Foreign Exchange Trading using a Learning Classifier System

Christopher Stone and Larry Bull

UWE Learning Classifier System Group Technical Report UWELCSG05-007
Faculty of Computing, Engineering and Mathematical Sciences
University of the West of England
Bristol, BS16 1QY, United Kingdom
christopher.stone@uwe.ac.uk
larry.bull@uwe.ac.uk

Abstract. We apply a simple Learning Classifier System that has previously been shown to perform well on a number of difficult continuous-valued test problems to a foreign exchange trading problem. The performance of the Learning Classifier System is compared to that of a Genetic Programming approach from the literature.
The simple Learning Classifier System is able to achieve a positive excess return in simulated trading, but results are not yet fully competitive because the Learning Classifier System trades too frequently. However, the Learning Classifier System approach shows potential because returns are obtained with no offline training and the technique is inherently adaptive, unlike many of the machine learning methods currently employed for financial trading.

1 Introduction

In previous work [21] we demonstrated a simple Learning Classifier System [8] performing well on a number of difficult continuous-valued test problems. In this paper, we apply the same Learning Classifier System to a foreign exchange trading problem. To provide a benchmark against which the performance of the Learning Classifier System can be measured, we adopt the framework and data used by Neely et al. [13].

The Learning Classifier System that we use is based on ZCS [23]. To improve performance and robustness on single-step Boolean problems, we use a simplified version of the NewBoole [4] update mechanism instead of the normal ZCS update algorithm.

We also employ two parameterless operators, a cover operator and *specialize*. The cover operator removes the need for the s_0 parameter, which is known to be highly sensitive and the specialize operator helps combat overgeneral classifiers. Results with continuous-valued test problems show that these new operators are effective with no detrimental side-effect on any of the problems tested.

Data snooping, where data is used multiple times for the purposes of statistical model building and evaluation, is a well-known problem and challenge in

data mining and machine learning [9] and also for technical trading [22]. The Learning Classifier System used in the present work was developed without recourse to financial data and to reduce the likelihood of data snooping we use it 'out of the box' for the foreign exchange trading problem.

This paper is arranged as follows. Section 2 provides an introduction to foreign exchange trading and an approach previously reported in the literature upon which our experiments are based. Details are given in Section 3 of the Learning Classifier System used and the experiments performed are presented in Section 4. Section 5 provides conclusions to the work.

2   Foreign Exchange Trading

2.1   Foreign Exchange Transactions

A foreign exchange (FX) rate is the amount of one currency (the *base currency*) that must be sold (bought) in order to buy (sell) one unit of another currency (the *counter currency*). An FX transaction consists of borrowing an amount of one currency and using the proceeds to fund the purchase of another currency. A *long position* in the counter currency is opened when a trader borrows enough of the base currency to fund the purchase of an amount of the counter currency at the prevailing exchange rate. The position is closed when the trader sells the counter currency at the then current exchange rate and uses this amount to repay the loan of the base currency. If, in the time between the position being opened and closed, the exchange rate rises then the trader will have made a profit on the transaction.

One of the attractions of FX trading is that the market is symmetrical and provides opportunities for profit in all market conditions. A *short position* may be taken if the trader believes that the exchange rate will fall. In this transaction the trader borrows an amount of the counter currency and uses this to fund the purchase of the base currency.

Most FX traders must operate a *margin* account. Trader's funds are held on deposit in the margin account and are used to allow risk management on the part of the institution providing dealing facilities.

Though most FX trading is carried out at high frequency, to simplify matters and to allow comparison with work appearing in the literature, we focus here on daily FX data. This is FX data where the quoted price is obtained by sampling once per trading day at a fixed time of day. In the scenario considered, a trader may, in addition to profit arising from changes in exchange rates, also benefit from any differential in the interest rate in force from positions held in the respective currencies.

2.2   Transaction Costs and Slippage

To be effective, a trading agent model must take account of *transaction costs* and *slippage* incurred in executing a trade. Transaction costs are the commission that

is paid for execution of the trade and the difference between the bid (selling) price of a security and its asking (purchase) price. Slippage is the difference between the estimated and actual transaction costs. These costs are measured in terms of *basis points.* A basis point is one hundredth of a percent of the cost of the transaction. Typical transaction costs for FX trading are in the order of 2–4 basis points. Early studies of trading models typically ignored the effects of transaction costs and slippage. They often concluded that it was possible to make significant profit using these trading models.[1] However, these conclusions are erroneous, since transaction costs and slippage have a large effect on the ability of the model to trade profitably [6].

