A real-time adaptive trading system using genetic programming

M A H Dempster and C M Jones[1]

Centre for Financial Research[2], Judge Institute of Management, University of Cambridge, Trumpington Street, Cambridge, CB2 1AG, UK

E-mail: mahd@jims.cam.ac.uk and cmj24@cam.ac.uk

Abstract
Technical analysis indicators are widely used by traders in financial and commodity markets to predict future price levels and enhance trading profitability. We have previously shown a number of popular indicator-based trading rules to be loss-making when applied individually in a systematic manner. However, technical traders typically use combinations of a broad range of technical indicators. Moreover, successful traders tend to adapt to market conditions by 'dropping' trading rules as soon as they become loss-making or when more profitable rules are found. In this paper we try to emulate such traders by developing a trading system consisting of rules based on combinations of different indicators at different frequencies and lags. An initial portfolio of such rules is selected by a genetic algorithm applied to a number of indicators calculated on a set of US Dollar/British Pound spot foreign exchange tick data from 1994 to 1997 aggregated to various intraday frequencies. The genetic algorithm is subsequently used at regular intervals on out-of-sample data to provide new rules and a feedback system is utilized to rebalance the rule portfolio, thus creating two levels of adaptivity. Despite the individual indicators being generally loss-making over the data period, the best rule found by the developed system is found to be modestly, but significantly, profitable in the presence of realistic transaction costs.

1. Introduction

Some financial and commodity market traders study market price history with a view to predicting future price changes in order to enhance trading profitability. This study is known as *technical analysis*. In their surveys, Taylor and Allen (1990, 1992) found that over 90% of surveyed London foreign exchange (FX) dealers and traders use technical analysis of some sort. Furthermore, a 1995 survey of FX dealers in Hong Kong by Lui and Mole (1998) found technical analysis to be significantly more popular than fundamental analysis at shorter time horizons. Many traders aim to practice technical analysis

as systematically as possible without automation while others use technical analysis as the basis for constructing systems that automatically recommend trade positions. In this paper we construct such a system in an attempt to analyse the actions of the wide body of traders who practice technical analysis. An introduction to the practice of technical analysis is given by Neely (1997) from an academic viewpoint and a classic practitioner text is Colby and Meyers (1988).

Despite the wide coverage of trading systems in the existing practitioner literature, the majority of academic work in the area of systematic trading concerns itself with individual trading rules. When trading systems are discussed in either the practitioner (for example, Pardo (1992), Schwager (1996)) or academic (for example, Gencay *et al* (1998)) literature, little attention is paid to *system construction* as opposed to *rule construction*. An exception is Pictet *et al* (1992) who

---

presented a system-based approach, but here the emphasis is on a broad-based system that encompasses data-capture and storage through to bookkeeping, as opposed to a trading system *per se*. Furthermore, there has been little attention paid to systematic adaptation within trading systems (with the exception of Moody *et al* (1997)) despite the degree to which systematic traders constantly change their rules. In this paper we focus on system construction and consider adaptation. Also, for realism, we use intraday data in this work. The consideration of such data is particularly useful when analysing returns of technical trading rules; many traders apply rules at intraday frequencies[3] and of those traders who only analyse daily data for entry signals, many will use intraday data for trade exit, especially when stops[4] are placed in the market. Thus, the use of intraday data adds realism when analysing trading rules.

Other work in this area includes Levitt (in Refenes (1995)) who has proposed using the method of *genetic-based global learning* in a FX trading system. Here, genetic algorithms are used to attempt to find the best combination indicators (out of the moving average crossover and a more complex moving average-based indictor—the ADX) for prediction and trading. Results are shown to be profitable but are reported in too little detail for objective scrutiny.

Dunis *et al* (in Dunis and Zhou (1998)) have developed a FX trading system that uses genetic algorithms to optimize parameters (in the style of Colin in Deboeck (1994)) for a simple technical trading indicator—the RSI. This work has considerable merit since intraday data are used. However, the study considers a period of only 129 days and so results are far from conclusive. It is noted that the ultimate aim of such a project would be to create a system based on an ensemble of indicators—a task we attempt below.

In this paper, we introduce a framework for systematic trading system construction and adaptation, based on *genetic algorithms* (GAs)—in fact, *genetic programmes* (GPs). Previously, Neely *et al* (1997) and Allen and Karjalainen (1999) have used genetic programming to discover profitable trading rules, this work is further discussed in section 4. Here, our aims are twofold: to develop a system that trades profitably and to emulate the behaviour of a technical trader who adapts to the market. Previously, academics have asked the question 'can technical trading rules consistently make a profit?' being driven by the implications for market efficiency. Here we ask the questions: 'can a technical *trader* consistently make a profit?' and 'should a technical trader or trading system adapt to market conditions or is it better to use a static system?' The work that follows provides at least some provisional answers, which are further investigated in Dempster and Jones (2000) and Dempster *et al* (2000).

The paper is laid out as follows. Sections 2 and 3 respectively introduce briefly the concepts of GAs and GPs

and describe the GBP/USD data used. The trading system developed is described in section 4, while the details of the genetic programme utilized are given in section 5. Section 6 contains our trading results for both static and dynamic trading strategies. Conclusions are drawn and current research described in section 7 and an appendix contains precise specifications of the six technical indicators used in this study.

2. Genetic algorithms and genetic programming

Genetic algorithms, initially developed by Holland (1992, revised reprint of his 1975 PhD thesis, see also Holland (1976, 1980, 1986)), are iterative systems that aim to find near-optimal solutions to multi-extremal problems by imitating the process of evolution. GAs are search algorithms that differ from more standard search algorithms in that the search is conducted using a population of structures rather than a direction or single structure. GAs consider many structures as potential candidate solutions and work with a high level of global sampling of the search space which increases the probability of convergence to a global optimum. Although convergence to the global optimum is not guaranteed, GAs are quite robust in producing near-optimal solutions to a wide range of problems including problems with high levels of uncertainty and problems which are not easily reduced to a precise mathematical formulation. The search is carried out in a 'survival of the fittest' fashion by evolving a set of potential solutions until the most superior ones come to dominate. GAs are highly efficient in searching large spaces for attractive solutions to complex problems and so would seem well suited to trading rule selection using a large search space of rules tested over a considerable number of time series data.

The starting point in using GAs to solve a problem is to represent the problem in a way that a GA can work with. This often amounts to representing the solution space as a finite number of strings of binary digits. *Binary strings* are an effective form of representation since complex statements of Boolean logic as well as numerical values of parameters can be represented in this form, e.g. the statement

IF (Function 1 evaluated at Frequency 1) = TRUE
AND (Function 2 evaluated at Frequency 2) = TRUE
THEN *Buy US T-Bond Futures*

can be easily expressed as a binary string. The resulting search space is finite when parameters take only discrete values to yield a binary representation as a string of fixed length. Secondly, there needs to be a means of evaluating the *fitness* of the constituents of the solution space, i.e. the suitability of each potential solution, for how well they perform. For example, in the case of selecting trading rules the fitness could be viewed as the profitability of the rule tested over a time series of historical price data. Finally, an initial *population* of suitable solutions needs to be generated. The population is the subset of the solution space on which the GA operates at any one time step and in each *trial* it operates first on the *initial population*. The initial population is usually generated by randomly sampling a number of potential solutions from

---

[3] As a whole, the operations of intraday FX traders account for 75% of FX market volume (Bank of International Settlements 1998).

[4] A *stop* is a predetermined price level at which the trader will exit the trade. A stop that is placed in such a way that a trade is exited after the price moves down a set amount from the maximum price attained during the trade (and similarly *mutatis mutandis* for short trades) is known as a *trailing stop* (see Schwager (1996) and James and Thomas (1998)).

the search space. There are no strict rules for determining the population size, although populations of 100–200 are common in GA research. Users often experiment with population size, however, since larger populations ensure greater diversity at the cost of requiring greater computational resources. Once the problem has been developed as described above, GAs can be used to search for attractive solutions. Each stage of the algorithm is discussed in detail in section 4 and further information on genetic algorithms can be found, for example, in Haupt and Haupt (1998).

As we have implied above, a genetic algorithm only considers solutions with the same, fixed string length. An extension by Koza (1992), called genetic programming, allows string length to vary within the solution space. With GPs, strings (and hence solutions) can be seen as non-recombining *decision trees* with the non-terminal nodes as functions (including Boolean operators) and the root as the function output. We use GPs rather than GAs in our system, in order to harness their flexibility but restrict the form of the binary strings—equivalently decision trees—to prevent over-fitting (see section 5).

3. The data

This analysis was carried out on spot FX tick[5] data for the British Pound/US Dollar exchange rate (BPUS, or 'spot cable' as it is sometimes called) ranging from 6.89 to 12.97 inclusive.

These data were supplied by CQG Data Factory and FutureSource, two well-known data providers. The CQG data—ranging from 6.89 to 3.96 inclusive—were gathered from a number of FX brokers, whereas the FutureSource data, stored from a live satellite feed via the Omega TradeStation utility, are the amalgamated product of major bank FX quotes and make up the remaining part of the data set. The fact that the data set consists of actual quotes from two different source providers is not ideal, but such problems are typical with the analysis of high-frequency data based on non-exchange traded instruments, since the majority of live tick data providers do not retain historical data[6].

The convention for quoting BPUS is to quote a five-digit figure that represents the value of one British Pound in US Dollars (most other currencies are quoted in a style opposite to this) with an implicit decimal point after the first digit; e.g. a BPUS rate quoted 15104 means £1 = $1.5104.

The CQG data consist of *bid* and *ask* prices—the price that the quoter would buy and sell British pounds for, respectively, if approached in the market. The difference between the bid and the ask (bid–ask) is called the *spread*. The convention when dealing with such data is to convert them to *midpoint* data: $\frac{1}{2}$(bid + ask) or, by definition, (bid + $\frac{1}{2}$ spread) or (ask − $\frac{1}{2}$ spread). In the event that bid and ask quotes are uncoupled (which sometimes occurs), the bid or ask is converted to the

midpoint by respectively adding or subtracting one half of the spread calculated from the last bid/ask pair.

The above data tend to be well checked for errors by the vendor. All the same, the data have been screened for structural breakdown and irregular quotation by sweeping them with simple software that checks for conformity to the conventional, fixed-width, comma-separated ASCII format, for well-ordered temporal structure and for irregularly high or low ticks (which are more than 500 pips[7] from the last quote). This last screening has been backed up by graphical inspection of the data.

The data have then been aggregated to various frequencies in the standard *open-high–low-close* (OHLC) format. Consider the set of time-stamped *tick data* $\{(q_i, t_i)|0 < i \leqslant K; i, K \in \mathbb{Z}^+\}$ where K is *the number of ticks* in the set, q_i is the *price level* of the ith midpoint quote and t_i is the *time* at which the ith tick occurred (converted to be measured in minutes elapsed since the start time—22:00 hours—and date and so, on each day, $t_1 = 0$). The ticks are ordered temporally but more than one tick may occur within the same minute and so we have a weak inequality $t_i \leqslant t_{i+1}$. When such multiple ticks occur they are listed in order of occurrence.

