Multi-agent Foreign Exchange Market Modelling via GP

Stephen Dignum and Riccardo Poli

Department of Computer Science
University of Essex

Abstract. In this work we combine Genetic Programming (GP) and intelligent agents to build a realistic foreign exchange currency market simulator. GP is used to express and evolve trading strategies. In the paper we analyse the decisions made in the design of the simulator with respect to authenticity of the representation and the efficiency of the system. A number of experimental results are also reported.

1. Introduction

Multi-agent systems built using agents with simple behaviours can be used to demonstrate complicated, often counter intuitive, results seen in the real world [1]. A currency exchange market offers an ideal environment to put this idea into practice. Here, traders and their clients can be thought of as agents with simple desires, to exchange currencies; one to make a straightforward profit, the second to finance a venture outside of the market. The combination of the behaviours exhibited by the agents to achieve those desires creates a complicated, highly dynamic environment with 'emergent' behaviour that is often difficult to explain.

In Genetic Programming (GP) [2,3] genetic material takes the form of expressions and so GP is a natural way to evolve the decisions making strategies of trading agents.

A successful trader will pay no heed to past events, but simply react to current conditions quickly in order to take advantage of potential opportunities [4]. Our approach is, therefore, to evolve agents that react to immediate market conditions. These can take many forms; trade requests from other agents, the state of the market itself such as current trading rates, and external factors, e.g. the presence of good or bad news that could affect the value of currency.

In the remainder of this paper we describe how we modelled a multi-agent foreign exchange simulator based on GP, we analyse the decisions made in the design of the simulator with respect to authenticity of the representation, and we discuss any compromises made in order to improve the efficiency of the system. A number of experiments are also reported which illustrate the dynamics of the simulator and demonstrate how real world behaviour can be seen to occur during a simulation run.

2. The Trading Environment

Although currency transactions will only take place between agents and clients, a central location (the Currency Market) is required to record and calculate trading rates between currencies in order to facilitate those transactions. Within the simulation currencies are simply numbered from 0 to the total number of currencies specified by the user, currency 0 always being the 'base currency'. Rates are then stored for the base currency against each other currency. Cross-rates, i.e. rates between two currencies neither of which is the base currency [5], can be calculated by simply dividing the buy currency rate by the sell currency rate.

When a trade is transacted, the propositioned party (always a trading agent as clients only make requests) will send details of the agreed trade to the Currency Market. The rate of the currency that has been purchased is then updated. The exception to this rule is that if the base currency has been purchased its rate will stay the same and the other currencies will be updated in light of the currency's appreciation or depreciation. On notification of a transaction the Currency Market updates the new rate immediately.

In order to facilitate external influences upon the market, the concept of *currency news* has been introduced. The Currency News Feed component of the simulator provides current news to agents for each trading round. Global information that can affect the demand for currency such as interest rate movements, outbreak of war, etc have been abstracted to a simplified form. A severity code between 10 (for good news) and -10 (for bad news), is associated to a particular currency for a particular trading round. The amount of news released is limited by a parameterised variable and directly affects a client's desire to purchase or sell a particular currency.

Clients can be seen as simplified traders. They desire to conduct transactions to finance activities outside of the market itself and are therefore interested only in purchasing quickly at an acceptable price. A client's profit is made outside of the market through non-trading activities. Clients are, however, important to the market as they introduce new money that can be traded. In fact the market would not exist without the external need to exchange currency as traders would simply be trading a finite resource.

For efficiency reasons, our clients are relatively simple. Their activities are limited to choosing currencies to buy and sell and approaching traders willing to accept the transaction at their chosen rate.

A *client agent* chooses a pair of currencies to trade, one to buy, another to sell, based on current market conditions. The client then selects a trading agent (trader) to transact with; the decision regarding which agent to select is governed by previous transactions. If the trading agent refuses to trade another trader is chosen by the client. This process continues until either the proposed trade is successfully transacted or the list of active traders is exhausted.

The client stores a list of all traders, initialised in random order, and then steps through the list in order until a trade is accepted or the list is exhausted. The trader that accepts the exchange is then placed to the top of the preference list, and will be asked if it wishes to trade first by that client in the next round. By avoiding an extended bargaining period, this method provides insurance against combinatorial

explosion in simulations with large client and trader populations, whilst still mimicking the real life behaviour of clients.