2.3 Genetic Programming Approach

Neely et al. [13] used Genetic Programming (GP) [10] to evolve technical trading rules. This Genetic Programming approach built on work originally performed by Allen and Karjalainen [2], which, though published after that of Neely et al., was available earlier as a working paper. Much of the attraction for generating trading rules using GP is due to its ability to build arbitrary mathematical functions from a predefined set of primitive operators. In principle, this allows technical trading rules to be generated and tested with relatively little bias on the part of the researcher or practitioner. Previous work in the literature did not usually have this ability and typically studied small fixed sets of technical trading rules.

The data set used in this study consisted of six sets of financial time series, each containing daily data from January 1, 1974 to October 11, 1995. The time series contain data for the United States Dollar (USD), German Deutsche Mark (DEM), British Pound (GBP), Japanese Yen (JPY) and the Swiss Franc (CHF). Each data set consisted of three related time series: the exchange rate for the day concerned and the interest rates in force on each day for the base and counter currencies. Each time series contained around 5260 elements. Data from the year 1974 was used to provide a buildup period for the indicators used, as these require history information to operate. Further details of the time series are given in [13].

The procedure used for evolving rules was to evaluate a population of 500 initially random rules using data from 1975–1977 (the *training period*). The best performing rule from the training period was saved. A new generation of 500 rules was then created using recombination[2] and the best performing rule of that generation was compared over a *selection period* of data from 1978–1980 to the best performing rule saved previously. The better rule was retained. This procedure was repeated for 50 generations or terminated earlier if no new best rule emerged after 25 generations. If the resulting single rule produced a negative excess return (see Section 3.2 over the selection period it was discarded.

---

[1] Unfortunately, this practice still continues, see, for example, [11].

[2] The authors do not mention the use of any form of mutation operator for the generation of new rules.

Table 1. Genetic Programming approach with validation period 1/1/1981 to 30/9/1995

	USD/GBP	USD/DEM	USD/CHF	USD/JPY	DEM/JPY	GBP/CHF
APR	2.28%	6.05%	1.42%	2.34%	4.10%	1.02%
Monthly std dev	3.66%	3.49%	3.88%	3.48%	2.79%	2.92%
Rules +ve return	85%	96%	84%	65%	85%	89%
Sharpe Ratio	0.18	0.50	0.11	0.19	0.42	0.10
Number of trades	130.51	106.54	156.58	107.98	426.61	55.25
Long positions	63.42%	50.52%	81.73%	78.01%	49.91%	93.57%

Otherwise, its out of sample performance was determined on a *validation period* of data from 1981–1995. The study presented the results of 100 such rules, which are shown in Table 1. A detailed commentary of the results is given in Section 4.2.

3   Learning Classifier System Approach

3.1   Representation

There are two main differences to consider when adopting the Genetic Programming approach to a Learning Classifier System.

Firstly, the primary attraction of using a Learning Classifier System is to exploit the fact that it is an adaptive online learning technique. Although it would be possible to perform experiments in batch mode, as was done with the Genetic Programming approach, we would ultimately like to use the Learning Classifier System to facilitate online trading, so we will emphasize this aspect in our experiments.

Secondly, the representation we are using does not offer the degree of expression provided by Genetic Programming. In particular, we are not able to describe arbitrary mathematical expressions with the continuous-valued representation used throughout this work. It is important to note that this is a direct consequence of the choice of interval representation and is not an inherent limitation of a Learning Classifier System. The use of Genetic Programming s-expression representations have been investigated with Learning Classifier Systems in [1, 12].

To provide a reasonable choice of technical trading indicators within the constraints of an interval representation all indicators are of the form:

$$I_t = \frac{f_1(t, \Delta t_1)}{f_2(t, \Delta t_2)} \tag{1}$$

$$f_1, f_2 \in \{lag, avg, min, max\}$$
$$0 \leq \Delta t_1 < \Delta t_2$$
$$0 \leq \Delta t_2 < 255$$

In Equation 1, an indicator I_t for day t is the ratio of two primitive functions of the time series, f_1 and f_2. These primitive functions operate on elements of the exchange rate time series from the current day, t back in time to $t - \Delta t_1$ and $t - \Delta t_2$, as applicable. The primitive function comprising the numerator of the ratio is constrained to operate on a smaller time window $\Delta t_1 < \Delta t_2$. This halves the search space by avoiding redundant searches for equivalent indicators where one indicator is the reciprocal of another. Though the set of primitive functions is quite small they allow the construction of popular technical indicators such as moving average, filter and trading range breakout rules.

There are four primitive functions of the time series used. $lag(t, \Delta t)$ returns the exchange rate at day $t - \Delta t$. $avg(t, \Delta t)$, $min(t, \Delta t)$ and $max(t, \Delta t)$ return the mean, minimum and maximum value, respectively, of the exchange rate from day $t - \Delta t$ to day t.

