the percentage of runs that were profitable. Several general trends are apparent:

- Retaining population content between cycles of retraining (ContEv) appears to be detrimental to providing new trading agents capable of reacting to the next cycle of trading. However, neither ContEv nor StepEv appear to match the effectiveness of FXGP (dynamically identifying retraining intervals).
- Both FXGP and StepEv (the latter for trading periods of 1 to 4 weeks) have a strong preference for including the validation partition. Interestingly for the longer trading period under StepEv ($\delta = 1500$) there was no benefit in including a validation partition.
- Enabling the evolution of Stop / Loss thresholds did not result in a general pattern of preference for or against its inclusion. However, it does appear to be useful w.r.t. ordering FXGP.

[4]Note that the first population is evolved using data starting from July 2009 in order to support a Jan 2010 start to trading.



(a) pips



(b) USD

Figure 3: Trading profiles for (a) pips and (b) USD using best configured trading agent: FXGP with validation partition and S/L.

- In the case of the lengths of trading considered with StepEv and ContEv – $\delta \in \{120, 500, 1500\}$ – the longest trading period was most effective with ContEv, but least effective with StepEv. Further research would be necessary to obtain a clearer picture of the dynamics behind this.
- It is possible that any configuration can result in profitable (unprofitable) runs (see 'min' and 'max' columns of Table 4). However, in most cases at least 25% of runs are unprofitable (1st quartile). Given that it is not a priori possible to determine which runs will be profitable over the investment period, this means that it is very important to characterize performance over multiple runs. If only best case runs are considered, then even the static configuration is 'profitable'.
- Training once at the beginning of the three year period and then assuming that the resulting champion individual will be effective clearly does not work. This illustrates that the FX trading task is essentially non-stationary. Naturally, the use of a validation partition is irrelevant as 1 month of validation data is not sufficient for characterizing the following three years of trading when the underlying process is non-stationary.

In the case of the best configured framework – FXGP with validation partition and non-zero S/L threshold – we also plot the trading curves in pips and USD over the total trading period (Figure 3). The USD account assumes an initial balance of $100,000 and a leverage of 1:100. One pip is worth 0.0001 USD and buying or selling actions are limited to 2% of the account balance.[5] These curves reflect the corresponding best, worst and median scores from Table 4. By way of comparison, investing in a mutual fund over exactly the same period could result in yearly rate of return of 3.95% on the same initial investment ($100,000); or a gain of $11,947 by the end of the 3 year period.[6] However, to obtain such a result the mutual fund was selected post investment i.e., once the three year investment is complete; whereas most mutual funds failed to match this level of return. Conversely, even the typical performance of FXGP in Figure 3 better this return.

Figure 4 illustrates the distribution in the ultimate trading outcome for the best configuration from Table 4 i.e., FXGP, StepEv, ContEv and Static as indicated by the †symbol. The FXGP cases (with validation) were the only solutions with 1st, 2nd (median) and 3rd quartile investments that were non-negative. Thus, at least 75% of the runs returned profitable training strategies. Conversely, trading agents evolved under any other approach failed to reach this degree of repeatability. Moreover, this was not achieved at a loss of median or 3rd quartile profitability. Table 5 confirms the degree of independence in the distributions as a student T-test. FXGP is in the worst case independent at the 95 percentile (w.r.t. StepEv) and more than a 99 percentile in the case of ContEv (as typically used when deploying evolutionary trading agents) and Static.

[5] `http://www.investopedia.com/articles/forex/07/forex_leverage.asp`

[6] RBC Global asset management. `https://services.rbcgam.com/portfolio-tools/public/investment-performance/`

Table 4: Result ranking w.r.t. average pips. †cases illustrated in Figure 4.