For each trading round a client will attempt to trade a single currency pair. The selection of the currencies to buy and sell is stochastically based, biased in favour of current currency news. Each currency is given a starting value and cumulative news values are then added to that value. The 'roulette wheel' algorithm [6] is finally applied to select one currency based on the modified values.

The amount that a client desires to purchase is selected at random but limited to a maximum (parameterised) value. The client will also choose a required rate at random although the proposed rate is not allowed to deviate from the current rate by more than a parameterised percentage. This method prevents harsh currency movements, and imitates actual currency markets in that traders and clients would expect to trade close to a publicly quoted rate.

Once all the trade request details are selected, an Exchange Request is created to store the requirements. This is then sent to the traders in preferred supplier order as described previously.

Trading agents are more sophisticated than clients. They have holdings in particular currencies and can increment and decrement those holdings by conducting trades with both clients and other trading agents. They also incorporate the idea of currency specialisation when choosing agents with which to trade.

Traders exist within the market solely to make a profit from engaging in currency transactions. These agents will follow particular trading strategies in order to maximise their overall balance. To implement this functionality, the trading agent is created with a mechanism to store its current holdings and to update those holdings when a currency transaction is successfully completed.

In order to calculate an overall balance, the agent is also equipped with the ability to determine the value of its holdings expressed in the base currency. This process is known as 'marking-to-market' and is used by financial institutions to provide an indication of their holdings' current worth. This ability acts as a *fitness function* and determines whether the agent is allowed to continue trading i.e. whether it is still solvent. The fitness function also influences the likelihood of the trader being selected as a parent for a new agent during evolutionary processing, i.e. the wealthier the agent the more likely it is to be selected.

At the beginning of the simulation each trader's holdings are initialised by assigning a random quantity, limited by a parameterised value, to each currency. Also, on creation each trader is provided with a GP tree representing the current trading strategy. From the trader's point of view, this is simply a black-box for which it supplies the current values of a number of variables, and as a result is provided with an amount to trade.

The GP trading expression is used in two ways. First when a request to trade is received, if the amount returned from the expression after evaluation using current variable values, is greater or equal to the amount requested in the trade, the trader will accept that trade. Secondly, when the trader is given the opportunity to trade on its own account, it will choose two currencies at random and choose the appropriate trade amount by evaluating the expression. Note, selection of currencies by the trader is not influenced by currency news. However, the values of current news for the currencies selected can be passed as variables to the expression. Intuitively one would

expect the simulator to evolve expressions (those with access to news variables) that would try to obtain more of a currency that has positive news.

Trading agents are allowed to sell currency that they do not currently own i.e. to run a negative balance in a particular holding. This process is known as being 'short' [5], and can be thought of as entering into a contract to supply a currency at a later date. To be short in certain currencies is perfectly acceptable providing the agent has the resources to settle its debts. However, such agents are at the mercy of exchange rates, and what may look like an acceptable overall balance may soon deteriorate as rates move against the agent's positive holdings, thereby making agents bankrupt. From an evolutionary point of view, if we did not allow our traders to go 'into the red' we could have the situation where they have too small a balance in a currency to trade, irrespective of whether their strategy is any good they would eventually be removed as 'lazy traders', possibly losing some useful genetic material. Having this ability also allows us to model a set of the interesting market dynamics, most notably bankruptcy through the erosion of healthy balances by adverse exchange rate movements i.e. by the appreciation of currencies for which the agent holds negative balance, or depreciation of those for which it holds a positive amount.

In a similar manner to clients, traders also maintain a preferred supplier list of agents with which they wish to trade. Unlike clients, traders have a preferred list of suppliers for each currency. This allows us to model trader currency specialisation which is a feature of many currency markets.

Also included in the trading agent design is a counter that records the number of trades, this provides not only an interesting statistic to be analysed, but also facilitates the removal of lazy traders by the trading simulator.

A trading simulation begins by reading the simulation parameters and the currency news to be used during the simulation from file. This information is used to create a currency market with a set number of currencies and the currency news feed process.

