Adaptive Portfolio Trading Using Genetic Algorithms

Arthur Rabatin
Rabatin Investment Technology Ltd

Fifth International Conference on Forecasting Of Financial Markets
(London, May 1998)

Attention:

Isabelle Cremieux

BNP
8-13 King William Street,
London EC4P 4HS

Author:

Arthur Rabatin
Rabatin Investment Technology Ltd
11 Grosvenor Place, 5th Floor
London SW1X 7HH
Tel. 0171 887 0 887
Fax 0171 887 0 888
research@rabatin.com

Table Of Contents Page

1. Abstract 3
2. Introduction 3
3. Trading Behaviour Learning Process 4
3.1 Market Timing Decision 4
3.2 Risk and Portfolio Allocation Decisions 7
3.2.1 Price Risk Calculation 8
3.2.2 Portfolio Allocation Decision 8
3.2.3 Portfolio Risk Decision 9
4. Performance Measurement and Fitness Target 10
5. Cross-Validation and Learning Process Setup 11
6. Results - Leveraged Foreign Exchange Portfolio 12
6.1 Overview 12
6.2 Portfolio Results 13
6.2.1 Portfolio Performance Results 14
6.2.2 Performance Consistency and Predictability Measurement 14
6.3 Portfolio Results - Conclusion 16
7. Conclusion 16

1. Abstract

Adaptive Portfolio Trading involves the application of artificial intelligence (AI) computer programmes with the ability to learn their own behaviour rules, and to modify these rules during time. Such computer programmes can be designed and trained to perform successfully the functions of a human trader in the position of a portfolio manager.

This paper describes various aspects of the implementation of such an adaptive behaviour process, focusing on the trading of diversified, multi-currency portfolios. The example of a diversified Foreign Exchange portfolio shows how successful results can be achieved over several years of data, including data for training and testing of such a model.

The underlying AI technology applied to the modelling process are Genetic Algorithms, parallel processing non-linear search algorithms which allow effective integration of numerical and behavioural optimisation processes. The implementation is based on our Evolving Programming Library (EPL) which provides a Genetic Algorithm and Genetic Programming framework for single-threaded and distributed training process.

2. Introduction

Portfolio Trading presents a particular challenge to an adaptive trading model. In the context of a diversified portfolio, each trading decision must not only decide on the market direction, but also on price risk, portfolio allocation and portfolio risk.

All decisions must be taken within portfolio constraints, which are normally defined for the portfolio as a whole, with the system deciding on how to allocate risk between individual market components within these constraints.

Artificial Intelligence (AI) in finance is mostly applied to time series forecasting. Translating price forecasting results into a portfolio trading strategy however requires a number of risk and allocation decisions, which influence the resulting performance of the portfolio. Unless every market forecast is extremely accurate, even small changes in portfolio allocation or risk management can turn a theoretically profitable forecasting strategy into an unpredictable distribution of profits and losses.

Trading models developed within our framework do not focus on price forecasting, but on establishing consistent portfolio trading performance. Trading models therefore are trained to learn behaviour patterns that should be as independent as possible of the actual distribution of data during the learning process.

The trading model design described in this paper is based on the use of Genetic Algorithms (GA) to create such self-learning, adaptive trading models. Genetic Algorithms are a very flexible tool to integrate different types of decision making processes into one parallel processing learning algorithm. Especially the integration of market timing decisions and risk management decisions involves the combination of different types of behaviour patterns.

As a demonstration of an adaptive trading model, the results of a leveraged foreign exchange portfolio are shown.

3. Trading Behaviour Learning Process

The decision making process of the adaptive trading models simulates the human decision making process by learning behaviour patterns that are matched against patterns the system detects in the environment data. The behaviour of the trading model is therefore a result of events that take place in the environment - both being the market data and it's own accounting database.

It is the purpose of the system's training process to learn to detect what constitutes an *event* and what is the appropriate behaviour in response to it.

Because *behaviour* in the context of a trading model is always a decision to sell, to buy, or to "do nothing" (keep the current position unchanged) in a particular financial instrument, it also means that risk for the portfolio may either be increased (by establishing a new net position in a market) or decreased (by reducing an open net position in any market)[1]. The trading model is designed to retrieve information from both external data (market prices and other data) and from its own trading performance to decide on its trading decisions. These trading decisions (and subsequently decisions on any open positions) are also subject to the risk threshold placed upon the system by the trading manager or system supervisor.

Every trading decision has four components that simultaneously define the portfolio performance: market timing - price risk - portfolio allocation - portfolio risk. Below, the learning process for each of the components, and the integration into a final trading decision are described.