Indicators are created by the cover operator with the two primitive functions being chosen at random and the integer parameter Δ_t for the primitive functions being chosen uniform randomly from the appropriate range. These parameters are encoded into a single interval predicate $[\Delta t_1, \Delta t_2]$ using an 8-bit binary encoding. This technique is a convenient means of using the same interval representation for integers that we have previously used throughout for real numbers [19]. When using a one of m binary encoding for numbers, the encoding for integers differs from that of real numbers only in the size of the alphabet used and the consequent size of the genotypic search space.

Although this interval exists as part of the classifier's condition, its sole use is to describe an aspect of the trading rule and it is not used for matching. The Genetic Programming approach uses a 250-day history period for its trading rules, whereas the 8-bit encoding used for the Learning Classifier System allows parameters to be in the range [0,255]. Rather than having to deal with the additional complexity of out of range values we employ a 255-day history period.

A second, continuous-valued, interval predicate, is used to define the range that the indicator I_t may take for matching to occur. This interval is in the range $[-1, 1)$. In the Genetic Programming approach, foreign exchange time series values are normalized by dividing each raw value by a 250-day moving average. With indicators being constrained to the form shown in Equation 1 such normalization is not necessary since the indicators used are ratios that stay within certain empirical bounds.

Genetic search occurs on the two interval predicates representing an indicator's range and Δt parameters. To preserve the ordering semantics of the Δt parameters and to maintain continuity with previous work, crossover *between* predicates is used. Crossover occurs such that primitive functions and their parameters are exchanged. For example, if $log(4)/max(10)$ and $min(11)/avg(50)$ are crossed, the indicators that result are $min(4)/avg(10)$ and $lag(11)/max(50)$. As there are only two interval predicates, crossover between the condition and action parts of a classifier would, if allowed, change the action of a classifier with a high probability. For this reason, we restrict crossover to occur only in the condition part of a classifier. Mutation is used to vary the range and Δt

parameters and the classifier's action in the normal manner. We have not yet experimented with extending the genetic search to include more flexible forms of variation of primitive function types.

3.2   Excess Return

Each classifier advocates one of two possible actions, long (1) or short (-1), indicating the direction of the position recommended by the classifier. One trial occurs for each element t of the time series and the Learning Classifier System adopts for trial t the position chosen by its action selection mechanism. When the Learning Classifier System receives time series data for trial $t + 1$ it is able to calculate the additional return obtained by trading compared to that which would result from interest accrued on the margin. This is known as *excess return*.

The excess return r_t for a trial is influenced by the exchange rate S at trial t and $t + 1$ and the differential between the daily interest rate i_{bt} for the base currency in force at trial t and that of the counter currency i_{ct}. As the data sets contain only data for days when the markets were trading, a position open over a weekend or holiday will accrue interest for $\Delta t_p > 1$ days. The trading model adopted assumes that all of the agent's margin is traded on each trial.

For a long position, excess return is calculated according to the following formula:

$$r_t = \frac{S_{t+1}}{S_t} \frac{(1 + i_{ct})^{\Delta t_p}}{(1 + i_{bt})^{\Delta t_p}} \qquad (2)$$

For a short position, excess return is:

$$r_t = 2 - \frac{S_{t+1}}{S_t} \frac{(1 + i_{ct})^{\Delta t_p}}{(1 + i_{bt})^{\Delta t_p}} \qquad (3)$$

Excess return is calculated for both the position adopted and the position that was not chosen for a particular trial. From this information the correctness of the position is ascertained and classifiers are allocated to the correct and incorrect sets (see [21]) to receive the relevant reward.

The probability distribution of financial returns maps the size of a financial return to its probability of occurrence. The nature of these distributions is much studied and debated by financial researchers, but in general terms it is known that the distribution of FX returns is fat-tailed, approximately symmetric and with a mean of almost zero [5, chap. 5].

In a single-step model, such as that used here, the Learning Classifier System is rewarded based on the return received for a single transaction. The obvious choice is to arrange for the reward the Learning Classifier System receives to be related to the return generated by the action of the Learning Classifier System. This is a natural approach since the profit made from an action matches well the notion of utility used in many Learning Classifier Systems. The advantage

of this technique is that it allows rules that generate numerically higher returns to be distinguished from other rules.

The Learning Classifier System is rewarded with the value $10^6(r_t - 1)$ according to the excess return obtained. For continuity with earlier work, this value is designed to provide rewards of similar magnitude to those used in [20] for stochastic test problems. Reward received is shared amongst members of the correct set.

The effects of transaction costs mean that it is possible for the correct action to receive a negative excess return. This disrupts the fitness values of the Learning Classifier System and is incompatible with the operation of the Learning Classifier System's Genetic Algorithm (GA) [7]. To solve this problem, reward must be limited to positive values only or additional mechanisms must be added to the Learning Classifier System to prevent a rule's fitness becoming less than zero. We adopt the former solution and to provide appropriate reinforcement the correct set is rewarded with a fixed reward of 1000 in this case.