This data set is converted to sets of data aggregated to various frequencies τ, denoted as τ minute frequencies; e.g. if $\tau = 1$, then frequency is *minutely* and is denoted 1 minute (but 1440 minute is called *daily*).

The aggregation to OHLC τ minute frequencies results in the following data set:

$$\{(o_j, h_j, l_j, c_j, b_j)|0 < j \leqslant L; j, L \in \mathbb{Z}^+\},$$

where o, h, l, c denote, respectively, *open*, *high*, *low* and *close* quotes for *bar b* which for *bar number* $j > 0$ is determined by

$$b_j = (b_{j-1} + n\tau)$$
$$n := \inf\{s|\exists i \in [1, K] \text{ s.t. } t_i \in [b_{j-1}, b_{j-1} + s\tau), s \in \mathbb{Z}^+\}$$
$$b_0 := 0$$

and

$$o_j = q_{io} \qquad \text{where } io := \inf\{m|t_m \in [b_j - \tau, b_j)\}$$
$$c_j = q_{ic} \qquad \text{where } ic := \sup\{m|t_m \in [b_j - \tau, b_j)\}$$

$$h_j = \max\{q_{io}, q_{io+1}, \dots, q_{ic}\}$$
$$l_j = \min\{q_{io}, q_{io+1}, \dots, q_{ic}\}.$$

By convention, b_j is converted from *minutes elapsed* to *time and date* format when quoted. These somewhat esoteric definitions are required since the data are sometimes sparse out of peak trading times.

4. Trading system

As previously discussed, the aims of constructing a software entity that systematically trades financial markets are two-fold. Primarily, we have tried to develop a system that closely

---

[5] Here a new data point, or *tick*, is recorded with every change in price. As a result there are often several ticks per minute.
[6] It should be noted that the data consist of real quotes as opposed to the 'representative quotes' sometimes supplied by real-time data providers such as Reuters and collected by Olsen and Associates.

[7] A *pip* is the minimum allowable change in price—in this case $0.0001— also referred to as a basis point (bp) of a notional dollar.
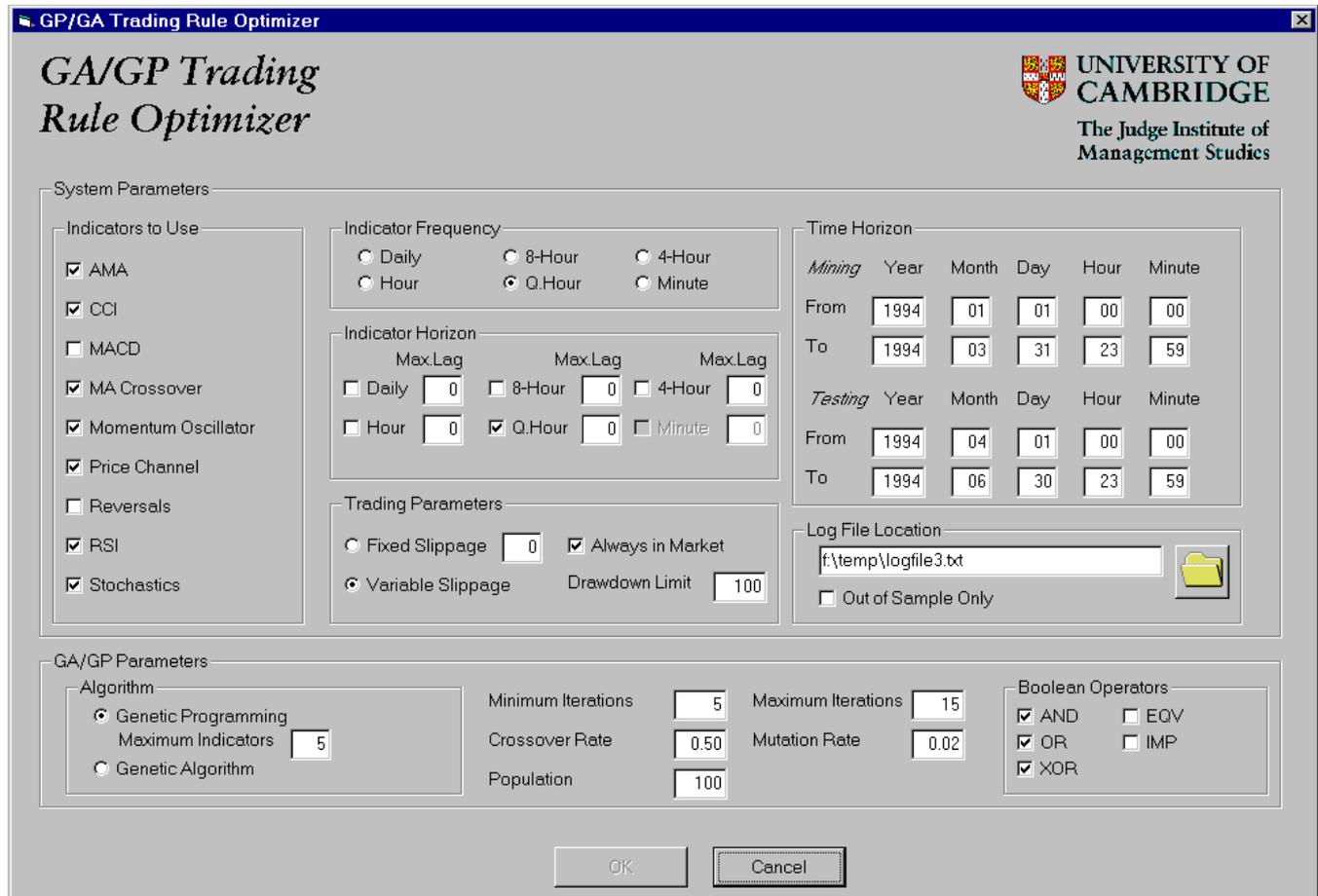
Figure 1. System user interface.

resembles a technical trader who rationally[8] chooses his or her trading strategies from an arsenal of popular technical trading rules.

Another aim of this work is to analyse the impact of changes in market conditions. Typically, after prolonged trading loss most systematic traders will change their trading rules (otherwise, they would lose the sponsorship of their backers or institution). By developing a trading system that has the potential to adapt to shifts in market conditions, we hope to evaluate the performance of such strategies relative to the more usually analysed static 'select and hold' strategy for trading rule utilization.

System architecture

The system consists of a genetic-programming-based rule selection engine which chooses trading strategies that are combinations of popular technical analysis indicators and rules. Choice is on the basis of selection criteria in terms of trading profit and downside risk over a user-defined period of time. The system has the capacity to use the chosen strategies to subsequently perform *out-of-sample testing* on unseen data or to *live-trade* in real-time using a live data feed. The trading recommendations of the selected strategies are overlaid by

a cash management filter that exits trades when trading loss exceeds a user-defined threshold.

Outcomes (profits, drawdowns, trades, etc) are reported by means of a 'trader's log' which summarizes individual rule performance over the chosen selection and out-of-sample periods. Below, we discuss each of these components in detail. Figure 1 displays the user interface and a diagram of the system architecture is given in figure 2.

Trading strategies

As is consistent with proprietary FX trading in a financial institution, we assume a fixed-credit line in dollars with notional principal one million to buy (long) pounds and the equivalent amount in pounds to short dollars. Thus the system buys or sells either currency in trades of a *fixed size* of $1 m equivalent. Trading *profit* or *loss* over a specified period (one quarter) is the cumulated value (in dollars) of net gains or losses on these trades when they are exited after allowing for (round trip) *transaction costs* (slippage).

As with previous work (see Dempster and Jones (1999a, b, 2000)), we use the following model for slippage. Slippage is a concept that is often alluded to but rarely specified. In this work, we take *slippage* to be the penalty incurred when a trade is placed as a result of the difference between actual price at transaction execution time and previously quoted mid-price.

---

[8] That is, systematically, based on prespecified evaluation criteria.
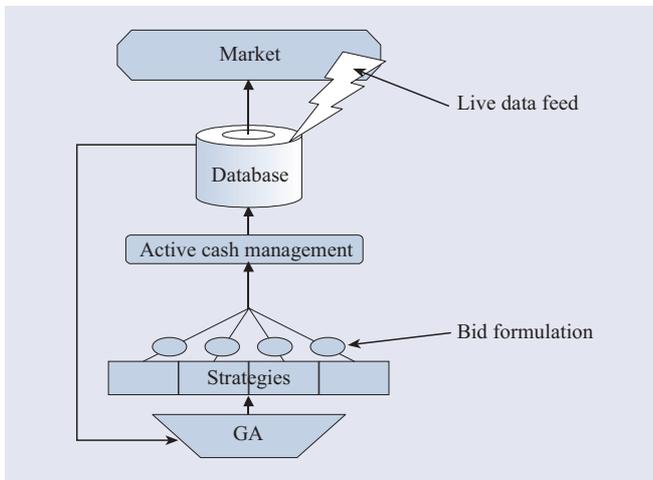
Figure 2. System architecture.

As a result, we have one penalty that covers both potential sources of slippage: *time delay* and *transaction cost*.

A flat 10 pips per round turn can be allowed for transaction costs and to compensate for discrepancies between data and actual prices. In addition, the following slippage per trade (not round turn) is deducted:

- trade time between 08:01 and 17:00 (London market); 2.5 pips
- trade time between 17:01 and 22:00 (New York market); 4 pips
- trade time between 22:01 and 08:00 (Asian market); 5 pips.

These values were assigned after extensive discussions with proprietary FX traders in several financial institutions.

Trading strategies are constructed by allowing the genetic selection engine to combine the individual technical indicator-based trading rules (see the appendix) with an array of Boolean operators to form a system rule. Rules are split into BUY and SELL rules (suffixed 'b' and 's' respectively) and each strategy consists of both a buy part and a sell part. However, buy rules are not forced to be linked to their corresponding sell rule and vice versa. When a BUY or SELL rule gives a buy or sell signal, the rule is evaluated as TRUE and is otherwise FALSE. For example, a rule may be of the form:

> IF RULE #1b IS *TRUE* AND (RULE #2b IS *FALSE* OR RULE #6b IS *TRUE*) THEN **LONG POSITION** | IF RULE #3s IS *TRUE* THEN **SHORT POSITION**.

Where RULE #1b IS *TRUE* could be, for example, *short-term moving average crosses above long-term moving average*. RULE #1b would, therefore be FALSE if, *short-term moving average does not cross above long-term moving average*. Similarly, RULE #1s would be TRUE if *short-term moving average crosses below long-term moving average*, and so on.