Algorithm	Validation (N_v) (500 / 0)	Stop/Loss (p_{sl}) (0.2 / 0)	% Profitable runs	Score (pips) min	1st quartile	median	3rd quartile	max
FXGP†	500	0.2	82	-3087.9	383.375	1522.5	2639.525	5298.6
FXGP	500	0	79	-2924.2	296.425	1447	2580.9	5086.1
StepEv 500†	500	0	73	-3879.1	-234.85	1118.3	1947.15	5231.8
StepEv 120	500	0.2	73	-2722.9	-89.3	995.85	1862.975	4403.1
StepEv 500	500	0.2	73	-3200.9	-33.55	922.75	2167.825	4902.1
FXGP	0	0.2	60	-4377.5	-746.05	671.4	1792.35	4682.8
StepEv 120	0	0.2	63	-3659.0	-717.775	580.35	1663.275	3959.7
StepEv 120	500	0	66	-3438.4	-433.05	550.45	1630.375	5795.5
ContEv 1500†	0	0.2	61	-3920.9	-1020.875	534.55	1852.025	4115.7
StepEv 1500	0	0.2	58	-3167.4	-852.825	365.55	1460.3	3624.0
StepEv 1500	0	0	55	-4017.5	-1373.3	302.65	1482.425	3715.8
StepEv 500	0	0	53	-4558.4	-1150.65	169	1275.2	4454.3
ContEv 500	0	0.2	53	-3205.8	-739.5	122.45	1136.85	3553.8
FXGP	0	0	53	-3841.7	-1029.8	66.65	1123.15	4763.9
StepEv 500	0	0.2	52	-3855.6	-938.95	59.2	1416.925	4695.4
StepEv 1500	500	0	50	-4205.9	-1205.95	55.35	1019.05	4049.9
ContEv 120	0	0.2	52	-4135.8	-1099.925	15.3	1372.625	3908.2
StepEv 120	0	0	49	-3051.7	-1194.75	-48.2	1013.55	4043.4
StepEv 1500	500	0.2	45	-4490.8	-1202.225	-68.95	1129.95	4086.9
Static†	0	0.2	38	-4960.9	-1358.375	-421.25	467.7	2602.1
Static	0	0	37	-3775.7	-1531.55	-668.3	780.65	3631.3
Static	500	0.2	23	-5733.7	-2031.125	-804.55	-144.1	3166.5
Static	500	0	23	-5911.1	-2347.125	-1161.7	-109.775	3844.3

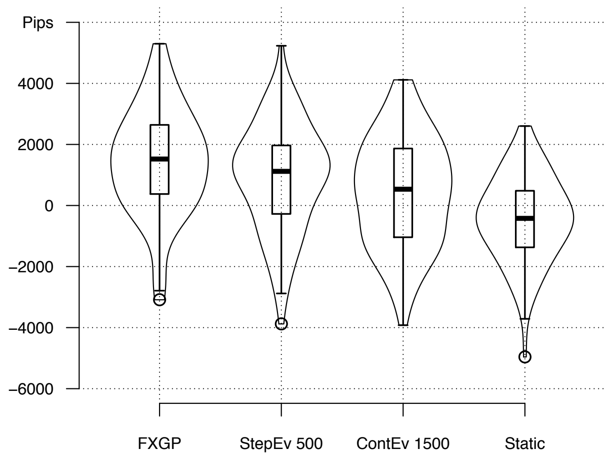


Figure 4: Comparison of 'pips' distribution associated with trading strategies from first ranked FXGP, StepEv, ContEv and Static parameterizations from Table 4. 100 runs per distribution.

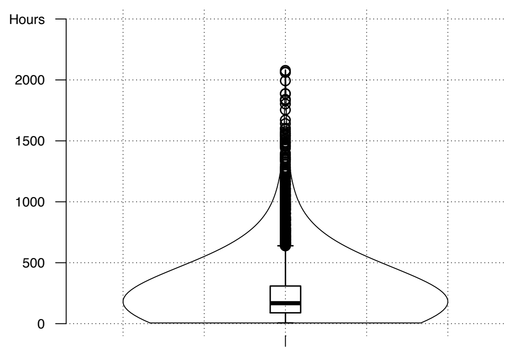


Figure 5: Distribution of intervals between re-triggering of FXGP training. Statistic collected across all 100 runs.

Table 5: *p-values* for pairwise Student T-test of FXGP versus StepEv, ContEv and Static identified by †in Table 4.

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

Finally, with respect to FXGP (with validation) we can also assess the frequency of training re-triggering. Figure 5 summarizes the trading interval between the re-triggering of training. The spread of the box plot (internal to the violin) reflects that variation in re-triggering does appear with a median of ≈ 200 hours, but a 1st / 3rd quartile spread of $[+140, -80]$; thus indicating that significant tuning of the retraining period is occurring.