3.3 Transaction Costs

In Section 2.2 it was noted that any real-world trading activity involves some form of transaction costs. In the trading model, the agent is constrained to be always in the market and there is both a position opened and a position closed each time the current position is changed. Equation 4 models this using a charge c_p to cover the cost of changing position. This proportional cost is deducted from the excess return figure for each position.

$$c_p = \frac{(1 - c)}{(1 + c)} \qquad (4)$$

For all trials $c = 0.00025$ (2.5 basis points). This value is the same as that used in the Genetic Programming approach in the validation period. Neely et al. [13] use a higher transaction cost of $c = 0.0005$ for training and selection to bias the genetic search towards rules that trade infrequently. However, as the focus of the present work is for online trading where there are no training, selection and validation periods, we use the same transaction cost for all trials.

4 Experiments

4.1 Experimental Setup

The Learning Classifier System architecture developed in [21] is retained for use in these experiments on FX trading and, as described in Section 3, an 8-bit binary encoding used for real numbers and integers.

As the population size limit N and GA invocation rate ρ are problem dependent, we ran some experiments using the USD/GBP data to determine suitable settings for these parameters. We evaluated the effects of these two parameter

Table 2. Enhanced Learning Classifier System with validation period 1/1/1981 to 30/9/1995

	USD/GBP	USD/DEM	USD/CHF	USD/JPY	DEM/JPY	GBP/CHF
APR	3.86%	0.34%	2.80%	0.86%	3.58%	-1.23%
Monthly std dev	3.44%	3.49%	3.70%	3.35%	2.75%	2.70%
Runs +ve return	94%	48%	82%	56%	96%	20%
Sharpe Ratio	0.33	0.06	0.25	0.11	0.4	-0.12
Number of trades	627.84	648.68	659.21	663.2	589.7	638.45
Long positions	52.52%	50.77%	49.34%	46.33%	47.66%	44.61%

settings using a risk-adjusted performance measure over the entire data set and across 100 runs. This metric was the mean total excess return divided by the standard deviation of total excess return across runs. These tests yielded strong indications that a high GA rate relative to the population size limit was needed and that the GA rate required increased with population size limit. Since each classifier's condition consists of only two intervals a large population is not necessary and we use $N = 50$ and $\rho = 4$ for all experiments using the enhanced Learning Classifier System.

In our parameter setting experiments we found that because the number of elements in the time series was very low compared to the problem complexity, it was advantageous for the cover operator to introduce multiple classifiers into the population. This allows the GA to test multiple hypotheses arising from a single environmental situation and provides the diversity necessary for the GA to function effectively. As a result, in these experiments the cover operator introduces eight classifiers into the population each time it is invoked.

In earlier work, we did not investigate the effect of the GA crossover rate χ, mutation rate μ and mutation range m, so these parameters were left at the values used for previous experiments, $\chi = 0.5$, $\mu = 0.002$ $m = 0.1$. The learning rate was also set to the previously used value of $\beta = 0.2$.

One hundred runs were performed of each experiment. Results are shown for all six currency pairs, but as the USD/GBP currency pair was used to set the value of two of the Learning Classifier System parameters, results for this currency pair may not be statistically valid for the reasons of data snooping presented in Section 1.

4.2   Validation Period from 1981 to 1995

We first restrict the period under which the Learning Classifier System is evaluated to 1981–1995. This matches the validation period used for the Genetic Programming approach. In this scenario, the Learning Classifier System learns using data from 1975 through to 1995, but the performance of the Learning Classifier System is only measured during the period 1981–1995. Table 2 shows the performance of the Learning Classifier System using the architecture developed

in [21] with the parameter settings detailed in Section 4.1. For comparison, Table 1 shows the results published in [13] achieved by the Genetic Programming approach.

The first line of the tables shows the Annual Percentage Rate excess return achieved during the validation period 1981–1995. The results show that the Learning Classifier System achieved a positive excess return on five out of six currency pairs. In contrast, the Genetic Programming approach achieved positive returns on all six currency pairs. This is largely attributable to the fact that in the Genetic Programming approach rules achieving a negative excess return are discarded during the selection period. This is evidenced by the percentage of rules in the Genetic Programming approach that have positive returns, which is high in all cases.

In the Genetic Programming approach one fixed rule is used per run, whereas in the Learning Classifier System approach, 50 competing adaptive rules are used per run. However, the Learning Classifier System selects only one action per trial, so we can compare the results of 100 Genetic Programming rules with 100 runs of the Learning Classifier System. The number of Learning Classifier System runs with positive excess returns correlates well with the mean excess return achieved.