3.1 Market Timing Decision

The process of mapping environment data patterns to a trading behaviour pattern is implemented through two layers of objects that perform these functions: :a layer of objects performing calculations on data (*Calculation Node Objects*) and a layer of objects interpreting the results of calculations as events and mapping these events to possible decisions.

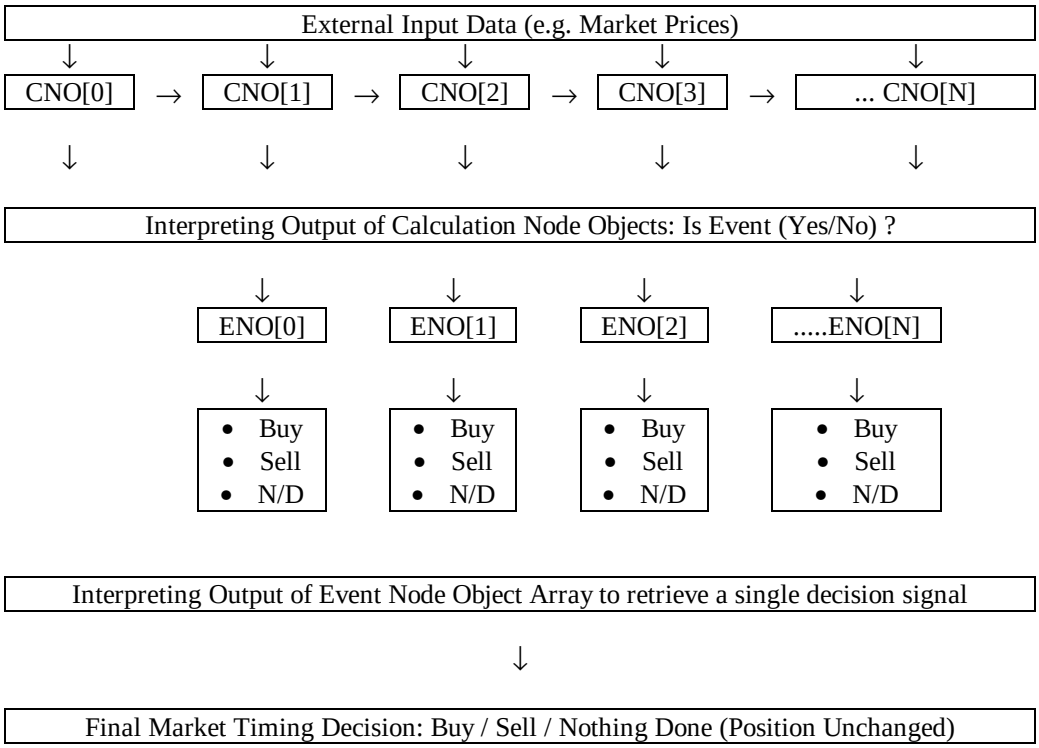
Calculation Node Objects (CNOs) are an array of objects that retrieve data and perform mathematical, statistical and logical operations on these data, returning a set of numerical values that will be interpreted by another layer as events. It is through the GA process that it is decided for each CNO which data are retrieved and which operator is applied. The calculation that is performed within each CNO can be either very simple (such as a logical comparison { e.g. ">" or "="} or a simple arithmetic operation {e.g. "+", "/" ...}), or it can involve complete statistical calculations, such as the historical volatility of a retrieved time series. Boolean values (true/false) are represented as numerical values (0 / 1).
CNOs are not limited to retrieving input only from an external database. The output of each CNO can also be selected in the learning process as input for calculations. This allows the trading model to detect complex patterns by connecting a large number of CNOs that each perform very simple calculations. In an array of (n > 1) number of CNOs, the n^{th} CNO can use the output of (CNO[0] ... CNO[n-1]) as input.
The actual number of Calculation Node Objects is a matter of trading model design and is largely dependent on available machine time, as increased complexity also dramatically increased the time required to train these trading models. The advantage of using an array of CNOs is the flexibility that otherwise would be restricted by a fixed length Genetic Algorithm.

Event Node Objects (ENOs) are an array of objects that interpret output values of CNOs by learning to map the CNO output to a logical value of true/false in terms of an *event* occurring yes/no.
Because the output of each CNO is a fixed structure of values that is always assigned valid data (independent of the calculation performed with a CNO), ENOs can always interpret the result of a calculation in terms of an event occurring or not.
Event Node Objects also learn to map a particular CNO output set to a particular type of decision - to buy, sell or do nothing in a market.

