November 3, 2000

COMPUTATIONAL LEARNING TECHNIQUES FOR INTRADAY FX TRADING USING POPULAR TECHNICAL INDICATORS

M.A.H. DEMPSTER, T.W. PAYNE, Y.S. ROMAHI AND G.W.P. THOMPSON
CENTRE FOR FINANCIAL RESEARCH
JUDGE INSTITUTE OF MANAGEMENT
UNIVERSITY OF CAMBRIDGE
EMAIL: {MAHD2,TWP20,YR206,GWPT1}@CAM.AC.UK
WEB: WWW-CFR.JIMS.CAM.AC.UK

There is reliable evidence that technical analysis, as used by traders in the foreign exchange markets, has predictive value regarding future movements of foreign exchange prices. Although the use of AI-based trading algorithms has been an active research area over the last decade, there have been relatively few applications to intraday foreign exchange—the trading frequency at which technical analysis is most commonly used. Previous academic studies have concentrated on testing popular trading rules in isolation or have used a genetic algorithm approach to construct new rules in an attempt to make positive out-of-sample profits after transaction costs. In this paper we consider strategies which use a collection of popular technical indicators as input and seek a profitable trading rule defined in terms of them. We consider two computational learning approaches, reinforcement learning and genetic programming, and compare them to a pair of simpler methods: the exact solution of an appropriate Markov decision problem and a simple heuristic. We find that although all methods are able to generate significant in-sample and out-of-sample profits when transaction costs are zero, the genetic algorithm approach is superior for non-zero transaction costs, although none of the methods produce significant profits at realistic transaction costs. We also find that there is a significant danger of overfitting if in-sample learning is not constrained.

1 Introduction

Since the era of floating exchange rates began in the early 1970s, technical trading has become widespread in the *foreign exchange* (FX) markets. Academic investigation of technical trading however has largely limited itself to daily data. Although daily data is often used for currency overlay strategies within an asset-allocation framework, FX traders trading continuously throughout the day naturally use higher frequency data.

In this investigation, the relative performance of various optimization techniques in high frequency (intraday) foreign exchange trading is examined. We compare the performance of a *genetic algorithm* (GA) and a *reinforcement learning* (RL) system to a simple *linear program*

(LP) characterising a *Markov decision process* (MDP) and a *heuristic*.

In Section 2 we give a brief literature review of preceding work in technical analysis. Sections 3 and 4 then introduce the GA and RL methods. The *stochastic optimization* problem to be solved by all the compared methods is defined in Section 5, while Sections 6, 7 and 8, describe in more detail how each approach can be applied to solve this optimization problem approximately. The computational experiments performed are outlined and their results given in Section 9. Section 10 concludes with a discussion of these results and suggests further avenues of research.

Reinforcement learning has to date received only limited attention in the financial literature and this paper demonstrates that RL methods show significant promise. The results also indicate that generalization and incorporation of constraints limiting the ability of the algorithms to overfit improves out-of-sample performance, as is demonstrated here by the genetic algorithm.

2 Technical Analysis

Technical analysis has a century-long history amongst investment professionals. However, academics have tended to regard it with a high degree of scepticism over the past few decades largely due to their belief in the *efficient markets* or *random walk* hypothesis. Proponents of technical analysis had until very recently never made serious attempts to test the predictability of the various techniques used and as a result the field has remained marginalised in the academic literature.

However due to accumulating evidence that markets are less efficient than was originally believed (see for example [1]), there has been a recent resurgence of academic interest in the claims of technical analysis. Lo and MacKinlay [2, 3] have shown that past prices may be used to forecast future returns to some degree and thus reject the random walk hypothesis for US stock indices sampled weekly.

LeBaron [1] acknowledges the risk of bias in this research however. Since various rules are applied and only the successful ones are reported, he notes that it is not clear whether their returns could have been attained by a trader who had to make the *choice* of rules in the first place. LeBaron argues that to avoid this bias, it is best simply to look at rules that are both widely used and have been in use for a long period of time. Neely *et al.* [4] use a genetic programming based approach to avoid this bias and found out-of-sample net returns

in the 1-7% per annum range in currency markets against the dollar during 1981-1995.

Although there has been a significant amount of work in technical analysis, most of this has been based on stock market data. However, since the early 1970s this approach to trading has been widely adopted by foreign currency traders [4]. A survey by Taylor and Allen [5] found that in intraday trading 90% of respondents reported the use of technical analysis, with 60% stating that they regarded such information as at least as important as economic fundamentals. Neely *et al.* [4] argue that this can be partly explained by the unsatisfactory performance of exchange rate models based on economic fundamentals. They cite Frankel and Rose [6] who state that

> ... no model based on such standard fundamentals like money supplies, real income, interest rates, and current-account balances will ever succeed in explaining or predicting a high percentage of the variation in the exchange rate, at least at short or medium-term frequencies.

A number of researchers have examined net returns due to various trading rules in the foreign exchange markets [7, 8]. The general conclusion is that trading rules are sometimes able to earn significant returns net of transaction costs and that this cannot be easily explained as compensation for bearing risk. Neely and Weller [9] note however that academic investigation of technical trading has not been consistent with the practice of technical analysis. As noted above, technical trading is most popular in the foreign exchange markets where the majority of intraday foreign exchange traders consider themselves technical traders. They trade throughout the day using high-frequency data but aim to end the day with a net open position of zero. This is in contrast to much of the academic literature which has tended to take much longer horizons into account and only consider daily closing prices.

Goodhart and O'Hara [10] provide a thorough survey of past work investigating the statistical properties of high frequency trading data, which has tended to look only at narrow classes of rules. Goodhart and Curcio [11] examine the usefulness of resistance levels published by Reuters and also examine the performance of various filter rules identified by practitioners. Dempster and Jones [12, 13] examine profitability of the systematic application of the popular channel and head-and-shoulders patterns to intraday FX trading at various frequencies, including with an overlay of statistically derived filtering rules. In subsequent work [14, 15] upon which this paper expands, they apply a variety of technical trading rules to trade such data (see also Tan [16]) and also study a genetic program which trades combinations of these rules on the same data [17]. None of these studies report any evidence of *significant* profit oppor-

tunities, but by focussing on relatively narrow classes of rules their results do not necessarily exclude the possibility that a search over a broader class would reveal profitable strategies. Gencay *et al.* [18] in fact assert that simple trading models are able to earn significant returns after transaction costs in various foreign exchange markets using high frequency data.

3 Genetic Algorithms

In recent years, the application of *artificial intelligence* (AI) techniques to technical trading and finance has experienced significant growth. *Neural networks* have received the most attention in the past and have shown varying degrees of success. However recently there has been a shift in favour of user-transparent, non-black box evolutionary methods like genetic algorithms and in particular genetic programming. An increasing amount of attention in the last several years has been spent on these genetic approaches which have found financial applications in option pricing [19, 20] and as an optimization tool in technical trading applications [17, 14, 4].

Evolutionary learning encompasses sets of algorithms that are inspired by Darwinian evolution. *Genetic algorithms* (GAs) are population based optimization algorithms first proposed by Holland [21]. They have since become an active research area within the artificial intelligence community and have been successfully applied to a broad range of hard problems. Their success is in part due to their several control parameters that allow them to be highly tuned to the specific problem at hand. *Genetic programming* (GP) is an extension proposed by Koza [22] whose original goal was to evolve computer programs.