Excess return in itself is insufficient to judge performance. The risk taken by a trading agent in achieving profit must also be taken into account. Measurement of risk involves consideration of volatility of returns. A simple way of doing this is to measure the monthly standard deviation of excess return over non-overlapping periods. This measure is comparable for both Genetic Programming and Learning Classifier System approaches across all currency pairs. This is explained in [13] as being because rules are required to be in the market at all times with a long or short position, which have similar variances.

$$D_t = R_{Ft} - R_{Bt}$$
$$\bar{D} = \frac{1}{T} \sum_{t=1}^{T} D_t$$
$$S_h = \frac{\bar{D}}{\sigma_D} \tag{5}$$

A metric encountered often in the finance literature is the Sharpe Ratio [17, 18]. There are several forms of the Sharpe Ratio. Here, we use the ex post Sharpe Ratio shown in Equation 5. This measures the mean differential return $\bar{D}$ achieved from the trading strategy R_F compared to that of a benchmark R_B divided by the standard deviation σ_D of differential returns over the period. In this case the benchmark R_B is the risk-free interest i received on the margin, that would accrue from a strategy of not trading. Returns are measured for non-overlapping periods of length t over a sample period T. Here, we measure annual returns over the validation period.

Sharpe Ratios for the Learning Classifier System approach reflect the distribution of returns, with a greater spread across the currencies than that seen with

Table 3. Enhanced Learning Classifier System with exploit trials in validation period 1/1/1981 to 30/9/1995

	USD/GBP	USD/DEM	USD/CHF	USD/JPY	DEM/JPY	GBP/CHF
APR	-3.05%	-3.60%	-2.60%	-0.85%	-0.19%	-2.32%
Monthly std dev	3.47%	3.46%	3.79%	3.32%	2.80%	2.75%
Runs +ve return	4%	2%	4%	41%	65%	2%
Sharpe Ratio	-0.18	-0.22	-0.11	-0.05	0.01	-0.22
Number of trades	849.55	674.91	610.85	949.33	490.79	489.53
Long positions	72.81%	16.04%	13.37%	79.45%	89.02%	12.09%

the Genetic Programming approach. Results show that the Learning Classifier System approach can achieve a higher Sharpe Ratio than the Genetic Programming approach, but that it is also possible to achieve poorer results. This may be due to the number of trades executed by the Learning Classifier System. For all currency pairs the number of trades is around 600, several times that of the Genetic Programming approach, which shows considerably more variability in trading frequency.

The number of long positions taken by the Learning Classifier System stays quite close to 50%. On the other hand, the Genetic Programming approach again shows more variability in the number of long positions. However, the Genetic Programming approach uses only a single fixed rule for each run, whereas the Learning Classifier System uses multiple adaptive rules, so we may expect that this effect would occur in an efficient market.

To determine the effects of adaptation on the results obtained, we repeated the experiment with no GA or update activity taking place during the validation period. In this situation, the Learning Classifier System trades using the fixed set of rules that were in force at the start of the validation period unless covering is necessary. To gauge the risk of the cover operator disrupting the population with random classifiers, we checked the frequency of covering during a single run of the Learning Classifier System and found only one occurrence of cover throughout the validation period.

The results of this experiment are presented in Table 3. These show that the Learning Classifier System produces a negative excess return for all currency pairs and that the number of runs with positive returns is much lower than when GA and update activity occurs. The number of long positions taken diverges considerably from 50% and shows significant variation across currencies, as does the number of trades, reflecting the composition of rules in force at the start of the validation period. However, such a snapshot of rules clearly do not provide a good trading performance over the validation period. This result supports the hypothesis that, during the validation period, adaptation of rules is occurring within the Learning Classifier System which is of benefit to the agent's performance.

Table 4. Enhanced Learning Classifier System with validation period 1/2/1975 to 30/9/1995

	USD/GBP	USD/DEM	USD/CHF	USD/JPY	DEM/JPY	GBP/CHF
APR	4.14%	0.67%	3.48%	2.92%	4.97%	0.75%
Monthly std dev	3.23%	3.31%	3.67%	3.25%	2.87%	2.92%
Runs +ve return	99%	61%	94%	89%	100%	67%
Sharpe Ratio	0.37	0.09	0.31	0.29	0.51	0.11
Number of trades	845.45	896.63	902.84	896.7	820.47	881.85
Long positions	56.00%	51.94%	49.52%	46.90%	47.02%	44.67%

4.3   Validation Period from 1975 to 1995

In the previous section, we used a validation period from 1981 to 1995 to allow a meaningful comparison of the Learning Classifier System and Genetic Programming approaches. To do this we compared the performance of 100 runs of the Learning Classifier System against that of 100 rules produced by Genetic Programming.