In each case, the GP/GA has the potential to choose from buy and sell rules based on six technical indicators defined in the appendix—the AMA, CCI, MA crossover, PCI, RSI and Stochastic—along with a simple measure of 'change in price'. Further details of these indicators can be found in Dempster and

Jones (2000) and the appendix. The Boolean operators AND, OR and XOR are used to compose these buy and sell rules. Note, however, that no (composite) system rule or trading *strategy* will be 'always in the market' as a result of imposing a cash management filter (discussed below). Finally, we allow system rules to use technical indicators evaluated at a range of frequencies and lags to result in strategies formed from rules based on a variety of indicators and frequencies. However, it is clearly necessary to ensure that the system should be able to respond quickly to the market, and thus not use rules which are infeasible in terms of real-time execution. Therefore, we force rules (and hence trade *entry*) to use 15 minutes (rather than 1 minute) as the highest frequency for rule evaluation. (Guillaume *et al* (1997) have found quotes to match traded prices at this frequency.) However, the system responds to the market at the 1 minute level for forced *exit* by the cash management filter, since a live trader would also be bound by such a rule.

By using a GP, rather than a GA, we allow the flexibility of *variable* string length to obtain optimized strategies based on single rules (and hence indicators) as well as strategies that utilize up to a fixed number of chosen rules (and hence indicators); the solution set of the GA consists *only* of strategies that utilize a fixed number of chosen indicators and is therefore a subspace of the GP solution space.

As discussed in section 1, there is previous work on rule discovery using GPs (Allen and Karjalainen 1997, Neely *et al* 1997). This work differs from that presented here in several ways. It aims to discover best rules in the context of investment (or one-way trading) in the foreign currency, whereas we aim to construct a trading system in which the emphasis is not on the performance of existing rules but on the *overall* performance (see also Neely and Weller (1999)). Furthermore, we use intraday data, *two-way* trading with equivalent credit lines in both traded currencies and cash management overlays in order to represent the actions of a 'live' trader. Finally, we allow the GP to construct strategies from a selection of indicators, whereas the bulk of previous work starts with a collection of simple functions and uses GA/GP technology to build complex rules (but see Neely and Weller (2000)). We have chosen the two-level two-way trading approach for a number of reasons. We aim to emulate a technical trader who him(her)self would build strategies from technical indicator-based rules but would not use strategies based on very complex indicators (since 'functions of functions' would not usually be allowed in this framework). Letting strategies evolve from simple functions results in very complex rules that are unlikely to be picked by a trader, lack transparency and, as a result, cannot be easily comprehended. Despite the trading approaches of Neely *et al* (1997) and Allen and Karjalainen (1999) not suiting our purpose, their methods are of course valid since their aims were different. They aimed to discover the most profitable one-way trading rules that could be built from combinations of simple functions, whereas we aim to discover whether or not *existing* indicator-based trading rules can be *combined* with Boolean operators (see section 4) to form profitable two-way trading strategies.

Selection criteria

There are many different ways of gauging the performance of trading strategies. In classical finance, the Sharpe ratio—essentially excess return divided by risk as measured by the standard deviation of return—is a common measure. However, in the world of trading, traders are not perturbed by the possibility of a volatile return structure provided that the strategy eventually makes a profit. This would imply that to measure 'risk' as the volatility of returns when simulating a technical trader is incorrect, even if only the volatility of negative returns, the 'semideviation', is considered as in the *Sortino* ratio. What does matter to traders is the maximum *drawdown*[9], since this represents the maximum losing streak that has been experienced and it is such losing streaks that potentially result in the reduction or withdrawal of trading capital from sponsors, be they investors or employers. As a result, the *Stirling ratio*—profit divided by maximum drawdown—is often used by traders and investors in alternative investment strategies (such as systematic trading or hedge funds) to assess performance. This ratio is frequently used in investment management and trading but is referenced little and so there is no conclusive definition (one definition can be found in the PerTrac User Manual at www.pertrac.com). Note that excess (over risk free rate of return) profits are not used here since—as we consider this system from the viewpoint of a bank trader—a credit line rather than actual funds is traded. From this point of view, all profits that we report are 'excess'. As a result, a trader will only have concern for his cumulated trading profits, as opposed to interest income, and that is what we consider here.

Given the undesirable behaviour (shared by any ratio index such as the Sharpe ratio) of the Stirling ratio when the denominator is small (namely large swings in the ratio for small swings in maximum drawdown), we make a modification to the divisor. Moreover, a trader will be indifferent between strategies that have a drawdown of less than a particular amount and the ratio should reflect this indifference. As a result of these factors, we use a *modified Stirling ratio* given by:

$$S := \frac{\text{Return}}{1 + \text{Modified drawdown}},$$

where *modified drawdown* is max(drawdown, 2% of current position) and return and drawdown are measured as a percentage of (notional) traded assets. Again this 2% drawdown tolerance level was arrived at by discussions with FX traders. Finally, since any form of risk-adjusted return is meaningless when returns are negative, we do not use the Stirling ratio in such instances, but use return alone instead. Thus S provides a continuous and sensible total ordering for all combinations of drawdown and return over a specified period.

Cash management

In contrast with other recent work (see Neely and Weller (1999)) and as noted above, we allow our system an equivalent

[9] Largest loss of trading capital from a current position throughout a given period in absolute return terms (%).
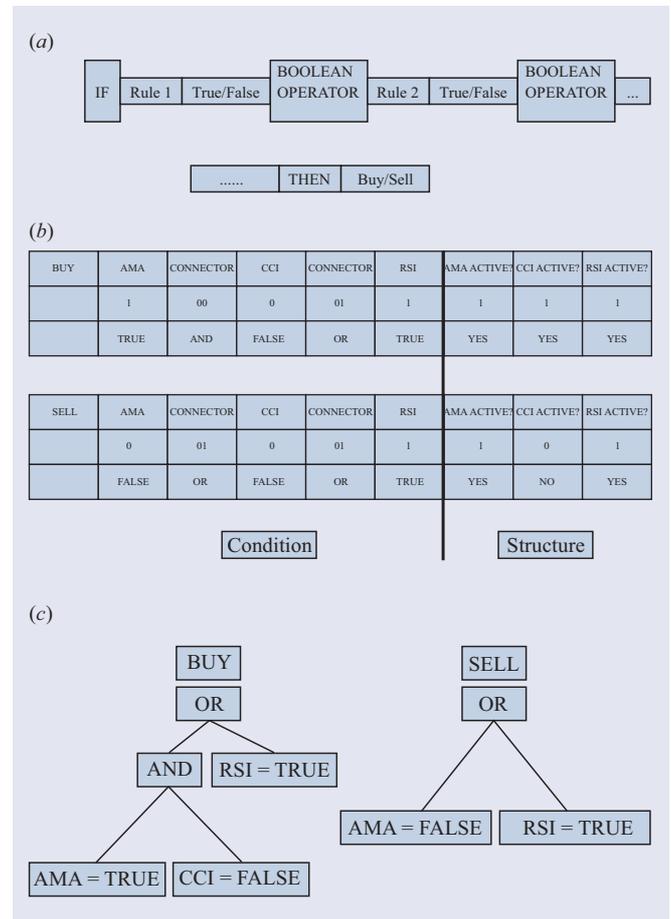
Figure 3. (*a*) Strategy structure, (*b*) binary string representation of rules and (*c*) tree representation of rules.

credit line in *both* currencies, USD and GBP, in order to represent the usual situation of a real FX trader with position limits. Thus, for return evaluation purposes, our system makes unit trades whose return is calculated (in dollars) when the trade is exited. These returns are neither compounded nor earn any form of (overnight deposit) interest as noted above. Returns on fixed-size (unit) trades are simply cumulated over the trading period. Thus, our system conforms with the usual practice of an FX trading desk which here is trading continuously over an extended period of time and would be appropriate to a computer-based trading system usable in actual trading.

The system currently implements cash management with a trailing stop, i.e. exit after price has dropped from the current price of a user-defined number of pips, here 100 pips or 1c for a USD trade.

5. Genetic programme

As discussed above, we use a genetic programme to select strategies of the form displayed in figure 3(*a*).

As can be seen from the figure, system rules or strategies are of the form

RULE|CONDITION|CONNECTOR|
RULE|CONDITION|CONNECTOR|.......|ACTION,

where individual rules are based on technical indicators, conditions are TRUE or FALSE, connectors are Boolean operators (in this case AND, OR and XOR) and actions are BUY or SELL. Thus system rules can be divided into rules for each currency.

To ease discussion and explanation, we will refer to each RULE|CONDITION|CONNECTOR group as a *unit* and, within a unit, to either RULE, CONDITION or CONNECTOR as a *constituent*.

In figure 3(*b*), we show how such a rule structure translates to a binary string in a simple case considering only three indicators: AMA, CCI and RSI. Furthermore, we only consider one frequency, say 15 minutes, and no lags. Each indicator has a particular *bit* that reflects that it is either true or false by indicating 1 or 0, respectively. Furthermore, each indicator (except the last) has a connector that is a Boolean operator (as noted above). For the sake of simplicity we will consider a choice of just three connectors: AND, OR and XOR, represented in binary by 00, 01 and 10, respectively, say. Note that if more indicators were to be considered, then the string would grow at 3 bits per indicator on both the BUY and SELL parts of the string since each new indicator would need a dedicated bit at a fixed position in the BUY and SELL part of the string to represent TRUE or FALSE and an accompanying connector. However, if more connectors were to be considered, then the string would increase non-uniformly since each connector is represented by a number translated to binary, e.g. 1, 10, 11, 100, 101, 111, etc. Three connectors are used here so that a two-bit representation for each connector is all that is needed. Finally, the last part of the string indicates which indicators are to be ignored (consider the string to consist of a condition part and a structure part). This allows strategies containing variable numbers of indicators (GP instead of GA). A '1' in the appropriate position (fixed for each indicator) shows that the indicator is included in the rule, whereas a '0' shows that the indicator is to be ignored.

From figure 3(*b*) it can be seen that the *buy* rule is 'BUY if AMA TRUE AND CCI FALSE OR RSI TRUE' since, the latter part of the string that dictates structure tells us that all indicators are active. Writing the bits as a string gives (1000011111). Note from the *sell* rule that the CCI (and its neighbouring connector) is not included and so the rule is 'SELL if AMA FALSE OR RSI TRUE' represented as (0010011101). Note that although the BUY and SELL rules are dependent on three and two indicators, respectively, they are of the same length.

In figure 3(*c*) we present the strategy in tree format with the connectors as nodes and right branches of depth one only.

At each trial a GP/GA performs the following stages in each iteration until convergence is achieved or the maximum-allowed number of iterations have occurred:

1. initialize population
2. calculate fitness
3. crossover
4. mutate.

We discuss each stage below.

Population initialization

We initialize the population using ancillary uniform pseudo-random integers which are translated to the strings that constitute the initial population using a methodology to be described below. We allow strings to be of various lengths but, for ease of representation, achieve this by generating strings of a fixed length and 'ignoring' parts of the generated string, as described above. The latter part of the string contains information about the structure of solution, namely which parts of the string are to be included or 'ignored'. This can be thought of as generating a string containing all available indicators and their associated units and then 'switching some on'. The following explains this procedure in more detail.

1. Assume there are a maximum of k included indicators and a minimum of j. Further assume that we may construct rules from a total of I indicators, numbered 1 to I. Generate a uniform pseudo-random *integer* variable $U(j, k)$, where $j \leqslant U(j, k) \leqslant k$. This represents the number of indicators in the forthcoming rule/string that are to be included ('not ignored'). Thus, the rule that is to be generated will be based on U indicators.