Pictet *et al.* [23] employ a GA to optimize a class of exponentially weighted moving average rules, but run into serious overfitting and poor out-of-sample performance. They report 3.6% to 9.6% annual excess returns net of transaction costs, but as the models of Olsen and Associates are not publicly available their results are difficult to evaluate. Neely and Weller [9] report that for their GA approach, although strong evidence of predictability in the data is measured out-of-sample when transaction costs are set to zero, no evidence of profitable trading opportunities arise when transaction costs are applied and trading is restricted to times of high market activity.

4 Reinforcement Learning

Reinforcement learning has to date received only limited attention in financial applications. The reinforcement learning technique is strongly influenced by the theory of *Markov decision processes* (MDPs) which evolved from attempts to understand the problem of making sequences of decisions under uncertainty when each decision can depend on the previous decisions and their outcomes. The last decade has witnessed the merging of ideas from the reinforcement learning and the control theory communities [24]. This has expanded the scope of *dynamic programming* and allowed the approximate solution of problems that were previously considered intractable.

Although reinforcement learning was developed independently of MDPs, the integration of these ideas with the theory of MDPs brought a new dimension to RL. Watkins [25] was instrumental in this advance by devising the method of *Q-learning* for estimating action-value functions. The nature of reinforcement learning makes it possible to approximate optimal policies in ways that put more effort into learning to make good decisions for frequently encountered situations at the expense of less effort for less frequently encountered situations [26]. This is a key property which distinguishes reinforcement learning from other approaches for approximate solution of MDPs.

As fundamental research in reinforcement learning advances, applications to finance have started to emerge. Moody *et al.* [27] examine a *recurrent reinforcement learning* algorithm that seeks to optimize an online estimate of the Sharpe ratio. They also compare the recurrent RL approach to that of Q-learning.

5 Applying optimization methods to technical trading

In this paper, following [15, 17, 14], we consider trading rules defined in terms of 8 popular *technical indicators* used by intraday foreign exchange traders. They include both buy and sell signals based on simple trend-detecting techniques such as moving averages as well as more complex rules. The indicators we use are the Price Channel Breakout, Adaptive Moving Average, Relative Strength Index, Stochastics, Moving Average Convergence/Divergence, Moving Average Crossover, Momentum Oscillator and Commodity Channel Index. A complete algorithmic description of these indicators can be found in [15, 14].

To define the indicators, we first aggregate the raw tick data into (here) quarter-hourly intervals, and for each compute the *bar* data — the *open*, *close*, *high* and *low* FX rates —

and the total *trading volume*. Most of the indicators use only the closing price of each bar, so we will introduce the notation $\boldsymbol{F}_t$ to denote the *closing* GBP:USD FX *rate* (i.e. the dollar value of £1) of bar t (here we use boldface to indicate random entities).

We define the *market state* s_t at time t as the binary string of length 16 giving the *buy* and *sell pounds* indications of the 8 indicators, and define the *state space* $\mathcal{S} = \{0,1\}^{16}$ as the set of all possible market states. Here a 1 represents a trading recommendation for an individual indicator whose entry is otherwise 0. In effect we have constructed from the available tick data a discrete time data series: at time t (the end of the bar t interval) we see F_t, compute s_t and must choose whether or not to switch currencies based on the values of the indicators incorporated in s_t and which currency is currently held. We consider this time series to be a realization of a binary string valued *stochastic process* and make the required trading decisions by solving an appropriate *stochastic optimization problem*.

Formally, a *trading strategy* ϕ is a function $\phi : \mathcal{S} \times \{0,1\} \to \{0,1\}$, $(s,h) \mapsto \phi(s)$, for some *current position* h (= 0, dollars or 1, pounds), telling us whether we should hold pounds ($\phi = 1$) or dollars ($\phi = 0$) over the next timestep. It should be noted that although our trading strategies ϕ are formally *Markovian* (feedback rules), some of our technical indicators require a number of periods of previous values of F to decide the corresponding 0-1 entries in s_t. The objective of the trading strategies ϕ used in this paper is to maximize the expected dollar return (after transaction costs) up to some *horizon* T:

$$\mathbb{E}\left\{\prod_{t=1}^{T-1}\left(\frac{\boldsymbol{F}_{t+1}}{\boldsymbol{F}_t}\right)^{\phi(\boldsymbol{s}_t)}(1-c)^{|\phi(\boldsymbol{s}_t)-\phi(\boldsymbol{s}_{t-1})|}\right\}, \tag{1}$$

where $\mathbb{E}$ denotes *expectation*, c is the proportional *transaction cost*, and ϕ is chosen with the understanding that trading strategies start in dollars, observe s_1 and then have the opportunity to switch to pounds. Since we do not have an explicit probabilistic model for how FX rates evolve, we cannot perform the expectation calculation in (1), but instead adopt the familiar approach of dividing our data series into an *in-sample* region, over which we optimize the performance of a candidate trading strategy, and an *out-of-sample* region where the strategy is ultimately tested.

The different approaches utilised solve slightly different versions of the in-sample optimization problem. The simple heuristic and Markov Chain method methods find a rule which takes as input a market state and outputs one of three possible actions: either 'hold pounds', 'hold dollars' (switching currencies if necessary) or 'stay in the same currency'.

The GA and RL approaches find a rule which takes as input the market state and the

currency currently held, and chooses between two actions: either to stay in the same currency or switch. Thus the RL and GA method are given slightly more information (their current position) than the heuristic and MDP methods and we might thus expect them to perform better. The GA method also has an extra constraint restricting the complexity of the rules it can generate which is intended to stop overfitting of the in-sample data.

6 Applying RL to the Technical Trading Problem

The ultimate goal of reinforcement learning based trading systems is to optimize some relevant measure of trading system performance such as profit, economic utility or risk-adjusted return. A standard RL framework has two central components; an *agent* and an *environment*. The agent is the learner and decision maker that interacts with the environment. The environment consists of a set of *states* and available *actions* for the agent in each state.

The agent is bound to the environment through *perception* and action. At a given time step t the agent receives input i, which is representative of some state $s_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states in the environment. As mentioned in the previous section, s_t is defined here as being a combination of the technical indicator buy and sell pounds decisions prepended to the current state of the agent (0 for holding dollars and 1 for pounds). The agent then selects an action $a_t \in \mathcal{A}$ where $\mathcal{A} := \{0, 1\}$ telling it to hold pounds ($\phi = 1$) or dollars ($\phi = 0$) over the next timestep. This selection is determined by the agent's *policy* π ($:= \phi$, i.e. defined in our case as the *trading strategy*) which is a mapping from states to probabilities of selecting each of the possible actions.