[1] Portfolio Risk is here referred to as the aggregated risk of all positions, not the variance of the portfolio returns itself

After both layers of objects have been constructed, the trading model possesses a pattern to interpret data it retrieves from the external database in terms of possible trading decisions. Each Event Node Object that - for the actual data - returns *true* for an *event* to exist, also returns a signal for a decision, to buy, to sell or to remain unchanged in the current position (if any). The trading model then chooses the one decision that occurs most frequently as ENO result, as a final decision to be made in the moment the calculation is performed.

The chart below illustrates the process described above.



N/D indicates a decision result of "nothing done", i.e. current position unchanged.

Dynamic Parameter Optimisation

Since the trading system is designed to learn behaviour patterns, the optimisation of fixed numerical parameters (such as the number of days over which historical volatility is calculated) should be avoided. Not only is such a parameter unlikely to be usable across different markets, it is also very dependent on the actual distribution of prices within the training period. It must therefore be expected that the optimum parameter changes dramatically when market condition change. Using statically optimised parameters leads to inflexible curve-fitting which - in our previous research - has shown not to hold up in subsequent out-of-sample tests.

The concept used by these trading models are "dynamic parameters"; i.e. parameters for which the value is not directly optimised, but for which the model learns to develop rules for calculation. This is done within the Calculation Node Objects. For any calculations that cannot be developed as rules within the trading model, a hard coded time frame of 200 data periods is used as starting point for calculations.

Example: The system may learn to use a moving average of market prices as part of an estimate of near-term market direction. For this moving average, a parameter is needed for the number of days over which the calculation is performed. Rather than optimising this parameter directly (within a given range of possible values), the system learns to calculate the value based on another calculation, e.g. market volatility (calculated by a different Calculation Node Object). The systems retrieves current market volatility reading, and the min / max value for that calculation (over the time span calculated by the other CNO). The last reading is then expressed as ratio within the historical min/max values. If min = 8 and max = 20, a current volatility value of 15 would result in a ratio value of 0.58. The general formula is: Ratio = (Current - Min) / (Max - Min).
This ratio is the current value expressed in percentage of the range. This percentage value is then applied to the min/max range of possible moving average values used by the system. Assuming we define possible moving average parameters between (10;100) periods, the actual moving average parameter used in this calculation would be ((100-10)*0.58) ≈ 52 period moving average. Note that this resulting value changes very time the underlying calculation data (here: volatility reading) changes.

The benefit of dynamic parameter optimisation is that the learning process focuses on developing calculation *rules* rather than optimising parameters. Such rules can be applied to all markets, since they will adjust automatically to new data by re-calculating the actual parameters used for calculation of events.

3.2 Risk and Portfolio Allocation Decisions

Risk and allocation decisions are learned differently by the trading model, as these decisions are not "event-driven", but only depend on a position in a market to exist.

Learning market timing behaviour differs from the other components in the way that we have little knowledge on how buy and sell decision should be made. The system can and should therefore have a large degree of freedom to develop these decision making rules by learning to develop appropriate adaptive behaviour patterns. For risk and allocation decisions, we do however have some knowledge about possible rules (e.g. restricting risk exposure, diversifying portfolios), and we especially have certain constraints that we want to enforce on the trader / the system. Consequently, the trading system can have freedom to decide on allocation strategies and risk exposure within a portfolio, but it must do so by observing the global risk and allocation thresholds.

The learning process for risk and allocation decision is a rule-based learning process that uses a "bottom-up" approach by combining simple rules ("sub-rules") using mathematical and logical operators, resulting in a more complex decision rule, onto which certain risk thresholds are applied by the trading model.
For the GA process, each sub-rule and each operator (mathematical or logical) is recognised as a simple integer value indexing the rule within the risk management object of the trading model.
The GA learning process of the trading model creates a rule which is a formula consisting of data parameters (i.e. calculation results of the sub-rules) and the operators. Although the structure of the formula is limited in its flexibility compared to the market timing decision, the trading model can still create a relatively complex structure of rules very different from the simple components it is based on.

Although we will focus here on the risk and allocation decision to be made as part of a new trading decision (resulting from a market timing signal), the same calculations are repeated at the end of each trading period (mostly end of trading day), in order to adjust existing open positions. These adjustments are particularly important because they make sure that the portfolio will always be maintained within the desired risk thresholds. Our testing has further found that the constant application of risk thresholds is one of the most significant contributing factors to creating a consistent and predictable performance pattern, largely independent of the actual market timing strategy used.

The trading model uses three levels of risk calculation:
- expected risk (calculating an estimation of the risk a certain trading position carries)
- accepted risk (a risk threshold which the trading model defines for each trading position)
- portfolio risk threshold (the overall constraint that the system supervisor or trading manager puts on the entire portfolio and must be observed by the trading model).