2. Generate U uniform pseudo-random integer variables V where $1 \leqslant V(1, I) \leqslant I$ without replacement: $V_1(1, I), \ldots, V_U(1, I)$, where the resulting pseudo-random integers are subscripted in ascending order. These numbers represent the indicators (which are themselves numbered 1 to I) which will be included in this rule/string (as described previously). Thus, '1s' are written at the corresponding indicator bit in the *structure*-related part of the string (see figure 3(*b*)). All other bits in this part of the string are then written as '0', since, if an indicator is not selected to be included, it must not be. In fact, we consider the entire string to be a fixed-length array of zeros initially until individual bits are overwritten.

3. Next, the condition part of the string is populated. Generate U uniform pseudo-random binary variables $B(0, 1) = 0$ or $1 : B_1(0, 1), \ldots, B_U(0, 1)$. Each B_j is written in the corresponding indicator bit (this position is represented by number V_j) in the *condition* part of the rule/string (see figure 3(*b*)).

4. Finally, the connectors need to be represented. Assume that there are C connectors allowed. Generate $U - 1$ uniform pseudo-random integer variables (since the last indicator is not followed by a connector) $D(1, C)$, where $1 \leqslant D(1, C) \leqslant C : D_1(1, C), \ldots, D_{U-1}(1, C)$. Each D_j is converted to fixed-width binary ('01' instead of just '1' if C is two bits long in binary) and written next to the corresponding indicator bit (this position is represented by number V_j) in the *condition* part of the rule/string (see figure 3(*b*)).

This process is repeated until the number of generated strings is equal to the (user-defined) initial population size.

Fitness calculation

At each trial over a user-defined period, each of these strategies is tested by simulating their trading performance over historical

data. The fitness performance evaluation of every string is calculated at each iteration using the criteria discussed in section 4. This is the lengthiest part of the algorithm due to the large number of (intraday) data that are used. Even when strategies contain indicators that are evaluated only at lower frequencies, data have to be processed minute by minute since this is the frequency at which the cash management exit rule is evaluated. A trade is entered (long or short) when a (BUY or SELL) rule gives a signal. Trades are exited either when the cash management rule is activated (trailing stop of 100 pips) or the trade is reversed when a SELL (BUY) rule is activated whilst a long (short) position relative to the purchased currency is held. Strategies are 'traded' over the user-selected (mining) period and slippage-adjusted profits are calculated on the basis of a notional US $1 m of trading capital and its GBP equivalent, also treated in USD for simplicity. Furthermore, the largest drawdown (as defined in section 4) is calculated. Drawdown and trading profit in terms of returns on a unit trade are then used to calculate the modified Stirling ratio as defined in section 4. This process is repeated for each system rule or strategy and all are ranked with respect to modified Stirling ratio (fitness) over the training period in descending order.

Crossover

Crossover is the process of cutting strategy string pairs at appropriate points and exchanging tails between heads to make a new pair.

Only the best (user-selected) $s\%$ (with respect to fitness) of the strategy population is considered for crossover and strategies are selected randomly for crossover with a probability biased by fitness rank, so that the higher the ranking, the greater the chance for selection. The probability p_i of selection for a string ranked i is calculated by the formula

$$p_i := \begin{cases} \frac{s.\text{PS}-i}{\sum_{j=1}^{s.\text{PS}} i} & i \geqslant s.\text{PS} \\ 0 & \text{otherwise,} \end{cases}$$

where s is the percentage of the population size to be considered for crossover (judiciously chosen so that s.PS is an integer), PS is the population size and i is the ranking of the string in terms of fitness (in descending order).

Based on this probability, pairs of strings are independently selected (with replacement) from the population and crossed over. Once a pair has been selected, a cut point for both strings is uniformly pseudo-randomly selected, so that both strings are cut into two usually unequal pieces with the 'head' and 'tail' parts of both strings being the same length. For example, given two strings 1010101 and 1111000, should the random event occur that the strings are to be cut after bit number 4, then the strings would be cut to 1010–101 and 1111–000. The 'tails' of the two strings would then be exchanged.

Crossover is not allowed *within* unit constituents so that, for example, a rule could not be cut in two. The resulting strings are next analysed to check uniqueness relative to the current population by comparing each string bit by bit. Each string found to be unique is placed in the least fit $s\%$ of the population to overwrite rules from previous iterations (but not rules created

from crossover in the current iteration). This process continues until the least fit $s\%$ of the population completely consists of rules created during the current iteration. Note that the 'parents' carry on to the next iteration.

Mutation

Mutation is the process of randomly changing appropriate bits in a strategy string and is executed in a bitwise manner. We maintain an elitist model since the top-ranked 5% of strings are spared mutation. This is done in order to preserve strings with high fitness since in this kind of optimization we are searching for maxima locally as well as globally, i.e. we attempt to generate an array of *good* solutions rather than just the best. After each attempted mutation, the string is checked for uniqueness and if so the mutation is said to be successful. The number of required successful mutations per iteration is user-selected.

Specifically, consider the mutation process with a mutation rate of 2%. The population, excluding the top 5% (as explained above), is earmarked for mutation. If the population of strings contains B bits, numbered 1 to B, then we would wish to mutate the nearest integer value M to $m := 0.02b$ of them. In this case, M uniform pseudo-random integers are generated between 1 and B without replacement, in the manner outlined in the *population initialization* stage. The M integers are ordered by magnitude and for each the correspondingly numbered bit is mutated (0 goes to 1, 1 goes to 0). After each string has been passed by the mutation process, a test is performed for uniqueness (relative to the population) and to see that only allowable bits, i.e. no connector bits, have been changed. If this test is failed then the mutation(s) to which that rule has been subjected are unwound. This process is continued on unmutated rules until the full M mutations have been successfully effected.

Trial termination

For each trial the process described above is repeated (excluding the *population initialization* stage, which is only performed at the first iteration) for a chosen maximum number of iterations or until convergence occurs, whichever is first. There is also a user-selected minimum number of iterations during which the process *cannot* terminate regardless of convergence. *Convergence* is said to occur for each trial of our system when mean fitness of a user-specified number of the most fit strategies and maximum fitness both change by no more than 1% between the previous and current iterations. The whole process may be repeated for a user-specified number of trials—equivalent to different starting points for a general global optimization algorithm—and the best performing 5% user population selected according to the fitness criterion with notional capital distributed equally amongst them or by some other (e.g. voting) rule.

6. Results of one minute trading

We report here five main experiments with the trading system outlined above using a single trial for each. As stated
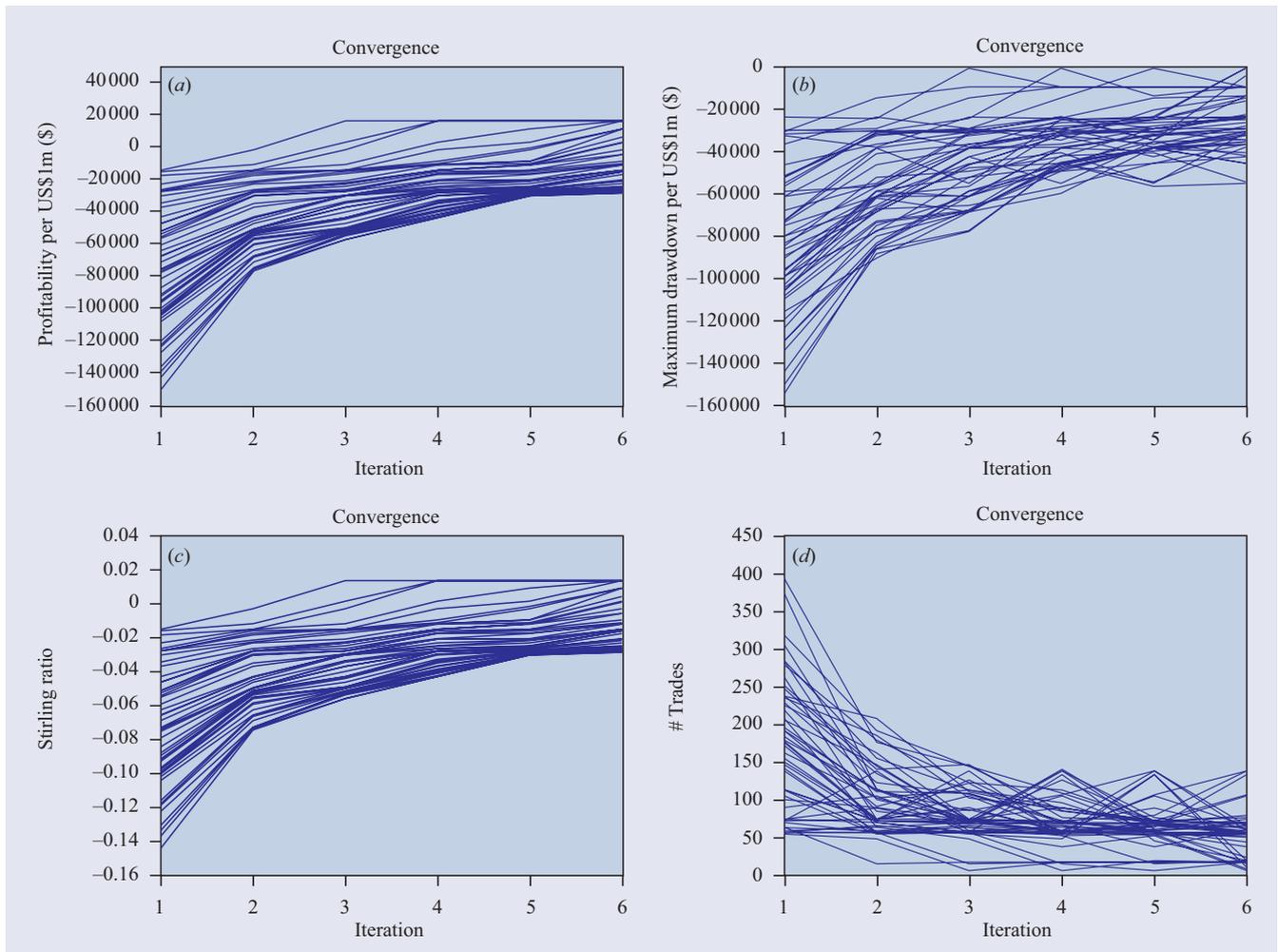
Figure 4. (*a*) Profitability by iteration, (*b*) maximum drawdown by iteration, (*c*) fitness by iteration and (*d*) number of trades per quarter by iteration.

previously, the GP has the potential to choose strategies from the AMA, CCI, MA crossover, PCI, RSI and Stochastic indicator-based rules detailed in the appendix, along with a simple measure of 'change in price'. The Boolean operators AND, OR and XOR were allowed. In all cases, the initial population size was chosen to be 100, the crossover rate 0.5 and the minimum number of iterations to be 6. The low population size reflects the number of possible strategies that can be chosen, since the solution space is much smaller than, for example, that of a standard mathematical integer optimization problem. The slippage model outlined in section 4 was used with a zero fixed cost and only the daily time varying components, so as to emulate a typical slippage encountered by a proprietay FX trader in a bank. With a similar aim, the system was constrained to create strategies based on a maximum of five indicators, since we conjecture that a systematic trader would have approximately such a limit with respect to complexity. This bound should also help to prevent overfitting the data in training periods. Finally, we apply the cash management exit rule to end a trade based on a trailing stop of 100 pips.