For learning to occur while iteratively improving the trading strategy (policy) over passes of the in-sample data, the agent needs a *merit function* that it seeks to improve. In RL, this is a function of *expected return* R which is the amount of return the agent expects to get in the future as a result of moving forward from the current state. At each *learning episode* for every time-step t the value of the last transition is communicated to the agent by an immediate reward in the form of a scalar *reinforcement signal* r_t. The expected return from a state is therefore defined as

$$\begin{aligned} R_t &:= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{T-1} r_T \\ &= \sum_{k=0}^{T} \gamma^k r_{t+k+1}, \end{aligned} \tag{2}$$

where γ is the *discount factor* and T is the final time step. Note that the parameter γ

determines the "far-sightedness" of the agent. If $\gamma = 0$ then $R_t = r_{t+1}$ and the agent myopically tries to maximize reward only at the next time-step. Conversely, as $\gamma \to 1$ the agent must consider rewards over an increasing number of future time steps to the horizon. The goal of the agent is to learn over a large number of episodes a policy mapping of $\mathcal{S} \to A$ which maximizes R_t for all $t = 0, \ldots, T$ as the limit of the approximations obtained from the same states at the previous episode.

In our implementation, two different approaches to rewards were followed. The first one we term *one way reward*. In this case, the agent is directly attempting to maximize (1). The reward signal is therefore equivalent to actual returns achieved from each state at the previous episode. This implies that whenever the agent remains in the base currency, regardless of what happens to the FX rate, the agent is neither rewarded nor penalized.

Often RL problems have a simple goal in the form of a *single state* which when attained communicates a fixed reward and has the effect of *delaying* rewards from the current time period of each learning episode. Maes and Brookes [28] show that *immediate* rewards are most effective — when they are feasible. RL problems can in fact be formulated with *separate* state spaces and reinforcement rewards in order to leave less of a temporal gap between performance and rewards. In particular it has been shown that continuity of rewards lead to effective learning. Matarič [29] demonstrates the effectiveness of *multiple* goals and progress estimators, for example, a reward function which provides instantaneously positive and negative rewards based upon "immediate measurable progress relative to specific goals".

For this reason we initially introduced a second *two way reward* approach in which the agent is rewarded or penalized by the amount the currency currently held moved against the other currency (minus transaction costs). In this approach the reward is given by

$$r_{t+1} := \left(\frac{F_{t+1}}{F_t}\right)(1-c)^{|\phi(s_t)-\phi(s_{t-1})|}. \tag{3}$$

Thus rewards are given at every time step in which a decision is made that is deemed correct by the environment, regardless of the actual monetary effect it has on the agent. Similarly every incorrect decision incurs a negative reward regardless of whether or not the agent would have actually lost money by taking that decision. Unfortunately however, by imposing this added constraint the algorithm is encouraged to trade too often. As a result, although it performed well at low slippage values, the two way reward variant was unable to infer successful trading strategies at more realistic slippage values. As the one way reward RL method consistently outperformed the two way reward method, we will only report results from the former in the sequel. However, in a trading environment in which a trader is allowed to bet an equivalent

limit amount in either currency, the two way reward method might be expected to be superior to the one way RL algorithm used here.

In reinforcement learning the link between the agent and the environment in which learning occurs is the *value function* V^π. Its value for a given state is a measure of how "good" it is for an agent to be in that state as given by the total expected future reward from that state under policy π. Note that since the agent's policy π determines the choice of actions subsequent to a state, the value function evaluated at a state must depend on that policy. Moreover, for any two policies π and π' we say that π is *preferred* to π', written $\pi > \pi'$, if and only if $\forall s \in S$, $V^\pi(s) \geq V^{\pi'}(s)$. Under suitable technical conditions there will always be at least one policy that is at least as good as all other policies. Such a policy π^* is called an *optimal policy* and is the target of any learning agent within the RL paradigm. To all optimal policies is associated the *optimal value function* V^*, which can be defined in terms of a *dynamic programming* recursion as

$$V^*(s) = \max_a \mathbb{E}\{\boldsymbol{r}_{t+1} + \gamma V^*(\boldsymbol{s}_{t+1}) | \boldsymbol{s}_t = s\}. \tag{4}$$

Another way to characterize the value of a state s is to consider it in terms of the values of all the actions a that can be taken from that state assuming that policy π is followed subsequently. This value Q^* is referred to as the *Q-value* and is given by

$$Q^*(s,a) = \mathbb{E}\{\boldsymbol{r}_{t+1} + \gamma \max_{a'} Q^*(\boldsymbol{s}_{t+1}, a') | \boldsymbol{s}_t = s, \boldsymbol{a}_t = a\}. \tag{5}$$

The optimal value function expresses the obvious fact that the value of a state under an optimal policy must equal the expected return for the best action from that state, i.e.

$$V^*(s) = \max_a Q^*(s,a).$$

The functions Q^* and V^* provide the basis for learning algorithms for MDPs.

Q-learning [25] was one of the most important breakthroughs in the reinforcement learning literature [26]. In this method, the *learned action-value function* Q directly approximates the optimal action-value function Q^* independent of the policy being followed. This dramatically simplifies the analysis of the algorithm and enables convergence proofs. As a bootstrapping approach, Q-learning estimates the Q-value function of the problem based on estimates at the previous learning episode. The Q-learning *update* is the backward recursion

$$Q(s_t, a_t) \leftarrow Q(s_{t_c}, a_{t_c}) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_{t_c}, a_{t_c})], \tag{6}$$

where the current state-action pair $(s_t, a_t) = (s_{t_c}, a_{t_c})$ from the previous learning episode. At each iteration (episode) of the learning algorithm, the action-value pairs associated with all the states are updated and over a large number of iterations their values converge to optimality for (5). We note that there are some parameters in (6): in particular, the *learning rate* α refers to the extent with which we update the current Q-factor based on future rewards, γ refers to how 'far-sighted' the agent is and a final parameter of the algorithm is the *policy* followed in choosing the potential action at each time step. Q-learning has been proven to converge to the optimal policy regardless of the policy actually used in the training period [25]. We find that following a random policy while training yields the best results.

In order for the algorithm to converge, the learning rate α must be set to decrease over the course of learning episodes. Thus α has been initially set to 0.15 and converges downwards to 0.00015 at a rate of $\frac{0.15}{(1+\frac{E}{10})}$, where E is the episode (iteration) number which runs from 0 to 10000. The parameter γ has been set to 0.9999 so that each state has full sight of future rewards in order to allow faster convergence to the optimal.

With this RL approach we might expect to be able to outperform all the other approaches on the in-sample data set. However on the out-of-sample data set, in particular at higher slippage values, we suspect that some form of *generalization* of the input space would lead to more successful performance.

7 Applying the genetic algorithm

The approach chosen extends the genetic programming work initiated in [17, 14]. It is based on the premise that practitioners typically base their decisions on a variety of technical signals, which process is formalized by a *trading rule*. Such a rule takes as input a number of technical indicators and generates a recommended position (long £, neutral, or long $). The agent applies the rule at each timestep and executes a trade if the rule recommends a different position to the current one.