A population of trading agents is then created to a fixed size. A fixed size has been used to control the amount of time that agents spend trading and also to simplify the evolutionary process i.e. so that we need only concern ourselves with the replacement of bankrupt or lazy traders that have left gaps within the population. There can, however, be advantages in having an unlimited number of agents in that this allows more successful genetic material to persist within the environment, thereby providing greater competition [7][8][9].

A list of all available trading agents is then used to initialise a population of clients in order that they have a reference to potential traders. The size of the client population is also fixed.

Finally the main simulation loop is then called, the first part of which is the trading round.

A *trading round* is a distinct trading session where all clients and traders have the opportunity to request a trade. A real world analogy for a trading round would be of a trading session within a financial market, the output of each round can be thought of as a set of 'close of business' valuations. This works as follows.

First, the currency news for this trading round is read. Each client is then asked in turn to make an exchange request to the trader population (in order of its own preferred supplier list). The propositioned trader will only accept an exchange if its

trading expression returns an amount greater or equal to the client's requested amount.

After each client has made a request, each trader is then asked in turn to make a request to other traders.

Not all trading rounds involve evolution. The ratio between evolutionary and non-evolutionary rounds is determined by a simulation parameter. By allowing the user to vary the number of trading rounds to evolution rounds, the simulation provides the concept of an 'averaging period' where an agent's performance can be judged over user specified numbers of trading rounds. If the current trading round is also an evolution round, evolution takes place as described below, otherwise the above process beings again.

In order to avoid the 'Red Queen' effect and to maintain trading competition, the simulator implements a 'Hall of Fame' [10] structure to record the genetic material of agents that competed well in previous trading rounds. This is to ensure that if genetic material is lost through the agent eventually becoming bankrupt, that material can be reintroduced to the population. Although, arguably, the agent will eventually be removed from the population once more, this method ensures that newly evolved agents will have to compete successfully against previously successful agent strategies.

In an evolution round, first the trading agent with the highest fitness in the current trader population is selected. If this agent has genetic material that is not currently contained in the Hall of Fame then a copy of the agent is added to the Hall of Fame. Next, bankrupt traders are removed from the main population. Also removed are traders that have not traded by a user specified number of rounds. These are termed 'lazy traders' and are removed to prevent agents existing in the population by simply refusing to trade.

At this point the simulator attempts to insert a member of the Hall of Fame into the first vacant slot in the population. This genetic operator is used with a pre-fixed probability, which is another parameter of the simulation. When invoked, the operation is performed by creating a pool of agents from the Hall of Fame that contain genetic material *not present* in the current population and then selecting one of those agents at random for insertion. The operation cannot be performed if there are no bankrupt or lazy traders to replace, or if all genetic material held in the Hall of Fame currently exists within the population.

The simulator then creates new trading agents by first selecting two distinct parents. Each parent is selected using tournament selection. The selection of parents is biased in favour of the wealth they currently hold. Wealth, therefore, acts as the *fitness measure* for each member of the trader population. The two selected parents are then allowed to reproduce to provide two new agents for the population. If there is only one remaining space available in the population, the first child is retained and the second discarded.

Once all population spaces are filled, the evolution round ends and the next trading round begins. This process continues until a parameterised number of trading rounds is completed.

The trader population is expected to get fitter during the course of a simulation run. The reasons for this are two-fold. Firstly, bankrupt agents are removed from the trader population during evolutionary processing and new ones are instantiated with positive

holdings, thus turning negative holdings into positive ones. Secondly, and most importantly, clients are making trades in a random manner. These trades do not have to be accepted by any trading agent, so, as the agent strategies evolve, only beneficial trades to the traders should be accepted. Clients, therefore, have the effect of introducing new money into the system.

3. Genetic Representation and Manipulation

Tree structures were chosen to represent trading expressions [11,3]. The "Ramped Half-and-Half" [3] method was employed for initialisation of each expression.

