On evolving multi-agent FX traders

Alexander Loginov[1] and Malcolm Heywood[1]

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

Article originally appears at EvoAPPS'14 under Springer copyright 2014
http://link.springer.com/chapter/10.1007\%2F978-3-662-45523-4_17

Abstract

Current frameworks for identifying trading agents using machine learning are able to simultaneously address the characterization of both technical indicator and decision tree. Moreover, multi-agent frameworks have also been proposed with the goal of improving the reliability and trust in the agent policy identified. Such advances need weighing against the computational overhead of assuming such flexibility. In this work a framework for evolutionary multi-agent trading is introduced and systematically benchmarked for FX currency trading; including the impact of FX trading spread. It is demonstrated that simplifications can be made to the 'base' trading agent that do not impact on the quality of solutions, but provide considerable computational speedups. The resulting evolutionary multi-agent architecture is demonstrated to provide significant benefits to the profitability and improve the reliability with which profitable policies are returned.

1 Introduction

Machine learning (ML) has had a widespread impact on the automatic identification of trading agents for identifying profitable trading strategies under stock or currency markets. From the perspective of a generic process, multiple factors should be considered. For example, technical indicators (TI) are used to provide temporal features from which a decision tree (DT) defines the training strategy (e.g., buy–stay–sell). Although the TI might be designed independently before a DT is constructed – much in the manner that attribute selection might be performed independently of classifier construction – the quality of the resulting trading strategy will be dependent on the quality of the initial set of TI. This sequential dependence has lead authors to adopt various strategies in which: 1) as wide a set of TI are initially included as possible after which the DT selects the most appropriate. For example, [3] used genetic programming (GP) to define the DT and inso doing noted that combinations of TI lead to better currency trading strategies. Other authors report similar findings using different ML paradigms e.g., [1]; 2) mechanisms are pursued that permit temporal feature construction at the same time as identifying a suitable DT. For example, the coevolutionary approach suggested by [8, 7] evolves two populations simultaneously representing TI and DT respectively.

Naturally, the design of TI has a considerable impact on how the trading data is 'interfaced' to the trading data. Thus, not only the type of TI, but the parameterization of the TI need to be considered [5]. Moreover, trading data is non-stationary, thus an agent strategy that is appropriate for one period of trading will become unprofitable under a future period. This has lead to the adoption of various schemes for re-training or incremental evolution e.g., [2, 9, 7].

Finally, we note that it has been known for a while that in the general machine learning setting, stronger models for regression or classification result when multiple models are combined in an 'ensemble' e.g., [6, 10]. Indeed, boosting has been reported for identifying multiple DT in the case of a 'multi-level' framework for stock trading [1].[1]

It is in this last respect that this paper is focused. Caveats that make the application of ensemble methods (cf., multi-population architectures) challenging under a trading agent scenario might include: 1) computational overhead of constructing multiple solutions; 2) non-stationary nature of the task implies that ensemble content is likely to (at best) go stale or (at worst) over-learn, and 3) how to recombine multiple GP solutions (say, one from each population) into a single cohesive solution.

[1]Such a scheme does not naturally carry over to the currency trading scenario investigated here on account of the DT being used for predicting the one-step-ahead return relative to a sample of β-portfolio of stocks.

With this in mind, we assume the general framework of FXGP [8, 7] for coevolving TI and DT (Section 2) and concentrate on assessing to what degree combining multiple DT from different populations has on the quality of the resulting agent strategy. The computational overhead of maintaining multiple populations is addressed by adopting an approach closer to the 'weak learner' methodology in which we reduce the functionality in the TI and DT, resulting in a threefold speedup in the time to evolve a single population. The non-stationary nature of the task is addressed through the use of the behavioural criteria for triggering re-training, as in the original FXGP framework. In the case of combining solutions from each population, an approach to voting is adopted which enables us to avoid the need to maintain a large number of parallel populations i.e., a computational overhead for real-time operation.