Potential rules are constructed as binary trees in which the terminal nodes are one of our 16 indicators yielding a Boolean signal at each timestep and the non-terminal nodes are the Boolean operators `AND`, `OR`, and `XOR`. The rule is evaluated recursively. The *value* of a *terminal* node is the state of the associated indicator at the current time; and the *value* of a *non-terminal* node is the associated Boolean function applied to its two children. The overall *value* of the *rule* is the value of the root node. An overall rule value of one (true) is

interpreted as a recommended long £ position and zero (false) is taken as a recommended neutral position. Rules are limited to a maximum depth of four (i.e. a maximum of sixteen terminals) to limit complexity. An example rule is shown in Figure 1. This definition of a rule generalizes that used in [17, 14] which allows trees in the comb form of Figure 1, but to depth 10.

The *fitness score* of such a rule ϕ is defined as the total return cumulated over the appropriate data period (*cf.* (1)), i.e.

$$\prod_{t=1}^{T-1} \left(\frac{F_{t+1}}{F_t} \right)^{\phi(s_t)} (1-c)^{|\phi(s_t)-\phi(s_{t-1})|}. \tag{7}$$

The genetic algorithm is used to search the space of all such rules and is tuned to favor rules that trade successfully (i.e. achieve high fitness scores) in the in-sample training period. An initial population of 250 rules is randomly generated and each rule is evaluated according to (7). A new population of rules is generated from this in which high scoring rules are preferred to low scoring rules. This bias means that the fitness scores of the new population should be greater than those of the old population. New rules are generated by two processes: crossover and mutation.

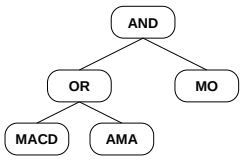

Figure 1: An example rule

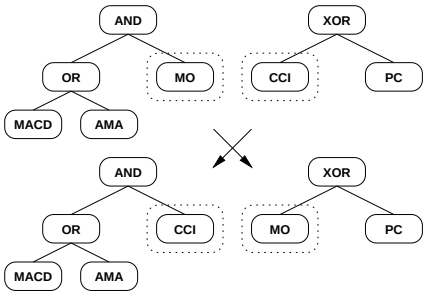

Figure 2: Genetic algorithm crossover

To use *crossover* two parent rules are selected from the current population. The selection process is biased towards fitter (better performing) rules: all rules in the current population are ranked in order of fitness score, and are chosen with a probability linearly proportional to their rank. A random subtree is chosen from each parent rule and these two subtrees are

swapped between the parent rules to create two new rules, each of which inherits characteristics from both parents. This process is shown in Figure 2. The two new rules are then inserted into the new population. For *mutation*, a single rule is selected from the current population (again biased towards the better performers) and a random node is replaced with a random node of the same type (e.g. an `AND` might become an `OR`). The mutated rule is then inserted into the new population.

New rules are generated using 75% crossover and 25% mutation until a total of 250 new rules are generated, when the new population is evaluated for fitness scores. The average score of the new population should be greater than that of the old due to the favoritism shown to the better performing rules in the old population. This process is repeated 100 times (*generations*) and the best rule found during the entire run is selected as the *output* of the genetic algorithm.

The rules found by this process exhibit a number of desirable properties. First, with careful tuning of the GA they should perform well in-sample. Secondly, a rule can be understood by a humans: it is clear what the rule does, even if it is not clear why it does it. Thirdly, this structure limits (but does not prevent) how much a rule can learn in detail (i.e. *overfit*) the training data set. It is to be expected that this enforced generalization will lead to better out-of-sample performance with a possible reduction in in-sample performance.

8 Markov Chain and Simple Heuristic

In addition to the RL and GA methods we will consider two alternative approaches. The first replaces the in-sample dataset with a *Markov chain* on a small set of market states and replaces the problem of maximizing the profit made over the in-sample period with that of maximizing a total expected discounted return assuming Markov dynamics. This approach is described in detail in Subsection 8.1

The second method is a simple *heuristic*: with each state we associate a number (which will be interpreted as the expected rise in the exchange FX rate over the next ℓ trading periods for some ℓ) and consider strategies which buy pounds if this number exceeds one threshold, and sell pounds when it falls below a second threshold. We then optimize over ℓ and the two thresholds to maximize the in-sample profit. More details on this method are given in Subsection 8.2.

These two methods were used to benchmark the success of the true computational learning

approaches in maximizing the in-sample profit. Neither are likely to attain the true optimum, but they should perform reasonably well. The heuristic was more successful at solving the in-sample problem with non-zero transaction costs than any of the other approaches, although it does not perform particularly well out-of-sample.

8.1 A Markov-chain linear programming approximation

Recall that s_t denotes the market-state at time t (the values of the indicator recommendations on which the trading decision at time t is based). In this section we will let $\bar{\mathcal{S}}$ denote the set of all market states which occur in the in-sample dataset.

We will construct a *controlled Markov chain* on the set of pairs (s, h) where $s \in \bar{\mathcal{S}}$ denotes a market state present in the in-sample dataset $\bar{\mathcal{S}} = \cup_{1 \leq t \leq T} \{s_t\}$ and $h \in \{0, 1\}$ indicates whether or not pounds are currently held. The *controls* at each timestep are 0 or 1, indicating the currency we wish to hold over the next timestep as before.

Denoting by $N(s_1)$ the number of times state s_1 appears in the in-sample dataset and by $N(s_1, s_2)$ the number of times state s_1 is followed immediately followed by state s_2, we define a *controlled Markov chain* $\boldsymbol{X}_t$, $t = 1, 2, \ldots$, by fixing $\boldsymbol{X}_1 := (s_1, 0)$ and choosing the probability of a transition from (s_1, h_1) to (s_2, h_2) using control k to be

$$P^{(k)}((s_1, h_1), (s_2, h_2)) := I(h_2 = k) N(s_1, s_2) \Big/ \sum_{s \in \bar{\mathcal{S}}} N(s_1, s).$$

For $s \in \bar{\mathcal{S}}$, $h \in \{0, 1\}$ we define an approximation to the *expected dollar return* over the next timestep given we are in state s and hold currency h as

$$R(s, h) := I(h = 1) \frac{1}{N(s)} \sum_{1 \leq t < T : s_t = s} \log\left(\frac{F_{t+1}}{F_t}\right).$$

We are now in a position to replace the problem of maximizing the in-sample return with the problem

$$\max_{\pi} \mathbb{E} \sum_{t=1}^{\infty} \gamma^{t-1} \left[R(\boldsymbol{X}_t, \pi(\boldsymbol{X}_t)) + |\pi(\boldsymbol{X}_t) - \pi(\boldsymbol{X}_{t-1})| \log(1 - c) \right] \tag{8}$$

where π is a *feedback map* from the state-space of the Markov Chain to the set of controls $\{0, 1\}$ and we treat the term $\pi(X_0)$ as zero (since we must start in dollars). The constant γ is a discount factor, chosen arbitrarily to be equal to 0.9999. This is an approximation to the objective (1).

Since the set $\bar{\mathcal{S}}$ is quite small, this problem can be solved exactly using linear programming. If $J(s, h)$ denotes the value of problem (8) when the initial market state is s and our initial

currency is h, we can solve the problem above by solving the *linear programme:*

$$\min_{J(\cdot,\cdot)} \sum_{s\in\bar{\mathcal{S}},h\in\{0,1\}} J(s,h)$$

subject to for all $s \in \bar{\mathcal{S}}$ and $h, k \in \{0,1\}$

$$J(s,h) \geq R(s,k) + |k-h|\log(1-c) + \gamma \sum_{s'\in\bar{\mathcal{S}},h'\in\{0,1\}} P^{(k)}((s,h),(s',h'))J(s',h').$$

The optimal action in state (s,h) is any k maximizing the right-hand side of the above inequality and is found for each in-sample period using the CPLEX commercial LP solver.