The function set includes the simple arithmetic operations. The division function is 'protected' [12] to ensure 'closure' [3]. The choice of variables has been made based upon educated guesses as to possible useful components of an agent's strategy. The terminal set is given in the following table:

Terminal	**Description**
buyHolding	Trader buy currency holding.
sellHolding	Trader sell currency holding.
buyNews	Current cumulative news for buy currency.
sellNews	Current cumulative news for sell currency.
proposedRate	Proposed cross-rate between buy and sell currencies.
currentRate	Current cross-rate between buy and sell currencies.
buyRate	Current rate between buy currency and base currency.
sellRate	Current rate between sell currency and base currency.
tradeCount	Total trades so far transacted within the simulation run.
Constants	Random constants in the interval [-10,10]

Standard sub-tree crossover was used. A variant of subtree mutation was also used where if the mutation point is a terminal then only the terminal is replaced, while if a function is chosen the sub-tree rooted at that function will be replaced.

Once the trading expressions for the child agents are created the phenotypes of their respective agents need to be created. There are a number of issues, however, regarding the initialisation of holdings for the new agents [13,14]. If the standard initialisation process is used, that is of selecting a limited random number of holding for each currency, we are introducing an inflationary effect where the negative holdings of deleted traders are replaced by the positive holdings of the new children. Also, as exchange rates will have moved, the average wealth of the new traders will be different from those created when the simulation began. The simulator provides two additional holding initialisation methods for experimental investigation. The first averages the holdings of each parent by currency, and assigns this to the child. This has the effect of creating children with similar wealth to their parents. The second method initialises the children as normal. However, the average of each child currency holding is subtracted from each parent.

4. Experimental Results

As described in the previous section there are numerous aspects of the simulation that are suitable for parameterisation. Even by limiting ourselves to the twenty or so parameters made available by the parameters file we are faced with an enormous amount of parameter value combinations to choose from.

The approach taken in the following experiments was to choose an initial set of values suggested by the GP literature for our evolutionary process, set real world values for trading limits, rate variation amounts etc; and finally to start with small populations of clients and traders varying their numbers as required.

The experiments have been designed mainly to test the simulator, show its basic features and some limitations. More extensive experimentation is planned and will be reported in future work.

Since many of the phenomena emerging in our experiments happen stochastically at unpredictable times, below we will report data from single, but typical, runs rather than averages.

The aim of a first experiment was to show the effect of evolution on the wealth of the trader population. In figure 1 we provide a graph where the overall fitness of the trading agent population is plotted against trading round for a non-evolved population. A small population of 10 traders was used, with a set of 30 clients.

As we can see the overall fitness varies over time with the value becoming more volatile as the number of trading rounds increase. Traders are slavishly following the strategies they have been given on initialisation. The effect of movements in trading rates makes the value of their wealth and trades become increasingly volatile as the number of opportunities to alter these rates grows.

Next, we analysed the effect of non-evolved trading on the mean fitness of the individuals within the population and the best and worst fitness values recorded for each trading round. This is shown in figure 2.

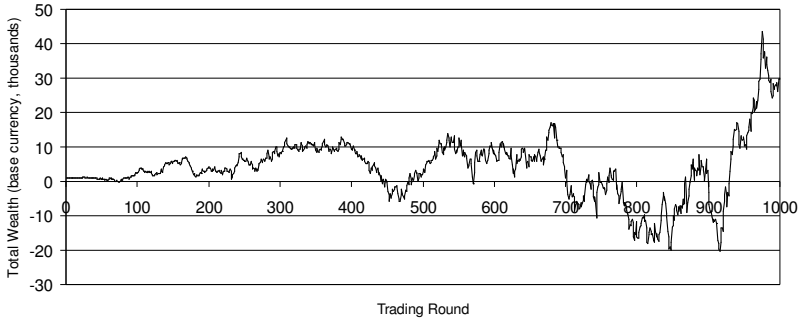

Fig. 1. Population fitness when evolution is switched off.