2 The FXGP Algorithm Overview

The original FXGP framework proposed a symbiotic framework in which a decision tree (DT) and technical indicator (TI) population are symbiotically coevolved [8, 7]. Adopting such a framework enables FXGP to:

- assume a representation appropriate to each task. Specifically, in the case of this work the DT takes the form of a sequence of conditional statements with typing constraints between first and second arguments. A single DT may then index multiple individuals from the TI population. However, the TI takes the form of simple linear GP expressions in which different types of TI functionality are supported.
- TI are evolved to be specific to the task as opposed to relying on prior TI definitions. In this case the insight guiding this is that TI, as typically deployed, often represent combinations of more 'elemental' TI e.g., moving average, weighted moving average or specific samples of trading data.

In addition FXGP assumed an interface to the (stream) trading data in which re-training was triggered by a set of trading criteria. That is to say, after an initial period of training, a champion individual is identified (validation) and deployed for trading until trading criteria flag a deterioration in trading performance. At this point a new DT–TI population is coevolved relative to data leading up to the point of failure. In the case of the FX task, such a scheme was demonstrated to be more effective than continuously evolving against the trading data [7].

3 Multi-agent FXGP

This section describes the new version of the algorithm in which teams of FXGP individuals suggest the trading action; hereafter FXGPT. As indicated in Section 1 we perceive several potential pathologies that could detract from realizing the benefits of pursuing a multi-agent / ensemble approach to currency trading. Section 3.1 will address simplifications that we introduce to reduce the computational footprint of the original FXGP framework; hereafter *simple* FXGP (*s*FXGP). FXGPT is then defined relative to *s*FXGP (Section 3.2).

3.1 *simple* FXGP

The original TI function contains seven instruction types [8], whereas analysis of the resulting solutions indicated that only three functions where typically employed (Table 1). In addition, operations were removed from the instruction set that typically resulted in intron behaviour. Thus, multiplication that produces a TI with the wrong scale, two division functions and a square root function that frequently resulted in illegal operations (e.g. division by zero or square root of negative value) were all dropped from the instruction set. Needless to say, this also removes instruction types that have a longer (computational) latency i.e., division and square root; thus an expected improvement to TI execution, where it is the evaluation of TI that account for the majority of CPU time.

Table 1: TI functions.

Function	Definition
Addition	$R[x] \leftarrow R[x] + R[y]$
Subtraction	$R[x] \leftarrow R[x] - R[y]$
Division	$R[x] \leftarrow R[x] \div 2$

Originally three types of TI parameter were supported [8], whereas the new set of the TI parameters contains only two (Table 2). The calculation of the Weighted Moving Average (WMA) requires more computational resources compared to estimation of the Moving Average (MA). At the same time experimentation indicated that

Table 2: TI parameters

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

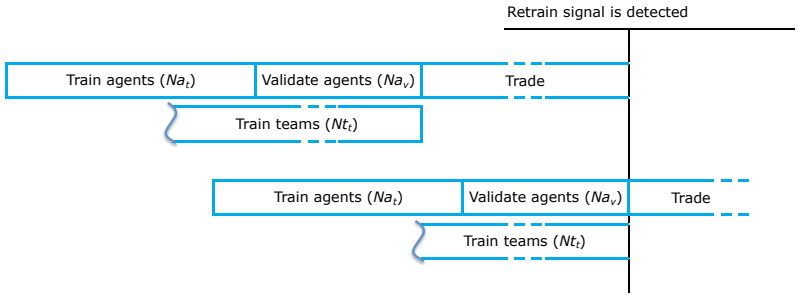


Figure 1: The Train–Validate–Train–Trade cycle. Independent populations are evolved during Training partition Na_t and a subset of *s*FXGP individual identified from each using the Validation partition Na_v. The FXGP teams are trained over partition Nt_t (mode 1 only). The point at which retraining is invoked corresponds to three trading criteria.

the effectiveness of the WMA failed to improve on that using TI based on MA alone, resulting in the simplified set of TI parameters.

3.2 Constructing FXGP teams

Multiple *s*FXGP populations are to be evolved relative to the current historical trading data. That is to say, each time the retraining criteria flags poor trading behaviour, all populations will be re-evolved.[2] The team is built in two ways:

Mode 0: Given P independent *s*FXGP populations, identify one champion trading agent from each using the validation data, Na_v (Figure 1).