8.2 A simple heuristic

One objection to the method of the previous section is that when transaction costs are large the solution obtained above may perform badly in-sample; indeed, it may even lose money when the trivial strategy 'always hold dollars' gets a higher return, namely 0. As an alternative, we consider a *heuristic* defined in terms of three parameters (b,x,ℓ):

$$\textit{buy pounds} \text{ when } \mathbb{E}\left(\log\left(\boldsymbol{F}_{1+\ell}/\boldsymbol{F}_1\right)\middle|\boldsymbol{s}_1 = s\right) \quad > b \tag{9}$$

$$\textit{sell pounds} \text{ when } \mathbb{E}\left(\log\left(\boldsymbol{F}_{1+\ell}/\boldsymbol{F}_1\right)\middle|\boldsymbol{s}_1 = s\right) \quad < x. \tag{10}$$

The expected value in (9) and (10) is just the *expected return* available if we held pounds for the next ℓ days given that the current market state is s. Since we do not have a stochastic process model for FX rate movements, this expectation must also be estimated from the in-sample data (assuming the ergodic theorem holds) as

$$\mathbb{E}\left(\log\left(\boldsymbol{F}_{1+\ell}/\boldsymbol{F}_1\right)\middle|s_1 = s\right) \approx \frac{1}{|Y|}\sum_{t\in Y}\log\left(\frac{F_{t+\ell}}{F_t}\right),$$

where Y is the set $\{t : 1 \leq t \leq T-\ell, s_t = s\}$.

For a several classes of stochastic process models for FX dynamics, the optimal strategy for both very low and very high transaction costs has the form of (9) and (10), making it a plausible heuristic in general.

The optimisation of the three parameters of the heuristic is a non-convex multiextremal problem and for each in-sample period is solved by an appropriate genetic algorithm.

9 Numerical experiments

The results reported below were obtained by applying the approaches described above to the GBP:USD exchange rate data from January 1994 until January 1998 using a moving window of 1 year for training (fitting) followed by 1 month out-of-sample testing.

The cumulative (without reinvestment) returns over the period are shown in odd-numbered Figures 3 through 9 for selected in-sample (fitting) cases and evenly-numbered Figures 4 through 10 for corresponding out-of-sample (testing) cases. The annualised average monthly returns are shown in Table 1 for the various approaches in the out-of-sample case.

Table 1: Out-of-sample average annual returns

Slippage	0 bp	1 bp	4 bp	8 bp	10 bp
RL	93.8	16.3	-1.55	1.64	1.45
GA	94.5	21.6	1.67	1.17	1.71
LP	96.8	15.9	-1.76	0.53	0.432
Heuristic	96.3	8.56	-5.03	-4.89	-5.63

In-sample fitting performance has been shown for completeness to demonstrate the learning ability of the various approaches. It is clear that on the in-sample data set, the simple heuristic approach consistently outperforms all the other methods except in the no slippage case, when all methods were able to fit the data to essentially the same degree. The out-of-sample test results demonstrate, however, that the heuristic approach was in fact significantly *overfitting* the data.

For the out-of-sample back-tests, we note that the genetic algorithm and reinforcement learning approaches tended to outperform the others at lower slippage values. In order to gain further insight into the overall best performing GA, a plot of how often it inferred rules using each indicator was generated for each slippage value and is shown in Figure 11. Figure 12 shows the frequencies that the GA employed specific indicators over the entire 4 year data period, aggregated for all slippage values and into quarters, with considerable variability in the patterns evident. The GA's reduction in trading frequency with decreasing transaction costs is demonstrated dramatically in Figure 13. Similar results apply to the other methods with the exception of the heuristic, whose high trading frequency at realistic transaction costs leads to its poor performance in out-of-sample back-tests. Data on the dealing frequency of all the different approaches is given in Table 2 in order to shed light on the risk profiles of