One significant difference between the two approaches is the number of evaluations occurring for each rule or run. In the Genetic Programming approach, the number of evaluations needed to create and test the 100 rules used is in the region 1000–2000 million. In contrast, the number of evaluations used for the 100 runs of the Learning Classifier System is in the region of only 26 million!

Of course, such comparisons are irrelevant provided that the elapsed time needed to create and test rules is less than the frequency with which new rules are needed and the rules provide some utility. Rather, it is informative to realize how much the Learning Classifier System is able to achieve when observing each daily price only once per run. Because the amount of data is rather small for efficient induction, and because the agent is intended to be used online, we now present results over the entire data set. That is, the validation period is extended to cover the period 1975[3]–1995 and the agent is now evaluated on every trading decision it makes during its lifetime.

Table 4 shows the results for the enhanced Learning Classifier System architecture. These correspond to results presented in Table 2 for the shorter validation period. Over the entire data set, the Learning Classifier System achieves a positive excess return on all currency pairs. Furthermore, for all currency pairs, excess return, Sharpe Ratio and number of runs with a positive excess return are higher than for the shorter validation period.

To assess whether the Learning Classifier System developments detailed in [21] indeed have any bearing on the results obtained, we ran a standard version of ZCS with the baseline parameters used in earlier work. These were $N = 50$, $\beta = 0.2$, $\tau = 0.1$, $\chi = 0.5$, $\mu = 0.002$, $m = 0.1$, $f_I = 20$, $s_0 = 0.25$, $\rho = 0.25$,

---

[3] The validation period actually begins on February 1, 1975 to allow for a 255-day buildup period at the start of each data set.

Table 5. ZCS with standard settings and validation period 1/2/1975 to 30/9/1995

	USD/GBP	USD/DEM	USD/CHF	USD/JPY	DEM/JPY	GBP/CHF
APR	1.29%	0.00%	0.00%	-0.20%	-1.13%	0.03%
Monthly std dev	3.43%	3.40%	3.89%	3.52%	3.09%	3.13%
Runs +ve return	97%	46%	6%	47%	33%	8%
Sharpe Ratio	0.14	0.03	0.03	0.02	-0.06	0.02
Number of trades	11.07	6.92	6.41	6.16	8.27	6.34
Long positions	93.20%	49.03%	71.49%	48.92%	42.43%	47.86%

$\phi = 0.5$. Only a single classifier was introduced with each invocation of the cover operator. Results are shown in Table 5. They show poor performance compared to that seen in Table 4. This is due to the extremely low number of trades made by ZCS.

Investigation of individual runs of this experiment revealed that lack of trading was caused by the population being taken over by classifiers advocating the same action. This is caused by the following sequence of events: Classifiers created by cover tend to be quite general and therefore take part in many match sets. Each time a classifier matches it gets taxed either by being promoted to the action set and paying a fraction $\beta = 0.2$ of its fitness into the bucket or by not making the action set and paying a tax $\tau = 0.1$ of its fitness. Because classifiers participate in many match sets, they must receive reward relatively often in order to avoid their fitness decreasing. If this does not occur, for example when a classifier makes a few incorrect predictions, fitness decreases quite rapidly. Under roulette wheel action selection, these classifiers are less likely to be selected for the action set and they become increasingly likely to remain in the match set, from which they cannot obtain reward. In this way a gap opens between the relative fitness of classifiers advocating opposing actions.

The problem is compounded by the low frequency of invocation of the GA relative to the frequency of matching and consequent fitness decay. Without frequent GA activity, it is possible for classifiers to have their fitness reduced to almost zero. When the GA does operate, the roulette selection mechanism used favours selection of the stronger action. If two classifiers selected for reproduction both have the same action, crossover will produce offspring with that action, which can only be changed with low probability by mutation. Under the GA, the stronger action is propagated within the population. Within a relatively few trials it is possible for all of the classifiers suggesting the weaker action to have such a low fitness that there is little realistic chance of them reproducing. Over a longer time period and within the 5000 or so trials of these data sets, replacement seals the fate of the weaker species and they are totally eradicated from the population which then consists solely of classifiers advocating the same action.

We repeated the experiment with a setting of $s_0 = 0.05$ to cause the cover operator to introduce narrower intervals into the population. Though trading

Table 6. Enhanced Learning Classifier System with random position and validation period 1/2/1975 to 30/9/1995

	USD/GBP	USD/DEM	USD/CHF	USD/JPY	DEM/JPY	GBP/CHF
APR	-6.77%	-6.02%	-5.89%	-6.26%	-6.13%	-6.21%
Monthly std dev	3.10%	3.10%	3.57%	2.93%	2.57%	2.75%
Runs +ve return	0%	0%	0%	0%	0%	0%
Sharpe Ratio	-0.67	-0.59	-0.5	-0.65	-0.72	-0.67
Number of trades	2621.02	2615.07	2618.78	2625.63	2618.65	2619.91
Long positions	50.00%	50.00%	50.02%	49.94%	50.03%	49.98%