Mode 1: As per mode 0, however, all *s*FXGP individuals passing the validation criteria form the basis for a new population, p^*. This population continues evolution with respect to partition Nt_t (Figure 1). Note that each individual from p^* is still treated as an independent trading agent.

Post evolution, each trading agent in the team returns one of three actions per trading interval (hourly in this work), where actions are mapped to an integer value using the following assignment: Sell $= -1$; Stay $= 0$; Buy $= 1$. The scheme adopted for combining the recommendation from each agent assumes the following form: $a = \sum_{i \in A} a_i$ where $a_i \in \{-1, 0, 1\}$ corresponds to the three possible actions that each champion can assume and A is the strongest subset of agents from p^* at the last generation. The resulting number line is then re-mapped into one of the three actions using the following rule:

$$\text{IF } (a \geq b) \text{ THEN } (buy) \text{ ELSE IF } (a \leq -b) \text{ THEN } (sell) \text{ ELSE } (hold) \tag{1}$$

Naturally, the value for the threshold b needs to be defined by the user and remains the same throughout the trading activity. We note that the generic form of this model has been adopted in the past for discretizing the output of (single) neural network trading agents into long and short positions [4] and 'risk management' in the case of boosted DT (γ_0 parameter in [1]).

[2] FXGP utilized three criteria: 1) max. single drawdown, 2) max. number of consecutive loss making trades, 2) max. number of bars without variation [8].

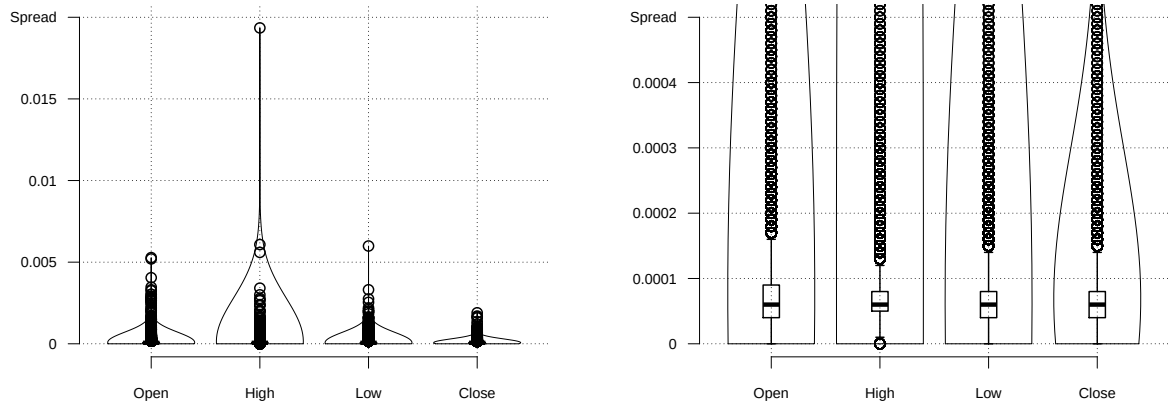
Table 3: Team specific parameters

Parameter	Value	Description
A	3	Number of trading agents in a team (number of independent DT–TI populations)
b	2	Team's trading signal threshold (Equ. (1))
mode	0 or 1	0 - team is built with champion agents, 1 - team is evolved
teams	100	Teams' population size ($\|p^*\|$)
teamsGap	25	Number of teams to be replaced in each generation
teamsGnrts	1000	Max number of teams' population generations
teamsPlt	200	Number of generations without best score improvement to stop training
testSize	500	The data partition size to evolve teams (i.e., $Nt_t = Na_v$)

4 Experimental setup

4.1 Source Data

The EURUSD tick-by-tick prices[3] were converted into one hour bars and used to define market activity during the period from January 3, 2010 to November 30, 2012. We used the same period of time as in [7] to establish a baseline for comparison. The distribution of floating spreads (the difference between Ask and Bid prices) of the hour bars during trading (Open, High, Low and Close prices) are shown in the Figures 2(a) and 2(b). The results in [7] were obtained with the assumption of a *fixed spread* value of 0.00002 USD based on the FxPro Group average EURUSD spread value.[4]



(a) All spreads (0. . . 0.02 USD) (b) Spreads within the range 0. . . 0.0005 USD

Figure 2: Spreads distributions. Internal box-plot provides quartile statistics. Violin profile characterizes the distribution.