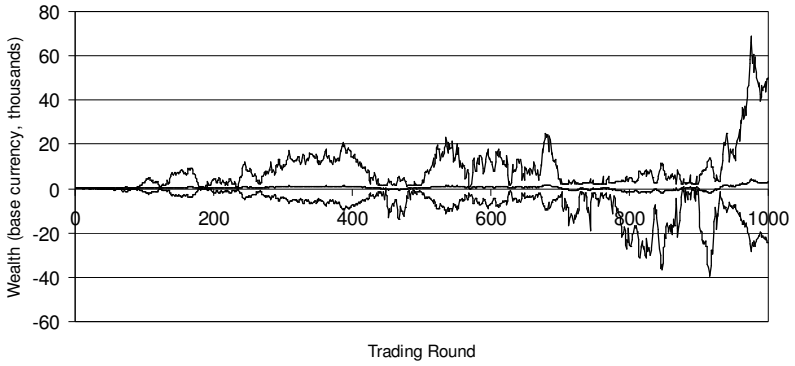

Fig. 2. Best, worst and average fitness when evolution is switched off.

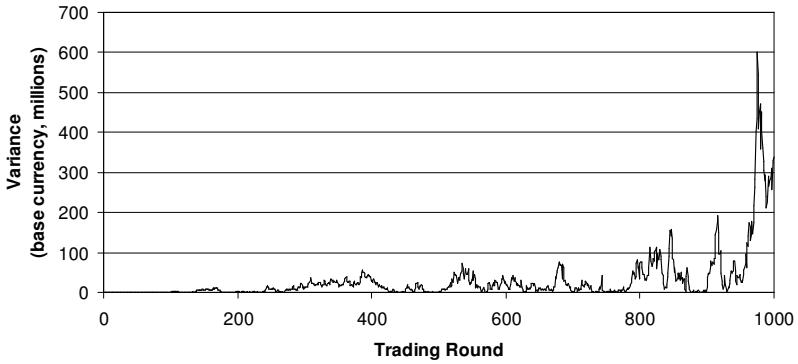

Fig. 3. Population variance when evolution is switched off.

First, we notice that the mean value remains fairly constant, within a hundred or so units either side of zero. However, the best and worst values again become increasingly volatile as exchange rates diverge and the effect of trades with both clients and other traders becomes more tangible.

To further illustrate this, figure 3 shows the effects of non-evolved trading on the variance of the population. Variance increases rapidly after approximately eight hundred rounds as exchange rates become more diverse.

Next, we analysed the effect of the evolutionary process on the population. Using the same trading parameters as for the non-evolutionary population, we have allowed evolution to take place every hundred rounds. Figure 4 shows that evolution has had a beneficial effect on overall population wealth. The negative fitness values have been removed whilst the overall population wealth is recorded in units five times greater than its un-evolved counterpart shown in figure 1.

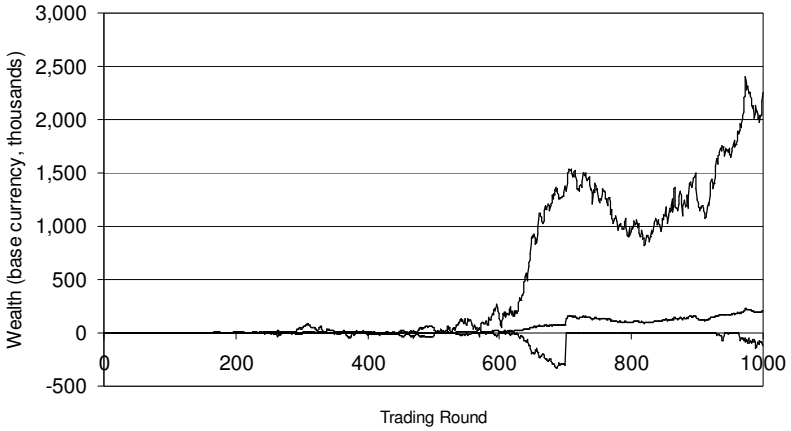

Fig. 4. Best, worst and average fitness when evolution is applied.

If we look at the best and worst fitness values recorded for the traders within the population during trading, we find a steadier, less volatile improvement in fitness for the best recorded values than for the non-evolved population (figure 1). The most noticeable improvement is that of the worst individual, which stays relatively close to a zero wealth value. Intuitively we would expect this to happen as the evolutionary process extracts bankrupt traders thereby removing the worst individuals. However, what is interesting is that we see this effect only once, when the seventh evolutionary round is applied (trading round 700). If this was the only reason for the improvement in population wealth, we would expect the 'saw tooth' effect – the gradual decline of the weakest individual from the application of evolution, which is bluntly rectified by the application of another evolution round – between every evolutionary stage.