The *expected risk* is used by the trading model to make an initial decision on the size of a trading position, similar to a human trader. Through the learning process described below, the model is trained to improve its risk forecasting ability. The *accepted risk* is a threshold that is associated with a given market position, and is calculated by the trading model as part of the learning process. This is not necessarily a "worst-case-scenario" threshold. More likely, the trading model uses risk thresholds to actively manage the size of open positions in a market to create a less volatile equity curve.

Portfolio risk thresholds are defined by the supervisor of the system, the trading manager. The trading manager normally has a clear view of the amount of risk he/she is prepared to accept for a given portfolio. This however is a threshold for the aggregated risk of all positions and the trading model learns to translate this global constraint into position limits for each individual component of the portfolio.

The risk calculation for a portfolio is performed in three steps: price risk - portfolio allocation - portfolio risk.

3.2.1 Price Risk Calculation

Price risk is associated with the market in which a position is taken and is independent of the size of a trading position. It is however closely linked to the type of market timing strategy developed by the trading model, since the expected variance of prices is a function of the time horizon of an investment[2].

The rules which the trading model can learn to use for calculating risk use as input:

- price volatility (in absolute terms over previous n number of days)
- price volatility (in terms of standard deviation)
- previous trading range (price high/low over previous n number of days)
- maximum / minimum price changes (over previous n number of days)
- average/total/minimum/maximum price risk calculations of others components of the portfolio
- price and volatility correlation to other components of the portfolio

The "sub-rules" are combined in a rule structure developed by the GA process that result in a risk estimate for a position in that market. Initially this risk estimate is also the maximum accepted price risk for the position. In trading terms, that price risk is the "Stop-Loss" level at which a position is closed out, because the market forecast, implied by the decision taken by the market timing strategy, is accepted to be wrong.
After the initial position is established, trading models re-calculate price risk estimates every time the trading model is updated with new prices. The however the trading model does enforce to cut the position size, if the estimated price risk exceeds the accepted risk. The accepted risk is only allowed to increase by a limited margin when applied to an existing open position.

3.2.2 Portfolio Allocation Decision

Portfolio allocation decides the percentage share of the overall portfolio value that is allocated to one portfolio component. Portfolio allocation within asset investment strategies is a defined process which could easily be implemented through any GA (or other) learning algorithm. The decision for a trading portfolio is more complex, because the system is not invested into all markets at the same time. The trading model must learn how to allocate a share of the portfolio for a new position to be taken, but also taking into account existing positions and any portfolio limits placed onto the system.

For any given portfolio component the allocated share of the portfolio is calculated by the trading model as

$$A_i = F_i \Big/ \sum_{n=1}^{N} F_n$$

where

A_i ..... allocation for the i^{th} market
N ..... number of markets available to the portfolio
F_i ..... the function expressing the rule used to calculate the allocation

Each rule developed by the GA learning process is based on a combination of sub-rules which are based on the input of:

- price data and time series statistics of the given market
- time series statistics in relation to the other markets in the portfolio (cross-correlation and relative strength)
- relative value of each markets performance within the portfolio

[2] In absolute terms, variance of prices is dependent on the time horizon, not the variance itself. Assuming a normal distribution of price changes, both a short term trader and a long term trader face the same probability of a 1-σ event in the market. In absolute terms (and in percentage of the portfolio) these will be different values.

The result of each calculation is always the re-calculation of allocation for the entire portfolio. This makes sure the trading model not only observes strictly any portfolio constraints, but it also ensures that the trading model can re-balance the entire portfolio after the addition of a new position to the portfolio. Since performance consistency is typically the most important criteria for evaluating trading models, the automatic re-allocation ensures the highest possible level of diversification within the portfolio.

Market Selection is a special case of the portfolio allocation decision, as it is also possible for the system to allocate a share of zero to a market, thus effectively excluding a market from the portfolio. When setting up the adaptive trading model, a number of markets are made available to the system. However, the trading model can in every stage of the calculations decide to reject a certain market either by not creating a buy or sell decision or by allocating a very small share of the portfolio to that market that cannot be traded in the market.

3.2.3 Portfolio Risk Decision

After the trading model estimates the price risk associated with a position and calculates the share of the portfolio allocated to a particular trade, it can calculate the actual quantity to trade, i.e. the size of the trade and of the open position.
In learning to perform this calculation, the trading model uses the calculated price risk, the allocated equity and all input data used by previous calculations to create the decision making rule for the trading quantity.

The decision on the size of each position has shown to be among the most relevant decisions determining the consistency of portfolio performance.