Both versions of the algorithm (unmodified and the version we describe in this work) allow user to specify the *limit spread* value to open the trades. If the spread exceeds the limit, the algorithm will not generate *buy* or *sell* signals to open a new trade. However, if the position was already opened, *buy* or *sell* signals to close an existing position will be generated regardless of the spread value. In this research the spread limit was selected empirically and set to 0.00025 USD for all cases i.e., a less conservative limit (relative to the earlier FXGP work), making the role of policy identification more prominent.

[3]http://www.truefx.com
[4]http://www.fxpro.co.uk

Table 4: Quartile performance of trading agents (pips). FXGP†was the best previous result using the original FXGP algorithm with prior knowledge of spread. FXGP is the same algorithm without accurate spread information. *s*FXGP, FXGPT(3) and FXGPT(3e) represent the proposed single agent and two 3 agent formulations.

Algorithm	Profitable runs (%)	Score (pips)				
		min	1st quartile	median	3rd quartile	max
FXGP	73	-3459.0	-63.3	996.2	1905.3	4159.5
*s*FXGP	74	-3153.8	-94.8	988.5	1917.8	4054.6
FXGPT(3)	78	-2219.2	70.2	1117.3	2144.5	6291.0
FXGPT(3e)	81	-1876.6	288.5	1489.2	2462.8	4362.4
FXGP†	82	-3087.9	383.4	1522.5	2639.5	5298.6

4.2 Parameterization

Both updated versions of the algorithm (*s*FXGP and FXGPT) inherit the parameterization of the original FXGP (as described in [8, 7]) with the addition of the parameters, specific for the evolution of teams (Table 3). All runs were performed on a 2.8 GHz iMac computer with Intel Core i7 CPU, 16GB RAM and Mac OS X 10.7.2. Where indicated, use is also made of the Apple GCD enqueue application which identifies tasks for simultaneous execution against the available CPU cores.

5 Results

The following experiments were performed within this research to distinguish between the various components of the system:

- FXGP – original FXGP version of the algorithm as described in [8, 7] i.e., wider range of TI and DT.
- *s*FXGP – *simple* FXGP version of the algorithm i.e., limited TI and DT (Section 3.1).
- FXGPT(3) – teams formed using three *s*FXGP champions under teaming mode 0 (Section 3.2).
- FXGPT(3e) – teams formed using three *s*FXGP champions under teaming mode 1 (Section 3.2)

The results of all three experiments are summarized in the Table 4 where the last line (FXGP†) repeats the best *s*FXGP result from [7] for a fixed spread of 0.00002 USD. Such a decision can only be made given suitable priori knowledge of the market behaviour. This assumption is not made in the case of *s*FXGP or FXGPT. Each experiment includes 100 simulation runs. FXGP and *s*FXGP make *no* use of GCD style parallelization, however, both forms of FXGPT are able to.

Table 4 provides the overview of both the number of profitable runs and the respective quartile statistics. Comparing FXGP to FXGP†indicates that removing the prior knowledge regarding spread limits results in an immediate significant reduction in performance. The simplifications introduced to provide *s*FXGP (from FXGP) have no measurable impact on the quality of trades. Introducing the simplest form of multi-agent behaviour, FXGPT(3) (sampling a single champion from each independently evolved population), results in a $\approx$ 13% improvement to the median score. Introducing evolution using teaming mode 1 (FXGPT(3e)) results in a tightening of the distribution of scores, as well as providing a 50% improvement relative to the single agent case (Table 3). This also results in FXGPT(3e) managing to match the performance of FXGP†, where the latter makes use of prior information in selecting an optimal spread.[5]

In order to characterize the computational costs of each algorithm, we report the total number of retraining events and the cost of any single retraining event. Figure 4 summarizes the total count of retraining events over the three year trading period. In the case of both single agent algorithms (FXGP and *s*FXGP), a significant reduction in the number of retraining events occurs. Given that there was no trading benefit in assuming the (original) FXGP framework over *s*FXGP, this reduction in the number of retraining intervals appears to indicate that *s*FXGP agents are more general. Conversely, there is a significant increase in the number of retraining events when teams of trading agents are assumed (either mode of FXGPT).