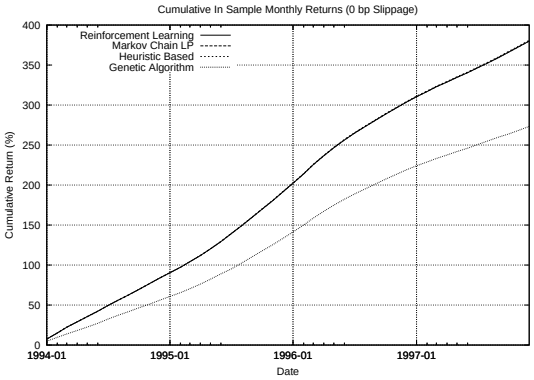

Figure 3: Cumulative in-sample monthly returns at no slippage

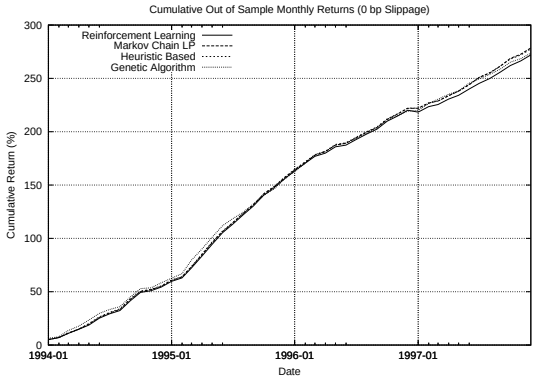

Figure 4: Cumulative out-of-sample monthly returns at no slippage

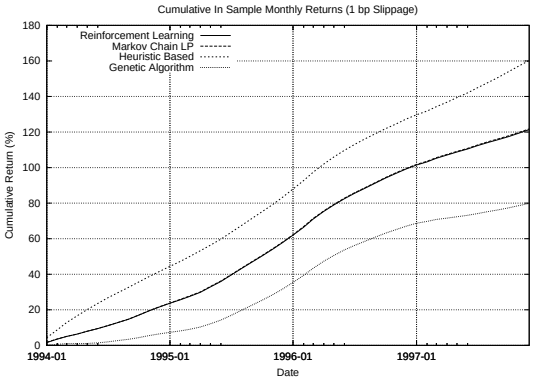

Figure 5: Cumulative in-sample monthly returns at 1 bp slippage

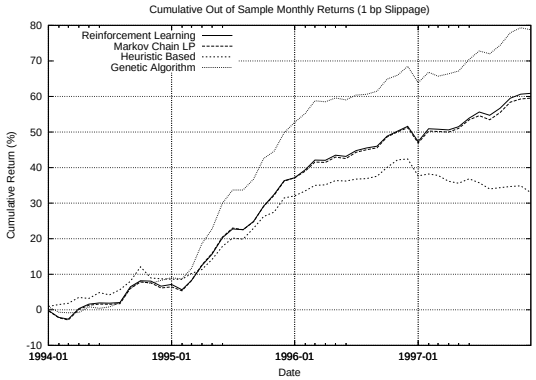

Figure 6: Cumulative out-of-sample monthly returns at 1 bp slippage

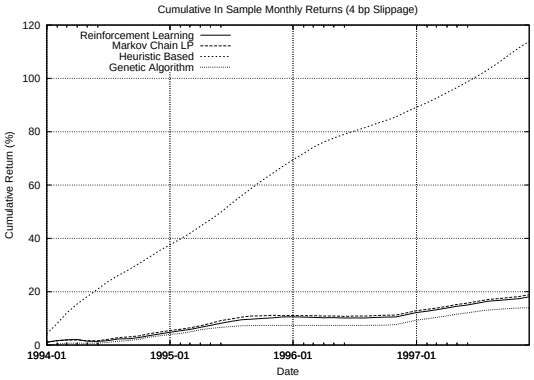

Figure 7: Cumulative in-sample monthly returns at 4 bp slippage

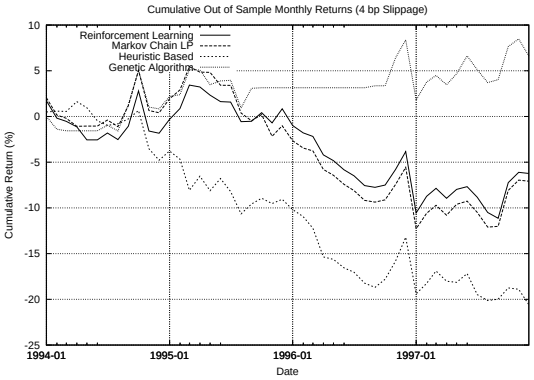

Figure 8: Cumulative out-of-sample monthly returns at 4 bp slippage

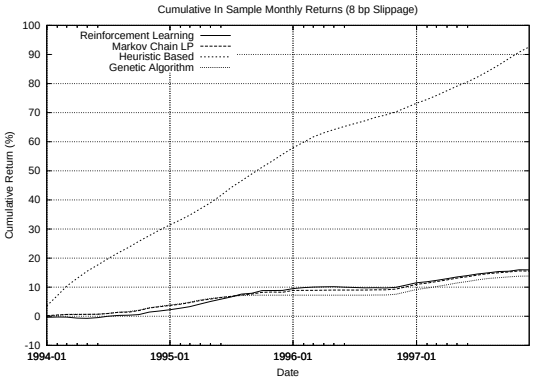

Figure 9: Cumulative in-sample monthly returns at 8 bp slippage

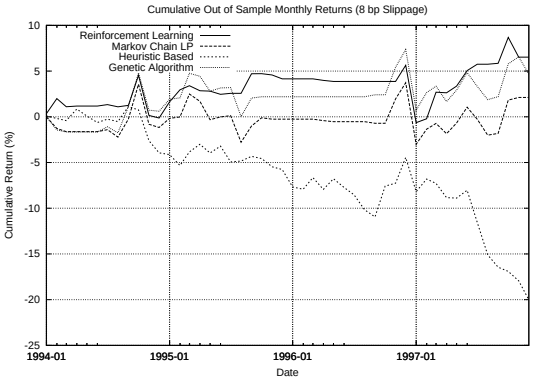

Figure 10: Cumulative out-of-sample monthly returns at 8 bp slippage

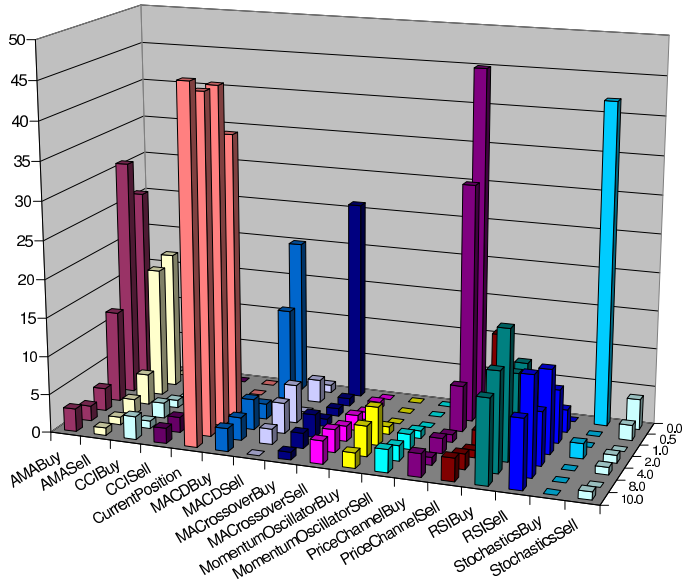

Figure 11: Indicators used by the genetic algorithm by slippage

the different methods. These results are discussed further in the final Section 10.

Table 2: Out-of-sample average monthly dealing frequency

Slippage	0 bp	1 bp	4 bp	8 bp	10 bp
RL	851	220	20.5	0.980	0.604
GA	759	254	0.688	0.688	0.667
LP	852	218	20.5	1.06	0.792
Heuristic	846	123	15.0	7.85	7.44

In order to evaluate the relative risk-adjusted performance of the trading models further, we now define several risk measures found in the financial literature. A risk-adjusted measure commonly used to evaluate portfolio models is the *Sharpe ratio*, defined as

$$\text{Sharpe Ratio} := \frac{\mu_{R_{\text{month}}}}{\sigma_{R_{\text{month}}}}. \tag{11}$$

The Sharpe ratio evaluations of our trading models, as shown in Table 3 demonstrate that on the dataset used we are able to gain significant risk-adjusted returns up to a slippage value of 1 bp. However the Sharpe ratio is numerically unstable for small variances of returns and cannot consider the clustering of profit and loss trades [13, 18]. Furthermore the Sharpe ratio penalizes strategies for upside volatility and its lack of dependence means that a strategy can appear to be successful but in fact suffers from significant drawdown [6]. *Maximum drawdown*

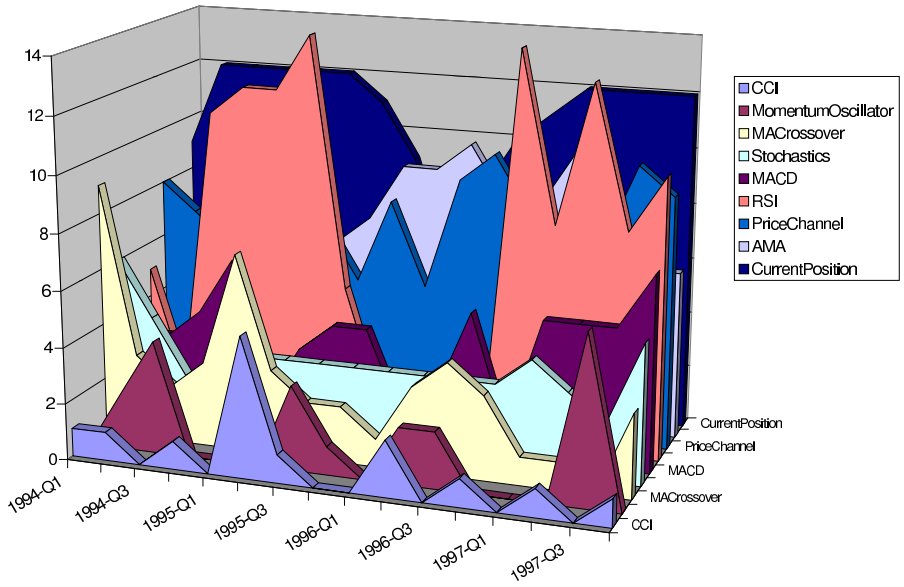

Figure 12: Relative frequencies over time of indicators used by the genetic algorithm