5 Conclusion

A benchmarking study is performed to assess the relative value of:

- Selecting a champion trading individual from an independent validation sample of data before deploying as a trader;
- Dynamic re-triggering versus continuous evolution of trading agents;
- Incorporating the evolution of task specific thresholds.

Validation of traders before deployment appears to be of central significance. A previous study pertaining to the use of early stopping criterion in financial applications [13] was not performed relative to streaming data (batch offline learning relative to static training and test partitions was employed). However, the authors did indicate that they believed that care was necessary when determining the criterion for applying early stopping. In this research, the very poor performance of the trader evolved under offline 'Static' training conditions verifies that the task was indeed non-stationary. It was also apparent that using trading criteria to re-trigger training was more effective than 'fixing' retraining at the end of each trading period as in continuous or stepwise evolution. Current practice is to assume the case of training on a continuous basis at the end of each trading period without the use of validation e.g., [2], [8]. This is certainly better than assuming 'static' offline deployment, but appears to be less effective than enforcing a fixed rate of *retraining* from an entirely reinitialized population (StepEv). Conversely, retraining at every sliding window location has been considered [15], but naturally incurs a computational overhead. Introducing retraining criteria provides a potential middle ground.

Part of the reason we see for continuous evolution to fare so badly in this context is that diversity is being compromised. In the monograph of Dempsey *et al.* [2], Chapter 2, various factors are identified as being of particular importance when constructing frameworks to operate on non-stationary domains. One of these was the maintenance of diversity. However, when evolution is performed on a continuous basis (ContEv), there is a natural bias towards exploitation rather than exploration. We do not see continuous evolution as fundamentally limited in this respect, but it certainly appears that the trading domain requires more emphasis to be placed on maintaining diversity than currently appears to be the case. Both of the frameworks that explicitly reset population content before retraining (StepEv and FXGP) avoided this pathology. In summary, rather than assume a specific interface between EC and trading data, the authors suggest that several schemes should be considered in which different exploration / exploitation tradeoffs are explicitly represented.

Acknowledgements

The authors acknowledge support from NSERC and MITACS grant programs.

References

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

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

[3] M. A. Dempster and C. M. Jones. A real-time adaptive trading system using genetic programming. *Quantitative Finance*, 1:397–413, 2001.

[4] 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.

[5] P. Fernandez-Blanco, D. Bodas-Sagi, F. Soltero, and J. Hidalgo. Technical market indicators optimization using evolutionary algorithms. In *ACM Conference Companion on Genetic and Evolutionary Computation*, pages 1851–1858, 2008.

[6] A. Hirabayashi, C. Aranha, and H. Iba. Optimization of the trading rule in foreign exchange using genetic algorithm. In *ACM Conference on Genetic and Evolutionary Computation*, pages 1529–1536, 2009.

[7] 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.

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

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

[10] P. Lichodzijewski and M. I. Heywood. Symbiosis, complexification and simplicity under GP. In *ACM Genetic and Evolutionary Computation Conference*, pages 853–860, 2010.

[11] A. Loginov and M. I. Heywood. On the utility of trading criteria based retraining in Forex markets. In *EvoApplications*, LNCS, to appear, 2013.

[12] E. P. K. Tsang and J. Li. Eddie for financial forecasting. In S. H. Chen, editor, *Genetic Algorithms and Genetic Programming in Computational Finance*, pages 161–174. Kluwer Academic, 2002.

[13] C. Tuite, A. Agapitos, M. O'Neill, and A. Brabazon. A preliminary investigation of overfitting in evolutionary driven model induction: Implications for financial trading. In *EvoApplications: Part II*, volume 6625 of *LNCS*, pages 120–130, 2011.

[14] 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.

[15] G. Wilson and W. Banzhaf. Interday and intraday stock trading using PAM GP and linear GP. In A. Brabazon, O'Neill, and D. G. Maringer, editors, *Natural Computing in Computational Finance 3*, volume 293 of *SCI*, pages 191–212. Springer, 2010.