In all cases we use only the data set 1993–1997 since, in previous work (Dempster and Jones 2000, Jones 1999) we have shown significant shifts in market behaviour pre- and post-

1992/1993—the period marked by the European Exchange Rate Mechanism crisis. Our system selects strategies based on data from three months (which, we believe, would be similar to a trader's test period) and then tests out-of-sample in a simulation of 'live' trading. The first quarter after optimization/mining is thought of as a validation period. Results are reported quarterly.

Static evaluation with 15 minute indicators

With the above settings we constrain the system to use only indicators evaluated at the 15 minute frequency whilst trading at the minute level with an initial trading capital of $1 million (US) (and its GBP equivalent as explained in section 4). We validate strategies based on Q1 1994 and test successive quarters out-of-sample from Q2 1994 to Q4 1997. The 20 strategies performing best in-sample in terms of Stirling ratio are chosen to be the constituents of the portfolio of trading strategies to be tested out-of-sample. However, only strategies that have been profitable in the sample validation period (quarter) are considered for further trading so that the out-of-sample trading portfolio may contain less than 20 strategies. Should no profitable rules have been found in this validation
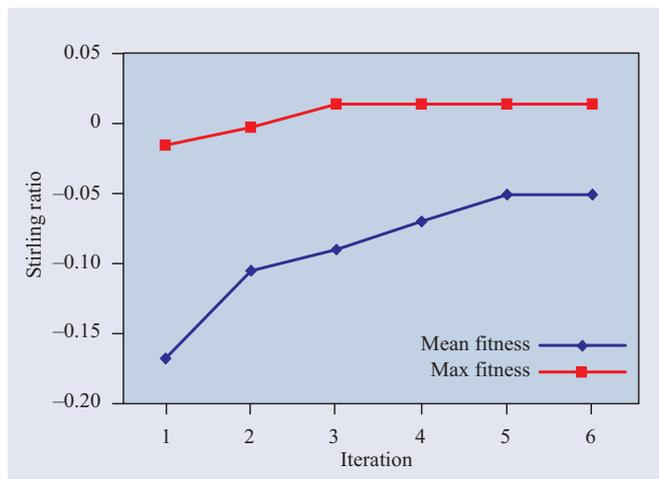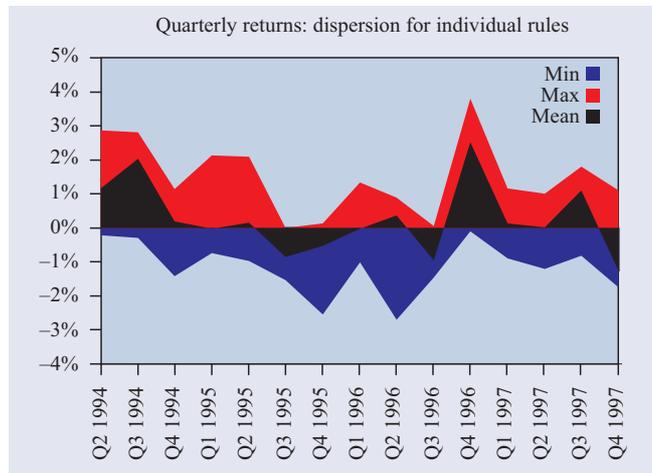
Figure 5. Convergence criteria by iteration.

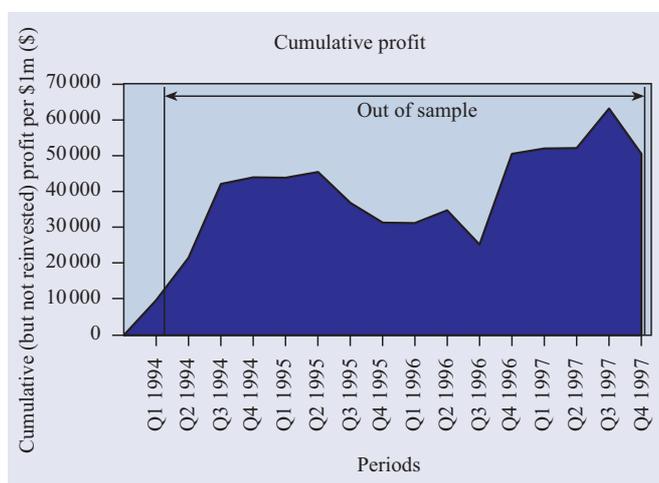**Figure 7.** Dispersion of returns.
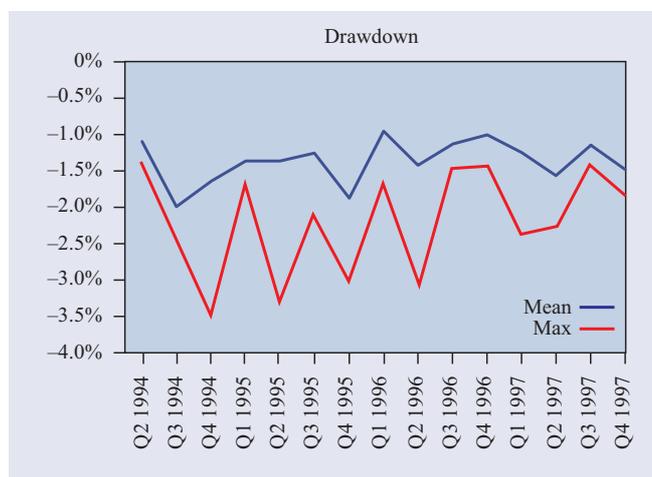

Figure 6. Cumulative profit.

Figure 8. Maximum drawdown by quarter.

period, then trading would have been suspended and the whole process would have been repeated in the next period. Once the maximum of 20 best in-sample strategies are selected, the $1 m 'trading capital' is distributed equally between them for trading and the strategy portfolio is tested out-of-sample.

There is no prior work to suggest that 20 is the optimal number of strategies to pick and this parameter should be user-defined since it is to some extent dependent on the environment in which the trading system operates; for example, in *live trading*, the complexity of the portfolio of strategies should be low enough to allow monitoring. To enhance the system, experiments can be conducted to find a suitable number of strategies in a systematic manner. However, the purpose of the initial work reported here is to test the system in a general framework rather than to optimize over all parameters; this way we hope, in the first instance, to avoid 'cherry-picking' and 'data-snooping'.

A high mutation rate—0.02—is used to ensure diversity of strategies and the maximum number of iterations allowed is 15, chosen by constraints on computation time—each iteration takes around three hours on a Pentium 400 PC with 256

MB of RAM running under Windows NT Workstation 4.0 since it processes over 100 million data points. For this first experiment, we analyse the genetic program in-sample iteration process in detail.

In this experiment, convergence was achieved within six iterations. Figures 4(*a*)–(*d*) show the convergence of the top 50 strategies in terms of profit, drawdown and risk-adjusted fitness. It can be seen that the profitability of top strategies increases with each iteration, while maximum drawdown diminishes and, as a result, fitness improves. Note from figure 4(*d*) that the number of trades also reduces as the process continues and prospective strategies are demoted for overtrading.

Figure 5 shows the convergence criteria—mean and maximum fitness for the top 20 strategies—being attained. It can be seen that the best-performing strategy was found quickly and that mean fitness steadily improved throughout the process.

Figure 6 shows the cumulative excess profit (not reinvested) based on two-way trading of US $1 m and it can be seen that the system is profitable in the validation and out-of-sample periods although profits at around 5% total are not large.
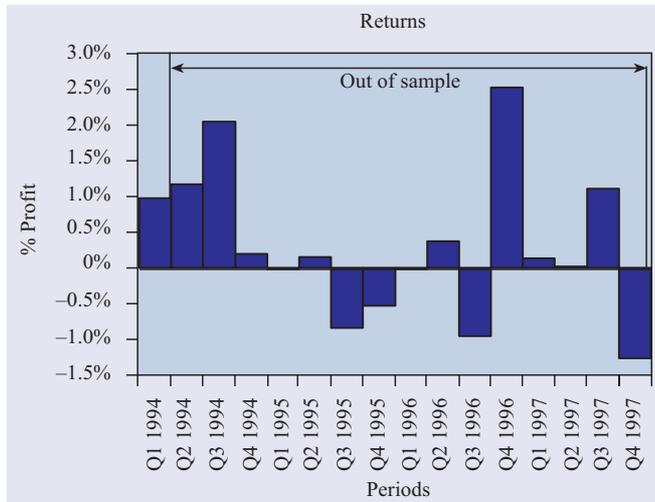
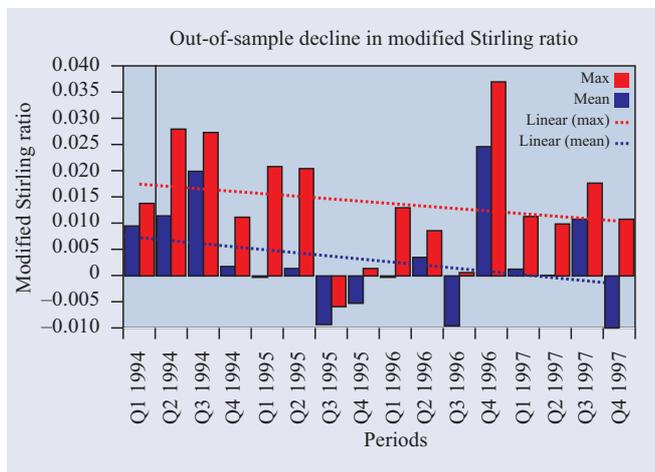**Figure 9.** Out-of-sample returns of the portfolio of chosen rules.

Figure 10. Out-of-sample tail-off of optimization criteria.

Figures 7 and 8 show that the array of strategies traded does not have a high dispersion of returns but that, by trading a combination of the strategies rather than an *individual strategy* alone, *maximum drawdown is reduced* and extreme negative returns avoided at the cost of reduced cumulative profit—a risk return trade-off.