The variance of the evolved population shown in figure 5 shows an even more marked increase in divergence within the fitness values displayed by the evolved population than that of the non-evolved set of traders, so much so that a logarithmic scale has had to be used. The application of evolution has therefore not affected all of the agents equally, and its application has increased the gap in wealth between traders.

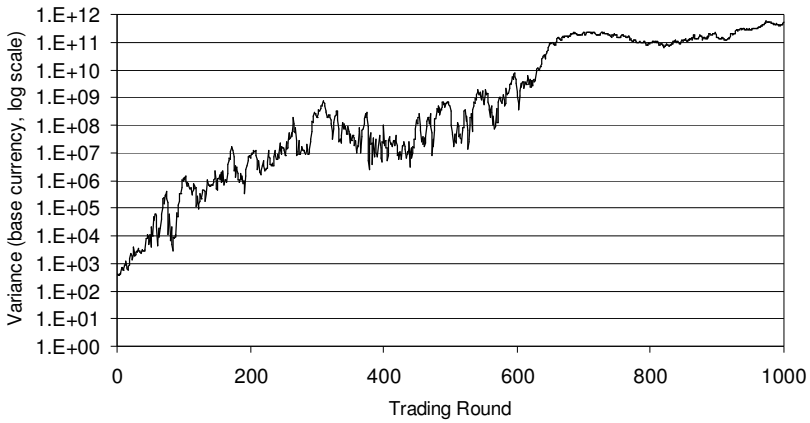

Fig. 5. Population variance when evolution is applied.

A further experiment was designed to show the effect of the number of trading rounds that are allowed to take place before applying evolution. This can be thought of as another way of viewing the environment. Rather than applying evolution to increase overall population wealth, we are using increasing numbers of trading rounds before applying evolution so that the agents can be tested over a longer period with the intention of selecting better agents as parents. The Evolution Ratio parameter has been altered so that increasing numbers of trading rounds are used before evolution is applied. In order that the results can be compared, the total trading round value is also altered to ensure that a set number of trading rounds take place within each simulation. The results of the experiment are shown in figure 6.

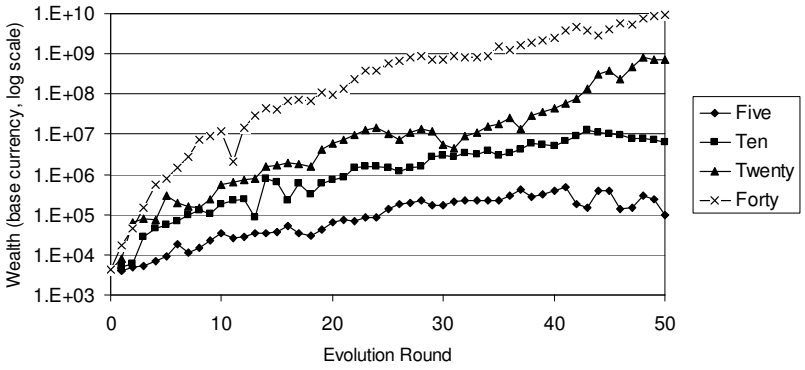

Fig. 6. Population wealth by evolution ratio.

We can see from the graph that by applying more trading rounds the overall wealth for each evolution will increase. The experiment does not tell us, however, whether this is due to the increased testing period producing better agents, or whether the

increased trading period has allowed agents that would have performed just as well in the other simulations more opportunities to accumulate wealth.

We then looked at the numbers of variables and their relationships with functions found in the later populations to see which configurations of these the agents have found useful. In a number of simulations there are a relatively large number of occurrences of the following sub-tree: (*DIVIDE buyRate buyHolding*). This suggests the calculation has some significance to the problem posed by the environment. A possible reason could be that as higher buy rates indicate higher worth of the currency to the base rate, the strategy tempers the overall desire to purchase the currency by dividing it by the holding of the currency it already holds. Also the *tradeCounter* variable is often used with the multiply function. The reasons for this are more straightforward to explain. As rate divergence increases with each trade, the variable provides a 'handle' on the process and allows the agent to trade more as the simulation increases.