[5]The *p*-values for a Student t-test at the 95% confidence interval as applied between each pairwise test of FXGPT(3e) against FXGPT(3), *s*FXGP and FXGP is 0.161, 0.047 and 0.008 respectively.

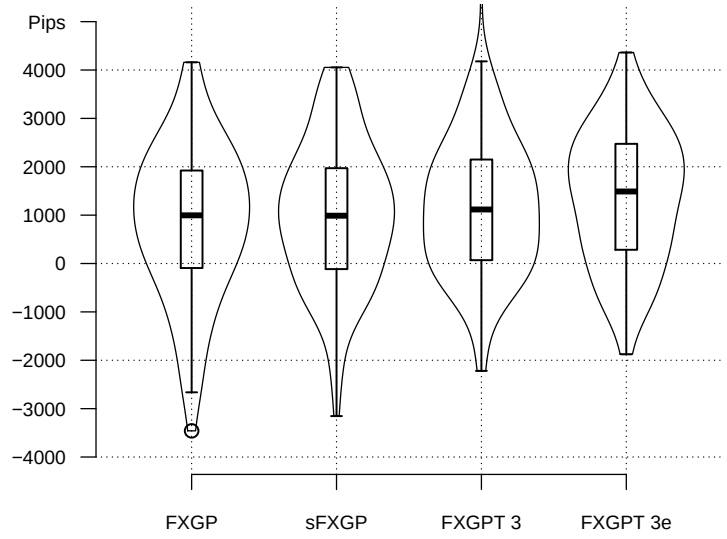


Figure 3: Distribution of scores in pips over trading simulation period of time. Quartile information appears in the box plot (illustrating the information from Table 4). The contours of the violin mimic the actual distribution of the underlying data.

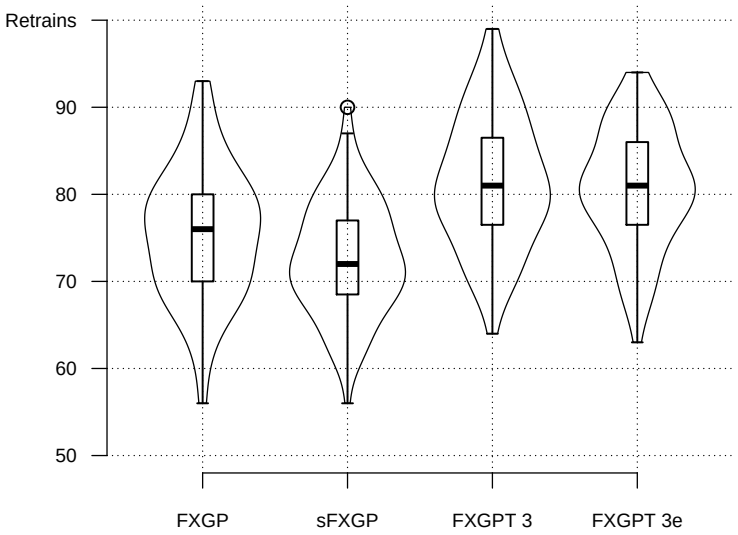


Figure 4: Distribution of number of retraining events over trading simulation period of time. Box plot define the quartile information and violin the actual distribution.

Figure 5 summarizes the cost of performing any single retraining event. It is immediately apparent that *s*FXGP is significantly faster than FXGP as originally conceived. Thus, the cost of supporting multiple types of moving average and division operators as well as a square root operator (TI population) does not result in any better trading performance while reducing the computational overhead by 65 – 70 % w.r.t. *s*FXGP. FXGPT is able to maintain the computational overhead at $\approx 40\%$, albeit with use of the coarse grained parallelism available through Apple GCD.[6]

[6]GCD does not facilitate speeding up evaluation of a single population.