In general terms, the position size is a function of the available (allocated) equity for a given market and the price risk associated with the market (which the trading model both learns to calculate/estimate), i.e. $U_i = f(A_i, P_i, R)$ where U_i is the number of trading units (position size), for the i^{th} market within the portfolio, P_i is the calculated Price Risk, for the i^{th} market, A_i is the calculated allocation for the i^{th} market and R is the portfolio risk accepted by the system expressed as a percentage of total capital accepted to be at risk.

The above function can also be re-written as $R_{actual} = f(A_i, P_i, U_i)$. In other words, the actual percentage of the portfolio at risk (R_{actual}) is a function of a market's price risk, the allocated share of the portfolio and the number of units bought or sold in the market. For the purpose of the learning process, the R_{actual} value should be smaller or equal to the accepted portfolio risk value, R. The learning process of the trading model therefore first finds a percentage portfolio risk value R that is compatible with the global parameters of the portfolio, and then, using both other parameters to the function, A_i, P_i, calculates the actual number of trading units (U_i) that would create the desired exposure.

The calculation of the trading size is the link between the market timing behaviour developed by the system and the required risk management strategy.

For every market timing decision made by the system, it will calculate the associated market risk (per unit price risk) and the associated portfolio allocation.
It is through the variation of the trading size (position size) that the system is performing a trade-off between higher (lower) per unit price risk and smaller (larger) size of the trading position.

Example A:	Proposed LONG position in US T-Bonds at	100.— (Unit Face Value US$ 100,000)
	Estimated (Accepted) Price Risk:	2%
	Allocated Equity to this trade:	US$ 1,000,000.—
	Accepted Portfolio Risk on Position:	1.5%

Calculating Accepted Position Risk in absolute US$ = (1,000,000 * 1.5%) = 15,000
Calculating Accepted Per Unit Risk in absolute US$ = (100,000 * 2%) = 2,000

Accepted size of the trading position = 15,000 / 2,000 = 7.5 units of T-Bonds at face value US$100,000 equalling a US$ 750,000 investment. If contract sizes do not permit the exact trade size as calculated, we mostly round down to the lower possible trading size (lower risk), here, a US$ 700,000 investment.

Through that process, the trading model can translate changes in both the estimated price risk and changes in the portfolio allocation, directly into changes in the position size, maintaining constant portfolio risk exposure, that is restricted by global portfolio risk thresholds.

Example B: Actual Market Price is now at 101.— (up from 101.--)
System calculates price risk to be 3% (up from 2%).
Allocated equity now at US$ 800,000 (due to losses in other markets within the portfolio)
Accepted Portfolio Risk unchanged at 1.5%

Calculating Accepted Position Risk in absolute US$ = (800,000 * 1.5%) = 12,000
Calculating Accepted Per Unit Risk in absolute US$ = (101,000 * 3%) = 3,030

Calculated accepted position size = 12,000 / 3,030 = 3.96 units of T-Bonds (at face value US$ 100,000), rounded up to 4 units, equals accepted position size of US$ 400,000. The system will therefore issue a signal to sell US$ 300,000 worth of T-Bonds in order to adjust the position to changes in the price risk and in the portfolio composition, expressed in the lower portfolio allocation.

Note that this partial liquidation actually returned a profit for this particular position, although the reason for selling this position was entirely due to the overall risk management of the portfolio. This process essentially helps the system to "smooth out the equity curve", especially when the portfolio is properly diversified.

This approach to managing the portfolio risk is in our view an essential component of a consistently developed trading strategy. It removes the uncertainty that is normally associated with profitable open positions; if profits should be kept "running" or positions liquidated in order to ensure that open profits are protected from any more risk.

These examples show the essential calculation of the position risk and position risk adjustment. The actual parameters and actual rules used by the system are also developed in a rule based learning process with input drawn from both price risk calculation and portfolio allocation calculation.

In order to develop consistent performance behaviour, the trading model must have the ability to manage constant risk exposure during changing portfolio composition and market events. This process enables the trading model to consistently balance the market price risk and the portfolio risk, by changing the actual position size it will have in any market. Increased risk per unit traded (price risk) can be matched by a decrease in number of units traded and vice versa. As with the calculation of portfolio allocation, portfolio risk management is always performed over the entire portfolio. Therefore, a change in one portfolio component can be matched by shifting (i.e. reducing) exposure in other markets. This allows the trading model to re-balance the portfolio either when a new market position is to be taken or when the daily mark-to-market process is performed.

4. Performance Measurement and Fitness Target

Performance measurement is an important aspect of a GA based learning process, because the learning process itself is based on the survival of the fittest concept and the selected performance benchmark represents the rating of an individual solution's fitness.