Analysing returns in greater depth shows that, further out-of-sample, returns become more volatile (as can be seen in figure 9). Regarding formal statistical testing of the significance of technical trading profits, first note that for our high-frequency data the powerful bootstrap approach (Levich and Thomas 1993) is not feasible on current machines. To determine the statistical significance of our trading results, we therefore utilize the following simple non-parametric binomial test. Assume as the null hypothesis that cumulative trading profits and losses are periodically sampled from a continuous time stationary ergodic process with state distribution having median zero. Note that this process assumption requires no finite moment assumptions and is consistent with heavy-tailed return distributions. Then under the null hypothesis cumulative quarterly profits or losses are equally likely and their signs
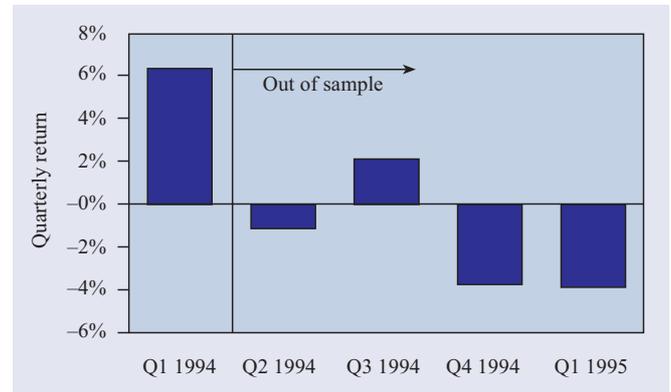

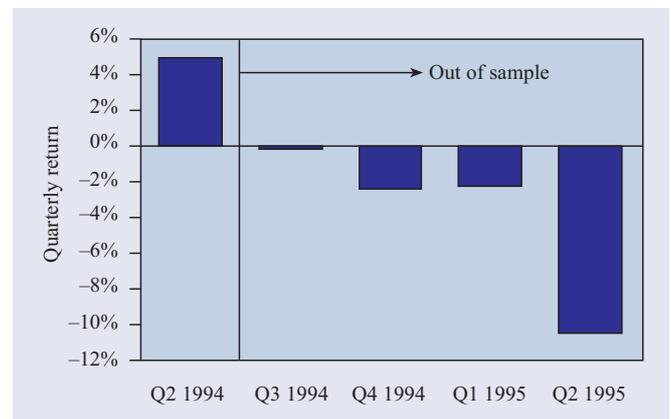**Figure 11.** Trading results for strategies using daily and 15 minute rules—first attempt.

Figure 12. Trading results for strategies using daily and 15 minute rules—second attempt.

are positive (profit equals 1) or negative (loss equals 0) with probability $1/2$. It follows that over n quarterly periods the number of profitable quarters n_+ is binomially distributed with parameters n and $1/2$. We may therefore test the two-tailed alternative hypothesis that median profit and loss is non-zero with the statistic n_+.

For the equally weighted portfolio of 20 best strategies whose quarterly profits or losses are depicted in figure 9, this binomial statistic $n_+ = 8$ over 15 quarters with a p-value of 50%—total insignificance—in spite of a cumulative return of 4.8% per annum (pa) in the out-of-sample period. On the other hand, the returns of 7% pa to the *best strategy*, with $n_+ = 14$ and a p-value of 99.8%, are significantly profitable at the 0.2% level as can be seen from figure 10.

Figure 10 also shows out-of-sample declines in mean and maximum modified Stirling ratios—the selection criteria—which have non-zero negative slope regression coefficients which are statisically significant at the 5% level out-of-sample. This would suggest that adaptation in the form of partial or constant reoptimization might improve performance and we will present such strategies in experiments 3 and 4 below.

Static evaluation with 15 minute and daily indicators

With the same settings as previously, our second experiment allowed the system to use indicators evaluated at both the
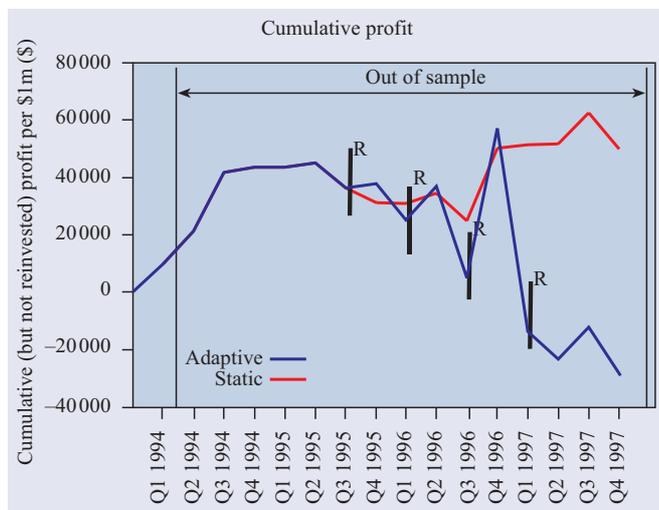
Figure 13. Cumulative strategic portfolio profit.

15 minute and daily frequencies whilst trading at minute level. As shown in figure 11, the strategy portfolio system is unprofitable when tested out-of-sample (Q2 1994–Q2 1995). Similarly, when reoptimized and revalidated in the next data period (Q2 1994), this system still fails to make a profit, as shown in figure 12. Since the addition of other evaluation frequencies does not appear to enhance results, no further attempt to include other frequencies is made in this paper, the principle having been tested and failed. However, this process should not be abandoned; these poor results may be due to overfitting in the optimization process in the training period, since in this experiment strategies have a greater potential for complexity given that information at the daily frequency is also available. In-depth analysis of many trials of the optimization process would be needed to support this theory and, if established, an increased data-mining period could be implemented as a potential solution. Furthermore, in this experiment strategies have the potential to contain many more different rules and so it may be that a much greater population size is needed. Although computation time (in hours) precludes the possibility of exploring this area further at present, further speed-related enhancements to the system and ever-increasing processor speed should make this analysis viable as further work. (In this regard see Dempster *et al* (2000), whose results however are not strictly comparable to the present results due to the one-sided nature of the trading in the cited paper.)

Adaptive optimization

In this third experiment, we use only the 15 minute indicators used in the first experiment. As noted above, performance decreases significantly as time out-of-sample increases, which would suggest the need for some form of reoptimization. In this experiment we *reoptimize* either when the system makes a quarterly loss greater than 1% or when the system has experienced two losing quarters in the last 12 months with a given set of strategies. All other settings remain the same as in the first experiment. Thus, the reoptimization here is
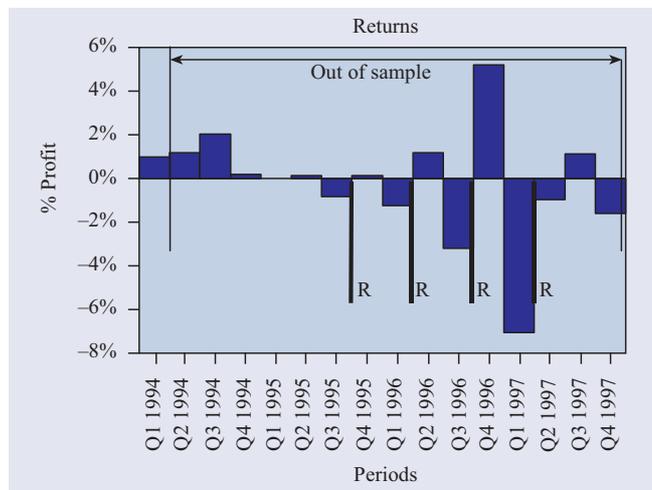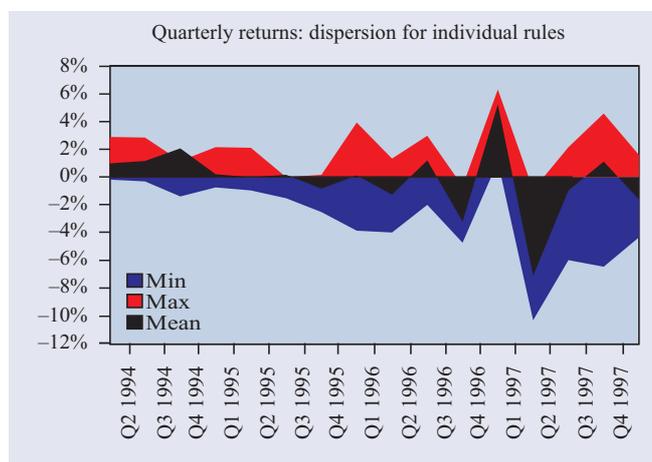

Figure 14. Out-of-sample returns.

Figure 15. Dispersion of returns.

triggered by feedback from the system's performance in the same way that a trader would be compelled to find new rules after losses.

Once again, the system was initially run on strategies validated over the period Q1 1994. Performance dictates that no further optimization is necessary until Q3 1995 when there are two losses in a 12 month 'lookback' period. The strategies are subsequently reoptimized over Q1 1996, Q3 1996 and Q1 1997, when the optimization is triggered by losses in excess of 1% in each case.

Figure 13 shows that adaptation (in fact, reaction) in this way results in the strategic portfolio underperforming (in terms of profitability) with respect to the static portfolio. In fact, the adaptive portfolio makes a loss, whereas in the static case the system was (insignificantly) profitable. After the points of reoptimization (marked 'R' on figures 13 and 14), the system generally makes a loss within two periods. One possible explanation for this behaviour is that, as a result of over-reaction to the market, the reoptimization procedure results in the overfitting of strategies to specific market conditions that are not typical of the market as a whole, leading to perpetual loss by 'chasing losses'. Applying the binomial test
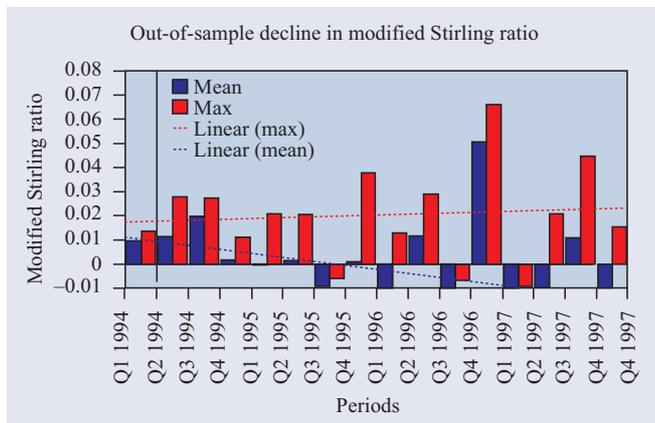
Figure 16. Out-of-sample behaviour of optimization criteria.

Figure 17. Cumulative profit.

for significant profits or losses to these results with $n_+ = 8$, see figure 14, again yields 'total' insignificance.

Figure 14 also shows that the return structure of the portfolio is relatively volatile with respect to the static case. Figure 15 appears to show that the dispersion of returns increases as time progresses, resulting in further reoptimization and seemingly further losses. As a whole, strategic portfolio performance is worse than for the static case as can be seen from figure 13.

Figure 16 shows however that re-optimization results in stability of the performance (in terms of the optimization criterion) of the *best-performing strategy* in each period—which was not found in the static case (*cf.* figure 7)—but a *decline in the mean performance*. Indeed, with binomial test statistic $n_+ = 12$ and *p*-value 98.2%, the best strategy achieved a 7.4% pa return—versus 4.8% for the static portfolio strategy—and was significantly profitable at the 5% level. Note, however, that the *best*-performing strategy in a particular period may be the *worst*-performing strategy in the next, so that implications with regard to optimal portfolio size are difficult to draw. Best-performing strategies in this experiment outperform those of the first experiment in many periods but there is also potential for a greater downside in returns, as can be seen in figure 14. The obvious decline in modified Stirling ratio shown in figure 16 has significantly negative regression slope at the 5% level for the strategy portfolio, while the marginal increase for the best strategy has slope not significantly different from zero.