5. Conclusions

Modelling trader and client behaviours within a simulated trading environment can provide a valuable technique to analyse the dynamics of foreign exchange markets. Using a multi-agent approach allows us to engineer and replicate, from simple behaviours, complicated, often emergent, aspects of financial markets.

Evolutionary methods are complementary to this process. They allow agents to adapt to new market conditions, and provide a facility for the researcher to not only identify expressions and variables pertinent to certain market conditions, but also to discover robust trading strategies that are successful over a large range of trading circumstances. This paper has concentrated on GP as the primary technique for evolutionary change. This has been shown to be a natural way to represent and manipulate an agent's decision-making process.

We have restricted the simulation to evolve a single aspect of the agent's functionality, the choice of an amount to trade. However, it is certainly possible to use the method to evolve other aspects of the agent's decision-making ability, such as the choice of buy and sell currencies or the setting of proposed rates. More sophisticated, 'knowledge' based operators will be required, however, to ensure syntactic correctness of the evolved programs [15].

Bibliography

[1] J. Rauch, Seeing around Corners, *The Atlantic Monthly*, April 2002, pages 35-48 [http://www.theatlantic.com/issues/2002/04/rauch.htm], Last visited: 26/08/2003

[2] J. R. Koza (1992), Genetic Programming: On the programming of computers by means of natural selection, The MIT Press, Cambridge, Massachusetts USA, 1992

[3] W. Banzhaf, P. Nordin, R. E. Keller, F. D. Francone (1998), *Genetic Programming - An Introduction; On the Automatic Evolution of Computer Programs and its Applications*, Morgan Kaufmann Publishers, Inc. San Francisco, California, USA 1998

[4] M. Lewis (1999), *Liars Poker: Two Cities True Greed*, Coronet Books, Hodder and Stoughton Ltd, 338 Euston Road, London NW1 3BH, 1999
[5] S. Valdez (1997), Foreign Exchange in *An Introduction to Global Financial Markets,* Macmillan Press Ltd, London UK, 1997, pages 137-172
[6] T. M. Mitchell (1997), 9.2 Genetic Algorithms in *Machine Learning,* McGraw-Hill, New York USA, 1997, pages 250-256
[7] R. E. Smith, C. Bonacina, P. J. Kearney, T. Eymann (2000), Integrating Economics and Genetics Models in Information Ecosystems, *Proceedings of the Congress on Evolutionary Computation (CEC 2000), La Jolla, USA 16-19 July 2000*, IEEE Press, pages 959-966
[8] R. E. Smith, C. Bonachina, P. J. Kearney, W. Merlat (2001), Embodiment of Evolutionary Computation in General Agents, *Evolutionary Computation*, The MIT Press, Cambridge, Massachusetts USA, Vol. 8, No. 4, pages 475-493
[9] R. E. Smith, N. Taylor (1998), A Framework for Evolutionary Computation in Agent-Based Systems, *Proceedings of the 1998 International Conference on Intelligent Systems,* ISCA press, pages 221-224
[10] C. D. Rosin, R. K. Belew (1996), New Methods for Competitive Coevolution, *Evolutionary Computation,* The MIT Press, Cambridge, USA, Vol. 5, No. 1, pages 1-29
[11] T. M. Mitchell (1997), 9.5 Genetic Programming in *Machine Learning,* McGraw-Hill, New York USA, 1997, pages 262-266
[12] W. B. Langdon (1998), *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!* Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, USA, 1998
[13] B. LeBaron (2001), Empirical Regularities from Interacting Long and Short Memory Investors in an Agent Based Stock Market, *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 5, October 2001, pages 442-455
[14] F. Nolan, J. Wilkiewicz, D. Dasgupta, S. Franklin (1999), Evolutionary Economic Agents, *Proceedings of the sixteenth national conference on artificial intelligence and eleventh innovation applications of AI conference,* American Association for Artificial Intelligence, Menlo Park, CA, USA, pages 38-43
[15] G. F. Plum (1998), Genetic Algorithms/Developments and Variations on the Genetic Algorithm/Representation/Programs in *Simulation for the Social Scientist*, The University of Koblenz, Germany.