To optimise a trading strategy for consistency of performance, the performance benchmark must measure this consistency. To create a trading model that adapts without human interference, the performance benchmark must also measure the absolute level of performance, relative to the expected return and the accepted risk. Portfolio performance is often measured by some form of risk-adjusted return, such as the Sharpe Ratio or Yield/Drawdown Ratio. For evaluating a trading system's performance in an automatic, self-learning process these measurements are not practical because they do not take into account the time structure of performance (consistency of performance) and do not include measurement of the absolute level of performance.

The target function of the learning process applied in our trading models is based on a user-defined Return Path (RP). This return path is a monthly or quarterly range of expected returns within which the system ideally performs. The fitness of the trading model is measured by the error of tracking this target range, the Return Path Error (RPE). Formally, the RPE is defined as

$$RPE = \sqrt{\sum_{n=0}^{N-1} e_n^2} \Big/ N ,$$

where:
N … number of periods (either calendar quarters or calendar months),
n is the n^{th} calendar period (indexed between 0 and N-1)
e_n is the actual tracking error for the n^{th} period.

e_n is defined as follows: if ($r_n > RP^+$): $e_n = (r_n - RP^+)W$, if ($r_n < RP^-$): $e_n = (r_n - RP^-)$, where r_n is the calculated actual percentage return of the portfolio for the n^{th} period, RP^+ is the upper limit of the return path target range, RP^- is the lower limit of the return path target range and W is a weighting applied to smooth the effect of *upside* errors of the portfolio (typically $0.3 \le W \le 1.0$; this model uses an error weighting of 0.4).

The GA process seeks to minimise the RPE, which ideally equals zero, when the system performs completely within the desired return path.

The advantage of using RPE as performance benchmark is that it emphasises and measures the consistency of performance in that it matches the user's expectation on return with any risk-thresholds attached to the portfolio. If the return expected from the model is not compatible with the risk constraints placed upon the portfolio, this discrepancy can then be already detected during the learning process. Either the portfolio constraints or the performance expectations will then have to be adjusted.

5. Cross-Validation and Learning Process Setup

Cross-Validation is an important tool in the evaluation of machine learning processes, dividing the available time series into periods for training and periods for testing, during which the performance of the trained parameters is validated.

An adaptive process uses a continuos training and application process to learn behaviour and apply this behaviour on new data. In the Foreign Exchange portfolio demonstrated below, the following setup for training/evaluation periods is being used:

	Training (Learning)		**Testing (Application)**	
	From	*To*	*From*	*To*
Period A	09-05-1990	04-06-1993	07-06-1993	08-11-1994
Period B	09-05-1990	08-11-1994	09-11-1994	12-04-1996
Period C	05-09-1991	12-04-1996	15-04-1996	17-09-1997

The choice of 3 adaptation periods is largely arbitrary and depends mostly on the available machine time for performing the testing. A higher frequency of re-training and adaptation would create a more continuos adaptive behaviour. We have opted for 3 periods to keep the training of the portfolio within the parameters of our available computer resources.

Each test period uses a defined training period for learning. The learning algorithm starts with a random initialisation (i.e. a state of no knowledge and random behaviour). An initial training period of 800 trading days (9 May 1990 to 4 June 1993) is assigned to the first testing period. During this period the system learns to develop basic rules and already eliminates a large number of consistently unsuccessful behaviour patterns. 800 trading days as an initial period represent about 1/3 of the database available. After that, a maximum amount of

1200 trading days is allowed for the training period to create similar training environments for each testing phase.

Top Fitness / Closest Fitness Behaviour Patterns

After the training process, the trading model selects one single rule system to be applied to new data. Typically this is the rule set which had resulted in the optimum theoretical performance during the training phase. At the beginning of the application phase of each new period, the trading model will adapt its behaviour according to the rules it has learned during the training process. Since every learning process is an optimisation process, the system always carries the risk of overoptimisation during the learning process.

Using overoptimised behaviour patterns on new data is very likely to result in undesirable, negative performance. We have therefore developed a concept of not using the optimised rule set for the actual trading period ("top-fitness" rule set), but to select one rule set, which is likely to be more robust in its real-time performance then a highly optimised behaviour, the "closest-fitness" rule set.

To calculate the "closest-fitness rule set", the trading model uses an internal threshold to find a range of behaviour rules, which resulted in acceptable performance during the training period, including the best strategy, i.e. the top-fitness rule set. Within this group of rule sets, the system then tries to find a smaller group with similar performance results. If such a group is found, the system selects the best rule set of this group to adapt to, for the new period. This rule set is referred to as the closest-fitness rule set. It may be the case that the selected closest-fitness rules set and the top-fitness rule set are identical, but more often this is not the case.