D_T over a certain period of length $T = t_1 - t_0$ is defined as

$$D_T := \max(R_{t_a} - R_{t_b} | t_0 \leq t_a \leq t_b \leq t_1), \tag{12}$$

where R_{t_a} and R_{t_b} are the total returns of the periods from t_0 to t_a and t_b as defined by (7) respectively. In our case T was defined as on a monthly basis and the mean over the out-of-sample back-test period is reported in Table 4. We therefore also quote the *Stirling ratio*, defined as

$$\text{Stirling Ratio} := \frac{R_{\text{month}}}{D_{\text{month}}}, \tag{13}$$

which is the average monthly return divided by the maximum drawdown within that month. This value averaged over the 48 monthly back-test periods is reported in Table 5.

Table 3: Out-of-sample Sharpe ratios

Slippage	0 bp	1 bp	4 bp	8 bp	10 bp
RL	2.10	0.682	-0.00711	0.885	0.0874
GA	2.18	0.732	0.0758	0.0531	0.785
LP	2.23	0.663	-0.0783	0.249	0.0202
Heuristic	2.22	0.397	-0.261	-0.279	-0.290

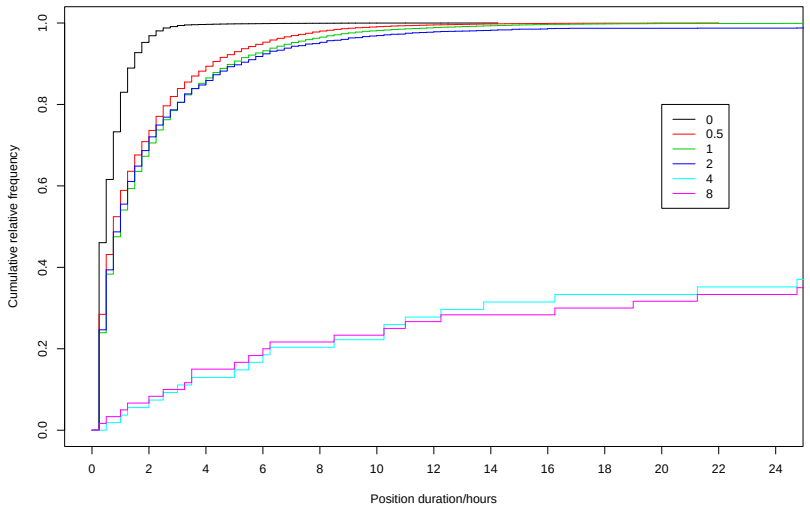

Figure 13: Genetic algorithm position duration by slippage

Table 4: Out-of-sample drawdowns

Slippage	0 bp	1 bp	4 bp	8 bp	10 bp
RL	1.38	1.71	2.25	1.29	0.932
GA	1.45	1.67	1.96	1.94	1.88
LP	1.36	1.72	2.28	1.91	1.89
Heuristic	1.36	1.90	2.03	1.91	2.16

The current RL implementation requires about eight minutes CPU time on a 650MHz Athlon per single training optimisation. The GA is implemented in the interpreted language Scheme, but evaluation is parallelised over multiple similar CPUs. It also takes about eight minutes CPU time per optimisation on a single machine. The Markov chain and heuristic approaches execute in four seconds and approximately four minutes respectively.

10 Discussion and Conclusions

In this paper we have developed three trading strategies based on computational learning techniques and one simple heuristic based on trading thresholds over a fixed horizon. The strategies based on the genetic (programming) algorithm (GA) and reinforcement (Q-) learning train at 15 minute intervals on the buy-sell signals from eight popular technical trading

Table 5: Out-of-sample Stirling ratios

Slippage	0 bp	1 bp	4 bp	8 bp	10 bp
RL	5.44	1.31	0.115	0.236	0.209
GA	5.08	1.72	0.246	0.225	0.239
LP	5.54	1.31	0.121	0.175	0.184
Heuristic	5.45	0.883	0.0375	-0.0152	-0.0457

indicators — some of which require a number of previous observations — and current positions over a one year period of GBP:USD FX data, while the Markov chain strategy use the *entire* set of training data to estimate the relative transition frequencies of the few hundred signal states that occur with a given year. Each of the four trading strategies is then evaluated out-of-sample at 15 minute intervals on the next month of indicator signals and this back-testing process is then rolled forward a month and repeated for a total of 48 months.

It is evident that in-sample, all approaches were able to infer successful trading strategies and also notable that the genetic algorithm consistently underperforms the other methods in-sample. This is undoubtably due to the constraint imposed on the complexity of the rules which was specifically imposed to avoid overfitting. In contrast, the non-GA approaches — in particular the trading threshold heuristic — may end up exploiting noise in the in-sample data set. At zero-slippage however we find that all approaches are able to infer similar strategies and perform similarly out-of-sample. There is evidence that the non-GA approaches do in fact overfit as the GA outperforms the other methods with non-zero transaction costs in the out-of-sample cases up to 8 bp slippage.

The fact that the techniques investigated here return positive results both in-sample and in out-of-sample back-tests implies that there is useful information in technical indicators that can be exploited. This is consistent with the tenets of technical analysis and contradictory to the Efficient Market Hypothesis. Furthermore, the GA's relatively good out-of-sample performance demonstrates that using a combination of technical indicators leads to better performance than using the individual indicators themselves. In fact, Dempster and Jones [15, 14] demonstrate that with a few exceptions these indicators are largely unprofitable on the same data when considered in isolation. Figures 11 and 12 demonstrate that some indicators also convey more information than others depending on the slippage value and the market state. We note that the Relative Strength Index (Buy/Sell) indicators are not used at zero transaction costs but as the slippage is increased, the GA tends to favour them. Indicators

such as Price Channel Breakout, Stochastics and Moving Average Crossover (buy) are very important at zero slippage, but the GA appears to disregard the information provided by them at higher slippage values. At zero slippage the GA is able to infer successful strategies without using the current position. However at higher transaction costs, knowing the current position becomes very important. This lends credence to the argument that this extra position information tends to favour the RL and GA approaches, since the Markov chain approach and the heuristic did not have this information available to them.

The RL approach, the Markov chain and the heuristic all exploit the fact that of the 2^{16} market possible states only a few hundred actually occur in the in-sample period. This number is small enough that each state may be considered individually when deciding a strategy. However, there are two problems with such a rule when it is back-tested out-of-sample.

Firstly, we may encounter a state in the out-of-sample data which was not present in the in-sample data. In that case some arbitrary action must be made and both the RL and the Markov chain method choose to hold their current position. This may be a disadvantage if many new states are encountered and it also ignores the fact that some new states may be very similar to states which were present in the in-sample period. The genetic algorithm on the other hand generates a trading rule whose structure tends to take the same actions in similar states.