Periodic reoptimization

In this experiment, we continue to use only 15 minute indicators. We have seen that performance decreases as time out-of-sample increases, which would suggest the need for some form of reoptimization. However, the adaptive reoptimization experiment showed that by reacting to market loss the strategy portfolio underperforms its static equivalent in terms of total return (*cf.* figure 13). In the present experiment we attempt to be proactive, rather than reactive, with respect to market changes. Thus, we reoptimize periodically every quarter and trade the recently discovered optimal rules for one
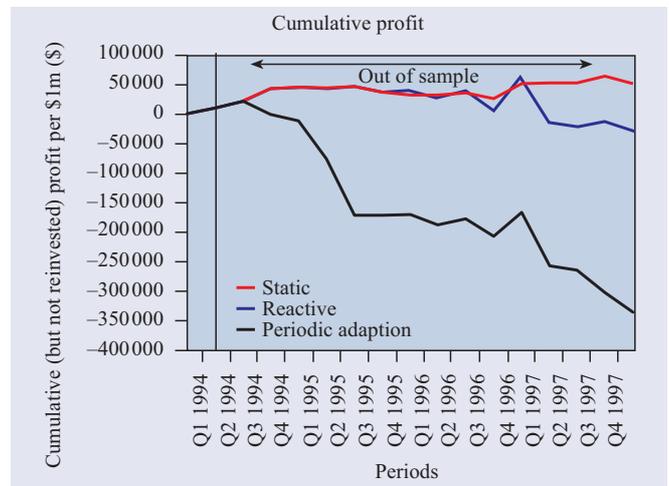
quarter out-of-sample before restarting the process. All other settings remain the same as in the first experiment.

From figure 17 we see that the out-of-sample performance of this 20 portfolio strategy is much worse than the static or reactive portfolio strategies. In this case, even the best strategy is loss making although the strategic portfolio value is not *significantly* loss making with binomial test statistic $n_+ = 5$ and *p*-value 15.1%. One potential reason for this is the 'modal' nature of the market; optimal rules are chosen when the market is behaving in a particular fashion or mode and then, as time passes, only perform well when this mode reappears. This can be seen in the return structure of the first experiment (figure 7) where the initial profitability recedes and then returns. By reacting to the market, rule sets are replaced after large losses but, since these are relatively infrequent, the resulting drop in profitability is small in comparison with the periodic reoptimization. Here, when a set of optimal strategies are mined, they only have one out-of-sample period to perform before being replaced, which may not be long enough for the market to revert to its in-sample behaviour. There are a number of solutions to this problem, the most obvious being to use a longer mining period (the optimization period has been chosen here to be three quarters primarily to achieve a tractable computation time). Another solution may be to run the optimization algorithm separately over a number of quarters and select a best strategy or representative sample of strategies from each solution set to use in construction of the trading system; this way, several market modes have the potential to be represented. In any event, this area requires further investigation (see Dempster *et al* (2000)).

Parallel optimization

In this experiment, we implement a simple parallelization of the trading system with a view to speeding up the optimization process as well as potentially enriching the quality of solutions (*cf.* Holland (1986) and Leinweber and Arnold (1995)). The existing optimization process is started remotely and simultaneously on a number of identical machines (P400s with 256 MB of RAM and 10 GB hard disks running MS Windows
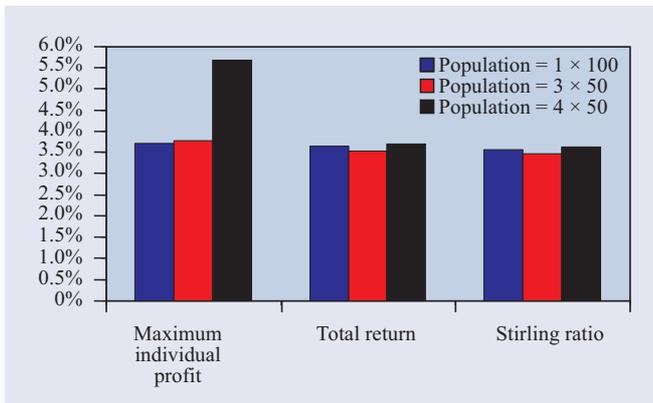
Figure 18. In-sample results of three- and four-machine parallelizations against benchmark.

Figure 19. Out-of-sample results of three- and four-machine parallelizations against benchmark.

NT Workstation 4.0 and MPI over a 10 Mb/s Ethernet) using the same settings as used previously but with a reduced initial population size of 50 (as opposed to 100) on each machine. After the slowest run is complete, the final populations are recombined with any identical rules eliminated and all the remaining strategies reordered in terms of fitness. As with previous experiments, the top 20 strategies (or less if there are not 20 strategies that are profitable in-sample) are selected and tested out-of-sample.

Here we use the Q2 1994 validation results (part of the previous experiment) as a benchmark since this was the first in-sample period in which all 15 iterations were completed. In the previous in-sample period, convergence was reached after six iterations and so any speed-up achieved for these data would not be a true reflection of the merits of parallelization in that case.

The simple parallel process described above was implemented on three and then four machines. The in- and out-of-sample results are displayed in figures 18 and 19, respectively. These graphs compare results in- and out-of-sample, respectively, in terms of the maximum individual profit of the best-performing strategy, the total return of the portfolio of the best 20 strategies and the modified Stirling ratio of that portfolio. In both three- and four-machine experiments, a speed more than 1.75 over the benchmark was achieved, which nearly halved the long computing times.

From figure 18 it can be seen that, when spread over three and four machines, an in-sample optimal value that is superior to the benchmark was achieved in terms of individual rules (maximum individual profit). In addition, the total in-sample return and Stirling ratio of the benchmark was improved by the results of the four-machine parallelization for the strategy portfolio. It can be seen from figure 19 that although negative, the out-of-sample total return results of both parallel procedures were better than those of the benchmark (note that the Stirling ratio is not presented since it is meaningless for negative results). Finally, it should be noted that the four-machine parallel procedure out-performed the benchmark with respect to out-of-sample total return despite having the worst individual losing rule (out-of-sample)—furthermore, this rule was the second best performer in-sample!
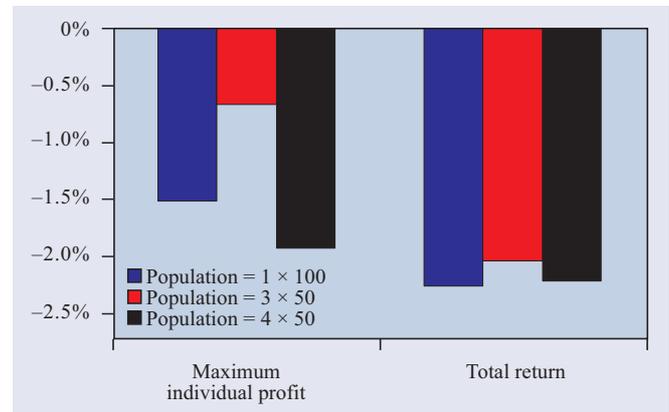
To summarize, our parallel implementation has achieved good speed-up without compromising the nature of the results. In fact, superior performance both in- and out-of-sample (with respect to the 'serial' benchmark) was achieved by the four-machine parallel strategy portfolio process.

7. Conclusions and future directions

We have shown in previous work (Dempster and Jones 2000) that the majority of indicator-based rules (with the same parameters used here) are loss-making when traded individually or collectively. However, we have found in this paper that it is possible to profit from trading technical rules when the trade entry signal is taken from combinations of indicators and we note that this is the manner in which a technical trader would use such indicators. The return from the 20 strategy portfolio system trading at 15 minute entry intervals is small and statistically insignificant and is in fact less than the interest rate differential between pounds and dollars over the same time period. This implies that better excess returns would be available from a static buy and hold strategy (based on three-month Euro sterling and Euro Dollar rates[10]). However, such a return would not afford the minute-by-minute liquidity that the system allows and, as a result, an investor would not be able to activate funds in the face of adverse market movements. However, when only the *best* strategy is employed it is modestly and statistically significantly profitable (returning 7% pa).

When traded in an adaptive manner, however, we find that the resulting portfolio strategies are ultimately loss-making, highlighting the penalty for over-reaction to short-term market behaviour. These findings are in accordance with work at lower frequencies that considers asset allocation strategies (Hicks Pedrón 1998) which showed that over-reaction to short-term market behaviour should be avoided in favour of rebalancing portfolios less frequently.

On the other hand, utilizing adaptively only the best strategy results in statistically significant profitability yielding 7.4% pa—40 basis points better than its static equivalent.

[10] Source: Bank of England.

Also, we note that for high-frequency FX trading the addition of information sampled at a lower (daily) frequency heavily reduced performance. Since the addition of a second entry frequency increases the complexity of the rules, it is highly probable that these disappointing returns are a result of overfitting. Improvement in this area may be gained by increasing population size and/or the length of the mining period (see Dempster *et al* (2000)). Finally, we have implemented a simple parallel procedure that, as well as providing a considerable speed-up, has produced better results with regard to profitability of the 20 strategy portfolio approach.

The above work is just a sample of what can be produced using such a system. Further adaptation techniques could be attempted; for example periodic reoptimization with longer and shorter periods could be analysed along with testing various triggers to reoptimization based on performance and market behaviour (*cf.* Dempster *et al* (2000)). Previously, we have suggested that the market may move between various modes and so it may be that analysing the market itself may hold the key to when it is optimal to reoptimize the trading strategy portfolio. Also, as well as testing the performance of the trading strategy portfolio using various mining period lengths, it may also be possible to include strategies discovered over different mining periods and hence create portfolios that perform well under different market conditions.

Further analysis of portfolio size (in terms of number of strategies) and other parameters, such as mutation rate and number of iterations, would also be of interest.

The system's ability to consider information at various frequencies and lags can also be investigated further. Although the initial results from including indicators evaluated at the daily frequency were disappointing, more attention has been paid to this area and the promising results will be reported elsewhere.

As was previously stated, this first test of our system aims to show that such a structure merits further work by analysing results in a general and unoptimized framework. Thus, initial results cannot be criticized as being a result of excessive *data-snooping* (the reuse of data to enhance results). However, now that such a validation has been completed, the way is left clear for further work to be completed on enhancements and improvements (see e.g. Dempster *et al* (2000)).

The main barrier to cross when attempting such further work is computation time. Each iteration of the algorithm reported here evaluates over 10 million data points and takes between two and four hours on a P400 PC with 256 MB of RAM. However, preliminary results from a simple parallel implementation to this system show increases in both the quality of solution and the speed at which such solutions are produced. Such calculation times could be reduced further by utilizing the inherent parallel nature of genetic algorithms and programming by developing algorithms that calculate fitness of strings in parallel as well as further distribution across machines in the style of our current parallel implementation.

Appendix

For further details on all of the indicators below, see Jones (1999).