Because this closest-fitness rule sets is not as highly optimised as the top-fitness behaviour, we have found a very significant increase in consistency of performance, when comparing the training results with the application periods.

Continuos Strategy Evaluation

During a real-time application of the trading model, at the end of each actual period (when the system prepares to adapt new behaviour patterns), the trading model already has generated a stream of trading decisions, which have resulted in a profit or loss, and the system may also have open positions in any of the markets of the portfolio. Although the division of the database into several training/application periods is necessary to create an adaptive learning process, it does not correctly reflect how the system would be applied in a real-time environment.

To replicate real-time behaviour, the trading model has the ability to dynamically adapt new behaviour while keeping all existing open positions and existing accounting values. This creates a continuous performance measurement and allows to measure the effect, switches in the behaviour patterns would have on existing market positions.

6. Results - Leveraged Foreign Exchange Portfolio

6.1 Overview

We present here a simulated foreign exchange portfolio which is designed to incorporate market timing, market selection and risk management into one integrated trading model.

Portfolio Performance Measures

Fitness Value	Reciprocal value of the Return Path Error (the reciprocal value is used to use an optimisation process that maximises, where the RPE should be minimised)
Yield	Annualised compound yield of return
YieldDD	Ratio of Yield/Maximum Drawdown ever occurred in the trading model (measured on a daily basis)
Profit Months	Percentage of Months Profitable
Profit Quarters	Percentage of Calendar Quarters Profitable (more important here because fitness value is based on quarterly return path optimisation)
No. Trades	Number of Trades during the relevant period (includes transactions which resulted in partial close-out of existing position due to risk management adjustment)

Measuring Adaptive Behaviour Predictability

Performance consistency can be measured using the Return Path Error or other risk adjusted return calculations. Performance predictability is the ability to forecast the actual result based on the performance exhibited by the trading model during the training process. Ideally, the real-time performance would be very close to the performance during the training process.

We have so far not established a defined process to measure behaviour / performance predictability, but it is possible to measure the predictive value of training performance but comparing several performance benchmarks between the training and evaluation periods. This will be shown as the ratio between the actual (hold-out) result to the training result. This analysis can also be used to decide on parameters for the learning process and to improve fitness calculation and portfolio selection.

Portfolio Specification

Portfolio Base Currency	US Dollar, Profits/Losses are converted to US$ at prevailing exchange rates as they are realised.
Individual Market Constraints	No internal restrictions on individual position size. Each market could be allocated any position size between zero and 100% of the available trading capital.
Global Portfolio Constraints	Maximum portfolio exposure must not exceed 3 times current portfolio value (including open positions evaluated at current market prices).
Transaction Costs	Each transaction is assumed to carry 0.1% of the price as transaction costs[3]. Swap costs/gains have not been included.
Return Path Specification	Quarterly Return Path of 3%-15%.
Drawdown Limit	A drawdown of 30% is considered a total loss on the portfolio. In other words, if at any time the trading model would lose 30% from the last equity peak, trading for this system is to be stopped.

6.2 Portfolio Results

Markets

US Dollar Rates: GBP/USD, AUD/USD, USD/CHF, USD/CAD
Cross Rates: GBP/DEM, DEM/JPY, DEM/CHF

[3] Typical Forex transaction costs may be around ¼ of this value, however, this assumption also allows for *slippage*, that is, an actual execution price worse than the desired trading price (e.g. due to volatile market conditions).

Data frequency: Daily Data (Open/High/Low/Close based on NY Trading Close)

6.2.1 Portfolio Performance Results

(1) Overview: Using the optimised top-fitness behaviour rule set during the application of learned behaviour

	Period Indx	Fitness	Yield	YieldDD	Profit Months	Profit Quarters	No. Trades
Learning	A	245.7437	14.19%	2.95	72.97%	100.00%	81
Application	A	39.8272	3.26%	0.59	70.59%	66.67%	29
Learning	B	112.0823	12.62%	1.73	72.22%	88.89%	56
Application	B	37.1702	5.01%	0.92	52.94%	50.00%	25
Learning	C	60.887	7.96%	1.00	67.27%	84.21%	106
Application	C	22.1744	-4.26%	-	41.18%	40.00%	36
Continuos Application	**(All)**	**31.276**	**0.51%**	**0.04**	**52.94%**	**47.06%**	**94**

(2) Overview: Using the more robust "closest-fitness" behaviour rule set during the application of learned behaviour