Secondly, there is a severe danger that these approaches may learn to well (overfit) the specific in-sample data; in other words the in-sample problem they attempt to solve is *too specific*. Indeed the simple heuristic method demonstrates this quite clearly: it achieves excellent in-sample performance but is mediocre out-of-sample and terrible at realistic transaction costs. The limit on the complexity of the GA is an artificial constraint which reduces the opportunity for the GA to overfit while not prohibiting simple trading rules. This limit effectively forces the GA to work with *generalised* (classes of specific) states.

The natural way forward is therefore to improve the current reinforcement learning approach by forcing state generalization, and also by improving its convergence properties. Another current avenue of research is to find constraints for the in-sample optimization problem which force state generalization (such as the rule-complexity constraint in the GA approach), but for which a heuristic similar to that of Section 8.2 can be applied.

As we have shown that some form of generalization has in fact lead to an improvement in results, a further avenue that can be explored involves the incorporation of generalization

methods into the broad RL approach. *Neuro-dynamic programming* (NDP) [24] attempts to combine neural networks with the central ideas of dynamic programming in order to address this. NDPs employ parametric representations of the value function (such as artificial neural networks) to overcome the curse of dimensionality. The free parameters are tuned using regression or stochastic approximation methods used in combination with classical dynamic programming methods. Further, Wilson's *X-classifier system* [30] attempts to merge ideas from reinforcement learning with those from the Classifier System community in order to incorporate generalization into a Q-Learning-like framework. We also intend to investigate this approach in the present context in the future.

Acknowledgments

The authors would like to thank Dr. Chris Jones for his insightful comments, advice and proofreading of the paper, and James Scott for lending his expertise in Scheme.

References

[1] B LeBaron, "Technical trading rule profitability and foreign exchange intervention," *Journal of International Economics*, vol. 49, pp. 124–143, 1999.

[2] Andrew W Lo and A Craig MacKinlay, "Stock market prices do not follow random walks: Evidence from a simple specification test," *Review of Financial Studies 1*, pp. 41–66, 1988.

[3] Andrew W Lo and A Craig MacKinlay, *A Non-Random Walk Down Wall Street*, Princeton University Press, Princeton, NJ, 1999.

[4] C J Neely, P A Weller, and R Dittmar, "Is technical analysis in the foreign exchange market profitable? a genetic programming approach," *Journal of Financial and Quantitative Analysis*, pp. 405–26, December 1997, Also available as Federal Reserve Bank of St. Louis Working Paper 96-006C.

[5] Mark P Taylor and Helen Allen, "The use of technical analysis in the foreign exchange market," *Journal of International Money and Finance*, vol. 11, pp. 304–314, June 1992.

[6] Jeffrey A Frankel and Andrew K Rose, "A survey of empirical research on nominal exchange rates," Tech. Rep., National Bureau of Economic Research, September 1994, Working Paper No. 4865.

[7] R Levich and Thomas L, "The significance of technical trading rule profits in the foreign exchange market: A bootstrap approach," *Journal of International Money and Finance*, vol. 12, pp. 451–474, October 1993.

[8] C R Osler and Kevin Chang, "Head and shoulders: Not just a flaky pattern," Tech. Rep., Federal Reserve Bank of New York, August 1995, Staff Papers No. 4.

[9] Chris Neely and Paul Weller, "Intraday technical trading in the foreign exchange market," Tech. Rep., Federal Reserve Bank of St Louis, 1999, Working Paper 99-016A.

[10] C Goodhart and M O'Hara, "High frequency data in financial markets: Issues and applications," *Journal of Empirical Finance*, vol. 4, pp. 73–114, 1997.

[11] C Goodhart and R Curcio, "When support/resistance levels are broken, can profits be made? Evidence from the foreign exchange market," Discussion Paper 142, London School of Economics, July 1992.

[12] M A H Dempster and C M Jones, "Can channel pattern trading be successfully automated?," *European Journal of Finance*, 2001, To appear.

[13] M A H Dempster and C M Jones, "Can technical pattern trading be successfully automated? 2. Head and shoulders," Working Paper 12/99, Centre for Financial Research, Judge Institute of Management, University of Cambridge, March 2000.

[14] C M Jones, *Automated technical foreign exchange trading with high frequency data*, Ph.D. thesis, Centre for Financial Research, Judge Institute of Management, University of Cambridge, June 1999.

[15] M A H Dempster and C M Jones, "The profitability of intra-day FX trading using technical indicators," Working Paper, Centre for Financial Research, Judge Institute of Management, University of Cambridge, October 2000.

[16] G W Tan, "Topics in foreign exchange trading systems," M.S. thesis, Centre for Financial Research, Judge Institute of Management, University of Cambridge, June 1999.

[17] M A H Dempster and C M Jones, "A real-time adaptive trading system using genetic programming," Working Paper, Centre for Financial Research, Judge Institute of Management, University of Cambridge, October 2000.

[18] R Gencay, Giuseppe Ballocchi, Michel Dacorogna, Richard Olsen, and Olivier Pictet, "Real-time trading models and the statistical properties of foreign exchange rates," *Internal document RAG.1998-12-01, Olsen & Associates, Seefeldstrasse 233, 8008 Zurich, Switzerland*, 1998.

[19] S Chen and W Lee, "Option pricing with gas: A second report," *IEEE International Conference on Neural Networks*, vol. 1, pp. 21–25, 1997.

[20] N. K. Chidambaran, C. H. Jevons Lee, and Joaquin R. Trigueros, "An adaptive evolutionary approach to option pricing via genetic programming," in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, Eds., University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998, pp. 38–41, Morgan Kaufmann.

[21] J H Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbour, MI, 1975.

[22] J R Koza, *Genetic programming: On the programming of computers by means of natural selection*, MIT Press, Cambridge, MA, 1992.

[23] Olivier V. Pictet, Michel M. Dacorogna, Bastien Chopard, Mouloud Oudsaidene, Roberto Schirru, and Marco Tomassini, "Using genetic algorithm for robust optimization in financial applications," *Neural Network World*, vol. 5, no. 4, pp. 573–587, 1995.

[24] D P Bertsekas and J N Tsitsiklis, *Neuro-dynamic programming*, Athena Scientific, 1996.

[25] C Watkins, *Learning from delayed reward*, Ph.D. thesis, Department of Engineering, University of Cambridge, 1989.

[26] R S Sutton and A G Barto, *Reinforcement learning: An introduction*, The MIT Press, 1998.

[27] J Moody, L Wu, Y Liao, and M Saffell, "Performance functions and reinforcement learning for trading

systems and portfolios," *The Journal of Forecasting*, vol. 17, pp. 441–470, 1998.

[28] P Maes and R Brooks, "Learning to coordinate behaviors," *Proceedings of the 8th National Conference on Artificial Intelligence*, pp. 796–802, July 29 – August 3 1990.

[29] M Mataric, "Learning in multi-robot systems," *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, pp. 32–37, 1995.

[30] S Wilson, "Classifier fitness based on accuracy," *Journal of Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, 1995.