Table 7. Enhanced Learning Classifier System with fixed short position and validation period 1/2/1975 to 30/9/1995

	USD/GBP	USD/DEM	USD/CHF	USD/JPY	DEM/JPY	GBP/CHF
APR	-1.63%	-1.75%	-1.24%	-4.82%	-2.91%	-0.14%
Monthly std dev	3.43%	3.41%	3.89%	3.48%	3.08%	3.11%
Runs +ve return	0%	0%	0%	0%	0%	0%
Sharpe Ratio	-0.03	-0.05	0.01	-0.28	-0.18	0.06
Number of trades	2	2	2	2	2	2
Long positions	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

frequency increased marginally, the change did not prevent population takeover by classifiers of the same action. It is of note that this situation cannot occur with the NewBoole update since the entire match set is updated on every trial, not just members of the action set as is the case with the ZCS update algorithm.

To provide further control experiments we tested versions of the enhanced Learning Classifier System used to provide the results in Table 4 with the exception that the position chosen was random or a fixed short or fixed long position. The results of these experiments are shown in Table 6 to Table 8 respectively.

A control strategy of choosing positions at random performs especially badly. This is no doubt due at least in part to the large number of trades and consequent

Table 8. Enhanced Learning Classifier System with fixed long position and validation period 1/2/1975 to 30/9/1995

	USD/GBP	USD/DEM	USD/CHF	USD/JPY	DEM/JPY	GBP/CHF
APR	0.48%	0.59%	-0.28%	3.98%	2.17%	-0.78%
Monthly std dev	3.43%	3.39%	3.89%	3.53%	3.10%	3.15%
Runs +ve return	100%	100%	0%	100%	100%	0%
Sharpe Ratio	0.1	0.1	0.04	0.34	0.24	-0.02
Number of trades	2	2	2	2	2	2
Long positions	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

transaction costs incurred in trading. The fixed short position provides negative excess return on all runs and is clearly a poor strategy. However, the fixed long position does surprisingly well for a naive strategy and provides a positive excess return on four of the currency pairs. Even so, the enhanced Learning Classifier System exceeds the performance of a fixed long position on five of the six currency pairs measured in terms of excess return and on four of the currency pairs in terms of Sharpe Ratio.

5   Conclusions

We have shown that a simple Learning Classifier System architecture, properly configured, is able to achieve a positive excess return in simulated trading. Although results are not yet fully competitive with those obtained from Genetic Programming, the Learning Classifier System approach is attractive because these positive returns have been obtained without any offline training. Furthermore, the technique is inherently adaptive unlike many of the machine learning methods currently employed for financial trading.

Trading frequency of the Learning Classifier System approach was much higher than that seen with Genetic Programming and this is almost certainly the factor limiting performance compared to the Genetic Programming approach. One area worthy of further study is to see whether trading frequency can be lowered to reduce the effect of transaction costs on the returns achieved. It would appear that this may be difficult using a single-step model as is used in the present work. This is because with a single-step model successive trials are independent and there is no direct method for the Learning Classifier System to control trading frequency, which is a function of the actions chosen in successive trials. Thus, it may be necessary to extend the agent architecture to model the trading activity as a multi-step problem to achieve any further control over trading frequency.

The availability of information and communications technologies that were not available until recently means that such returns may be difficult or impossible to achieve in live trading. For this reason we do not claim that the architecture or parameter settings used are in any way optimal or final. Rather we see the results presented here as providing evidence of the usefulness and potential of Learning Classifier Systems for financial trading. In this respect, these results echo those of Schulenburg and Ross [14, 15, 16], who also reported encouraging results from a simple Learning Classifier System architecture.

In the interests of comparing the Learning Classifier approach with that of Genetic Programming, we adopted a similar experimental setup. In particular, we provided the same sensory information to the agent as was used for the Genetic Programming approach, namely the exchange rate time series. Interest rate information was used only to calculate excess return and was not available to the agent for use when selecting a position. Yet, the differential between the interest rates received for the base and counter currencies forms an important part of the overall excess return achieved by the agent. It is quite possible for

there to be a large interest rate differential between the two currencies in a pair, which could dominate a trading decision, but with the experimental setup currently in use the agent does not have this information. It would seem that including such interest rate information as additional sensory input would be beneficial to performance.

Similarly, for transaction costs and slippage to be included in a trading agent's model of the environment, the environment should include a sensor representing the agent's current position. Without these changes the environment is partially observable, since the same apparent environmental state may result in differing rewards depending on the current interest rate differential and the position held by the agent in the previous time step.