Simple moving average crossover (SMA)

The arithmetic average of the last n bars (starting from the current bar). Formally, we define the n bar *simple moving average* at bar m ($m \geqslant n, n > 0$), $\mathrm{SMA}(m, n)$, as follows

$$\mathrm{SMA}(m, n) := \frac{1}{n} \sum_{i=0}^{n-1} c_{m-i},$$

where c_i is the closing price at bar i. Note, that $\mathrm{SMA}(m, 1) = c_m$.

Enter a long trade when a shorter term moving average ($n = 3$) crosses a longer term moving average ($n = 15$), enter a short trade when the opposite occurs.

Adaptive moving averages (AMA)

We define the adaptive moving average (AMA) as follows:

$$\mathrm{AMA}(k, m) := \alpha_{k,m} c_m + (1 - \alpha_{k,m}) \mathrm{AMA}(k, m - 1),$$

where

$$\alpha_{k,m} := \frac{|c_m - c_{m-k-1}|}{\sum_{i=m-k}^{m} |c_i - c_{i-1}|} \quad (= 0.15).$$

When trading using moving average crossovers, shorter term indicators ($n = 2$) are compared with longer term indicators ($n = 30$) with a hope of discovering the emergence of new trends. When trading using the AMA, we only consider the one indicator since, depending on market conditions, the AMA can exhibit characteristics of shorter term or longer term moving averages. Instead, the AMA is usually traded based on its current position relative to its recent history. First local highs (*lh*) and lows (*ll*) are defined as follows:

$lh_i := \mathrm{AMA}(k, i)$ if $\mathrm{AMA}(k, i) > \mathrm{AMA}(k, i - 1)$

$lh_i := lh_{i-1}$ otherwise;

$ll_i := \mathrm{AMA}(k, i)$ if $\mathrm{AMA}(k, i) < \mathrm{AMA}(k, i - 1)$

$ll_i := ll_{i-1}$ otherwise.

Then, a *short trade* is placed if the value of the AMA moves a fixed amount (F, say) below the value of the local high and a *long trade* is placed if the value of the AMA moves a fixed amount F above the value of the local high. In the analysis above, we consider F to be a fixed 20 pips (as is usual). However, F is not always fixed in practice, but is often based on the standard deviation of close price over the last few bars (see Kauffman (1998)).

Price channel breakout (PCB)

The n ($= 10$) period *trading range* at bar m ($m > n$) is defined by the following *upper* ($\mathrm{U}(m, n)$) and *lower* ($\mathrm{L}(m, n)$) bounds as follows:

$$\mathrm{U}(m, n) := \max(c_{m-1}, c_{m-2}, \ldots, c_{m-n})$$

$$\mathrm{L}(m, n) := \min(c_{m-1}, c_{m-2}, \ldots, c_{m-n}),$$

where c_i is the close price of bar i.

The trading rule for the PCB is very simple: buy long when $c_m > U(m, n)$, sell short when $c_m < L(m, n)$. Thus, in the absence of cash management exit strategies, the trading rule ensures that a market position is always held ('always in the market').

The stochastic

The n ($= 10$) period *stochastic*, $K(m, n)$ at bar m ($m \geqslant n$) is defined as follows:

$$K(m, n) := \frac{c_m - \text{Low}(m, n)}{\text{High}(m, n) - \text{Low}(m, n)},$$

where $\text{High}(m, n) = \max(c_{m-1}, c_{m-2}, \ldots, c_{m-n})$ and $\text{Low}(m, n) = \min(c_{m-1}, c_{m-2}, \ldots, c_{m-n})$, and c_i is the close price of bar i.

In the analysis of stochastics we work with the three-period moving average of $K(m, n)$, usually referred to as $SK(m, n)$ and the three-period moving average of SK, referred to as $SD(m, n)$. Formally

$$SK(m, n) := \sum_{j=0}^{2} K(m - i, n)$$

and

$$SD(m, n) := \sum_{j=0}^{2} SK(m - i, n).$$

There are many rules for trading using stochastics, some of which are highly subjective and cannot be easily automated. We have chosen to test a commonly used rule that is neither complex nor subjective. Here we *buy long* when $SK(m, n) > SD(m, n)$ and *sell short* when $SK(m, n) < SD(m, n)$. Once more, in the absence of cash management exit strategies, this is an 'always in the market' strategy.

The relative strength index (RSI)

The n ($= 10$) period *relative strength index* $RSI(m, n)$, at bar m ($m > n + 1$), is defined as follows

$$RSI(m, n) := \frac{RS(m, n)}{1 + RS(m, n)},$$

where

$$RS(m, n) := \frac{EG(m, n)}{EL(m, n)},$$

in which $EG(m, n)$ and $EL(m, n)$ are weighted averages of gains and losses, respectively, so that

$$EG(m, n) := \frac{\max(c_m - c_{m-1}, 0) + (n - 1)EG(m - 1, n)}{n}$$

and

$$EL(m, n) := \frac{\max(c_{m-1} - c_m, 0) + (n - 1)EL(m - 1, n)}{n},$$

with

$$EG(n + 1, n) := \frac{1}{n} \sum_{i=2}^{n+1} \max(c_i - c_{i-1}, 0)$$

and

$$EL(n + 1, n) := \frac{1}{n} \sum_{i=2}^{n+1} \max(c_{i-1} - c_i, 0)$$

where c_i is the close price of bar i.

The RSI is used in a contrarian fashion with the aim of identifying markets that are 'overbought' and 'oversold'. When trading using the RSI, it is usual for the market to be considered overbought when the RSI is above 70 and a *short position* taken. Similarly, the market is thought to be oversold when below 30 and a *long position* taken.

The commodity channel index

The n ($= 10$) period *commodity channel index* $CCI(m, n)$, at bar m ($m \geqslant n$), is defined as follows

$$CCI(m, n) := \frac{M(m, n) - \bar{M}(m, n)}{0.015 \bar{D}(m, n)},$$

and

$$M(m, n) := \tfrac{1}{3}(h_m + l_m + c_m),$$

$$\bar{M}(m, n) := \frac{1}{n} \sum_{i=0}^{n-1} M(m - i, n)$$

and

$$\bar{D}(m, n) := \sum_{i=0}^{n} \left| M(m - i, n) - \bar{M}(m, n) \right|,$$

where h_i, l_i and c_i, are the bar high, low and close, respectively.

The most common trading strategy is to *buy* when the CCI rises above $+1$ and sell the long position when the CCI falls back below $+1$. Similarly, a *short position* is entered when the CCI falls below -1 and that position is closed when the CCI rises once more above $+1$.

References

Allen F and Karjalainen R 1999 Using genetic algorithms to find technical trading rules *J. Financial Economics* **51** 245–79

Bank of International Settlements 1998 *Central Bank Survey of Foreign Exchange and Derivatives Market Activity* www.bis.org.

Colby R and Meyers T 1988 *The Encyclopaedia of Technical Market Indicators* (Homewood, IL: Irwin)

Deboeck G (ed) 1994 *Trading on the Edge: Neural, Genetic and Fuzzy Systems for Chaotic Financial Markets* (Chichester: Wiley)

Dempster M A H and Jones C M 1999a Can technical pattern trading be profitably automated? 1. The channel *Working Paper 11/99* Judge Institute of Management, University of Cambridge (revised as: 2001 Can channel pattern trading be profitably automated? *Euro. J. Finance* at press)

——1999b Can technical pattern trading be profitably automated? 2. The head and shoulders *Working Paper 12/99* Judge Institute of Management, University of Cambridge

——2000 The profitability of intra-day FX trading using technical indicators *Working Paper 35/00* Judge Institute of Management, University of Cambridge

Dempster M A H, Payne T W, Romahi Y S and Thompson G W P 2000 Computational learning techniques for intraday FX trading using popular technical indicators *Working Paper 30/00* Judge Institute of Management, University of Cambridge (2001 *IEEE Trans. Neural Networks* **12.4** at press)

Dunis C and Zhou B 1998 *Nonlinear Modelling of High Frequency Financial Time Series* (New York: Wiley)

Gencay *et al* 1998 Real time trading models and the statistical properties of foreign exchange rates *Working Paper* Olsen & Associates www.olsen.ch

Guillaume *et al* 1997 From the bird's eye to the microscope: a survey of new stylized facts of the intra-daily foreign exchange markets *Finance Stochastics* **1** 95–129

Haupt R and Haupt S 1998 *Practical Genetic Algorithms* (New York: Wiley)

Hicks Pedrón N 1998 Model based asset management: a comparative study *PhD Thesis* Judge Institute of Management, University of Cambridge

Holland J H 1976 Studies of the spontaneous emergence of self-replicating systems using cellular automata and formal grammars *Automata, Languages, Development* ed A Lindenmeyer and G Rozenberg (Amsterdam: North-Holland)

——1980 Adaptive algorithms for discovering and using general patterns in growing knowledge-bases *Int. J. Policy Analysis Information Systems* **4** 217–40

——1986 Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems *Machine Learning II* ed R S Michalski, J G Carbonell and T M Mitchell (New York: Morgan Kaufmann)

——1992 *Adaptation in Natural and Artificial Systems* (Cambridge, MA: MIT Press)

James J and Thomas M 1998 A timely exit *Risk* 74–6

Jones C M 1999 Automated technical foreign exchange trading with high-frequency data *PhD Thesis* University of Cambridge

Kauffman P 1998 *Trading Systems and Methods* (New York: Wiley)

Koza J 1992 *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (Cambridge, MA: MIT Press)

Leinweber D J and Arnott R D 1995 Quantitative and computational innovation in investment management *J. Portfolio Management* 8–15

Levich R and Thomas L 1993 The significance of technical trading rule profits in the foreign exchange markets: a bootstrap approach *J. Int. Money Finance* **12** 451–74

Lui Y-H and Mole D 1998 The use of fundamental and technical analyses by foreign exchange dealers: Hong Kong evidence *J. Int. Money Finance* **17.3** 535–45

Moody J *et al* 1997 Performance functions and reinforcement learning for trading systems and portfolios *J. Forecasting* at press

Neely C 1997 Technical analysis of the foreign exchange market: a layman's guide *Federal Reserve Bank of St Louis Review* 23–37

Neely C and Weller P 1999 Intraday technical trading in the foreign exchange market *Technical Report WP-99-016AM* Federal Reserve Bank of St Louis

Neely C, Weller P and Dittmar R 1997 Is technical analysis in the foreign exchange market profitable? A genetic programming approach *J. Financial Quant. Anal.* **32** 405–26

Pardo R 1992 *Design, Testing and Optimisation of Trading Systems* (New York: Wiley)

Pictet O *et al* 1992 Real-time trading models for foreign exchange rates *Neural Network World* **6** 713–44

Refenes A-P (ed) 1995 *Neural Networks in Capital Markets* (Chichester: Wiley)

Schwager J 1996 *Technical Analysis (Schwager on Futures)* (New York: Wiley)

Taylor M and Allen H 1990 Charts, noise and fundamentals in the London foreign exchange markets *Economic J.* **100** 49–59

——1992 The use of technical analysis in foreign exchange markets *J. Int. Money Finance* **11** 304–14