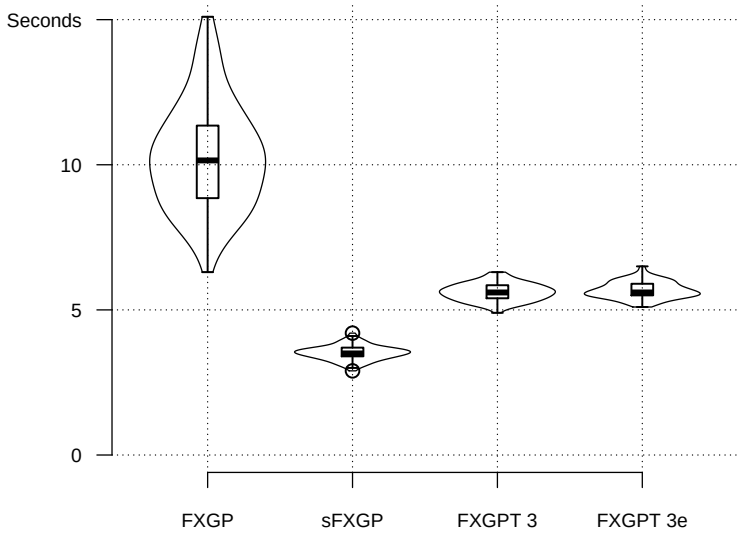


Figure 5: Distribution of average training times (over a run) per population (FXGP and *s*FXGP) and per team of three populations (FXGPT).

6 Conclusion

We can make following conclusions based on the obtained results:

- The use of real prices with floating spreads (Table 4, *s*FXGP) significantly affects the trading results and reduces the number of profitable solutions and scores compared to trading with assumed fixed spreads as previously reported (Table 4, FXGP†).
- Both single agent variants (FXGP and *s*FXGP) return a very similar number of profitable solutions and scores (Table 4). We conclude that the simplifications introduced to *s*FXGP did not reduce the performance of the algorithm and, at the same time, the training time was reduced by 65% (Figure 5). The average number of retrains was also reduced (Figure 4).
- The use of teams of champion trading agents (Table 4, FXGPT(3)) improves the negative spread of runs compared to that of a single trading agent (Table 4, *s*FXGP). At the same time the CPU cost for maintaining a team of champion agents is still significantly $\approx 40\%$ lower than that for the original FXGP.
- The use of evolved teams (Table 4, FXGPT(3e)) outperformed all other configurations and demonstrated the best results in all categories: trustability (the percentage of profitable runs) and quartile scores. Indeed, this configuration provides statistically significant improvements over the single population models (95 percentile) and adds > 350 pips to the median performance of FXGPT(3).

7 Acknowledgements

The authors gratefully acknowledge support from the NSERC CRD program.

References

[1] G. Creamer and Y. Freund. Automated trading with boosting and expert weighting. *Quantitative Finance*, 10(4):401–410, 2010.

[2] I. Dempsey, M. O'Neill, and A. Brabazon. Adaptive trading with grammatical evolution. In *IEEE Congress on Evolutionary Computation*, pages 2587–2592, 2006.

[3] M. Dempster, T. W. Payne, Y. Romahi, and G. Thompson. Computational learning techniques for intraday FX trading using popular technical indicators. *IEEE Transactions on Neural Networks*, 12:744–754, 2001.

[4] C. L. Dunis, J. Laws, and G. Sermpinis. Higher order and recurrent neural architectures for trading the EUR / USD exchange rate. *Quantitative Finance*, 11(4):615–629, 2011.

[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] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1996.

[7] 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*, pages 1341–1348, 2013.

[8] A. Loginov and M. I. Heywood. On the Utility of Trading Criteria Based Retraining in Forex Markets. In *EvoApplications: EvoFIN*, volume 7835 of *LNCS*, pages 192–202. Springer, 2013.

[9] 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.

[10] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259.