Finally, the Learning Classifier System approach is well suited to enhancement through an ensemble approach, which should aid performance. The low computational cost of an individual Learning Classifier System means that it is realistic to run many individual agents in parallel and still achieve high throughput. This is an important consideration when using high frequency data, as is necessary for a realistic trading model where near real-time decisions must be made. These topics, however, are beyond the scope of the present work and will be the subject of future developments.

Acknowledgements

Bibliography

[1] M. Ahluwalia and L. Bull. A Genetic Programming-based Classifier System. In Banzhaf et al. [3], pages 11–18.

[2] F. Allen and R. Karjalainen. Using Genetic Algorithms to find technical trading rules. *Journal of Financial Economics*, 51(2):245–271, 1999.

[3] W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, CA, 1999. Morgan Kaufmann.

[4] P. Bonelli, A. Parodi, S. Sen, and S. W. Wilson. NEWBOOLE: A fast GBML system. In B. W. Porter and R. J. Mooney, editors, *Proceedings of the Seventh International Conference on Machine Learning*, pages 153–159, San Mateo, CA, 1990. Morgan Kaufmann.

[5] M. M. Dacorogna, R. Gençay, U. Müller, R. B. Olsen, and O. V. Pictet. *An Introduction to High-Frequency Finance*. Academic Press, San Diego, 2001.

[6] M. A. H. Dempster, T. W. Payne, and Y. S. Romahi. Intraday FX trading: Reinforcement vs evolutionary learning. Judge Institute of Management Working Paper 23/2001, University of Cambridge, Cambridge, UK., 2001.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press. Republished by MIT Press, 1992, Ann Arbor, MI, 1975.

[8] J. H. Holland. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning, an Artificial Intelligence Approach. Volume II*, pages 593–623, Los Altos, CA, 1986. Morgan Kauffmann.

[9] D. Jensen. Data snooping, dredging and fishing: The dark side of data mining a SIGKDD99 panel report. *SIGKDD Explorations*, 1(2):52–54, 2000.

[10] J. R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.

[11] Y.-K. Kwon and B.-R. Moon. Daily stock prediction using neuro-genetic hybrids. In E. Cantú-Paz, J. A. Foster, K. Deb, L. D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation - GECCO-2003. Part 2*, volume LNCS-2724 of *Lecture Notes in Computer Science*, pages 2203–2214, Berlin, 2003. Springer.

[12] P. L. Lanzi. Extending the representation of classifier conditions, part ii: From messy coding to s-expressions. In Banzhaf et al. [3], pages 345–352.

[13] C. Neely, P. Weller, and R. Dittmar. Is technical analysis in the foreign exchange market profitable? a Genetic Programming approach. *Journal of Financial and Quantitative Analysis*, 32(4):405–427, 1997.

[14] S. Schulenburg and P. Ross. An adaptive agent based economic model. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Learning Classifier*

Systems. From Foundations to Applications, volume LNAI-1813 of *Lecture Notes in Artificial Intelligence*, pages 265–284, Berlin, 2000. Springer.

[15] S. Schulenburg and P. Ross. Strength and money: An LCS approach to increasing returns. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems. Proceedings of the Third International Workshop (IWLCS-2000)*, volume LNAI-1996 of *Lecture Notes in Artificial Intelligence*, pages 114–137, Berlin, 2001. Springer.

[16] S. Schulenburg and P. Ross. Explorations in LCS models of stock trading. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems. Proceedings of the Fourth International Workshop (IWLCS-2001)*, volume LNAI-2321 of *Lecture Notes in Artificial Intelligence*, pages 151–180, Berlin, 2002. Springer.

[17] W. F. Sharpe. Mutual fund performance. *Journal of Business*, 39, 1966.

[18] W. F. Sharpe. The Sharpe Ratio. *Journal of Portfolio Management*, 21, 1994.

[19] C. Stone and L. Bull. For real! XCS with continuous-valued inputs. *Evolutionary Computation*, 11(3):299–336, 2003.

[20] C. Stone and L. Bull. Comparing XCS and ZCS on noisy continuous-valued environments. Technical Report UWELCSG05-002, University of the West of England Learning Classifier Group, Bristol, UK, 2005. http://www.cems.uwe.ac.uk/lcsg/reports/uwelcsg05-002.ps.

[21] C. Stone and L. Bull. Configuring ZCS for continuous-valued single-step Boolean problems. Technical Report UWELCSG05-006, University of the West of England Learning Classifier Group, Bristol, UK, 2005. http://www.cems.uwe.ac.uk/lcsg/reports/uwelcsg05-006.ps.

[22] R. Sullivan, A. Timmermann, and H. White. Data-snooping, technical trading rule performance and the bootstrap. Disussion Paper 97-31, Department of Economics, University of California, San Diego, 1997.

[23] S. W. Wilson. ZCS: A zeroth order classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.