	Period Indx	Fitness	Yield	YieldDD	Profit Months	Profit Quarters	No. Trades
Learning	A	210.9709	14.52%	2.95	70.27%	100.00%	91
Application	A	40.2518	3.64%	0.64	70.59%	66.67%	31
Learning	B	112.0745	12.64%	1.74	72.22%	88.89%	56
Application	B	38.6552	5.69%	1.20	52.94%	50.00%	22
Learning	C	58.2502	7.54%	0.99	63.64%	73.68%	86
Application	C	43.1602	6.66%	1.00	52.94%	60.00%	18
Continuos Application	**(All)**	**42.119**	**5.11%**	**0.76**	**58.82%**	**64.71%**	**86**

Above tables show that the system had been able to develop profitable behaviour under the condition of realistic transaction costs. It can also be seen that using the "closest-fitness" rule set the out-of-sample results by far exceed the results of the highly optimised top-fitness behaviour rule set, while during the training period, the top-fitness parameters exhibited far better hypothetical performance.

6.2.2 Performance Consistency and Predictability Measurement

(1) Top-Fitness Behaviour Rule Set: Performance Consistency across Learning/Application Data Sets.

The highly optimised top-fitness behaviour rule set exhibits far less successful behaviour during the application data sets compared to the training data.
The average ratio of training result vs. application result is 0.48.

Fitness		**Period A**	**Period B**	**Period C**	**Average**
	Learning	245.74	112.08	60.89	139.57
	Application	39.83	37.17	22.17	33.06
	Ratio	*0.16*	*0.33*	*0.36*	*0.29*

Yield		**Period A**	**Period B**	**Period C**	**Average**

	Learning	14.19%	12.62%	7.96%	0.12
	Application	3.26%	5.01%	-4.26%	0.01
	Ratio	*0.23*	*0.40*	*N/A*	*0.31*

Profitable		**Period A**	**Period B**	**Period C**	**Average**
Months	Learning	72.97%	72.22%	67.27%	0.71
	Application	70.59%	52.94%	41.18%	0.55
	Ratio	*0.97*	*0.73*	*0.61*	*0.77*

Profitable		**Period A**	**Period B**	**Period C**	**Average**
Quarters	Learning	100.00%	88.89%	84.21%	0.91
	Application	66.67%	50.00%	40.00%	0.52
	Ratio	*0.67*	*0.56*	*0.48*	*0.57*

(2) Closest-Fitness Behaviour Rule Set: Performance Consistency across Learning/Application Data Sets.

Using the more robust closest-fitness behaviour rule set results in a more predictable performance of the trading model during the application data sets, compared to the training data.
The average ratio of training result vs. application result is 0.62, as compared to only 0.48 with the top-fitness rule sets.

Fitness		**Period A**	**Period B**	**Period C**	**Average**
	Learning	210.97	112.07	58.25	127.10
	Application	40.25	38.66	43.16	40.69
	Ratio	*0.19*	*0.34*	*0.74*	*0.43*

Yield		**Period A**	**Period B**	**Period C**	**Average**
	Learning	14.52%	12.64%	7.54%	0.12
	Application	3.64%	5.69%	6.66%	0.05
	Ratio	*0.25*	*0.45*	*0.88*	*0.53*

Profitable		**Period A**	**Period B**	**Period C**	**Average**
Months	Learning	70.27%	72.22%	63.64%	0.69
	Application	70.59%	52.94%	52.94%	0.59
	Ratio	*1.00*	*0.73*	*0.83*	*0.86*

Profitable		**Period A**	**Period B**	**Period C**	**Average**
Quarters	Learning	100.00%	88.89%	73.68%	0.88
	Application	66.67%	50.00%	60.00%	0.59
	Ratio	*0.67*	*0.56*	*0.81*	*0.68*

6.3 Portfolio Results - Conclusion

As shown in above tables, an adaptive trading model can be designed to focus on performance consistency, and this consistency can also be measured.
Especially important is the process of selecting a set of behaviour rules to provide a filter for over-optimised training sets. Since optimisation is an integral part of the learning process, it is important, as in the tables shown, to measure not only the result of the trading performance, but also the performance of the learning process itself, by estimating the predictive value of training set performance.

7. Conclusion

In this article we have demonstrated that intelligent trading models can be designed and tested in such a way that all functions of a human portfolio manager can be performed by an adaptive, self-learning process. Although such an approach lacks the flexibility inherent in human behaviour, the consistency of behaviour and consistency of performance are clearly more of concern to investors and trading managers.

Based on our existing research we are convinced that intelligent agents in portfolio management can contribute to a more effective and more predictable management performance.

We are currently beta testing a distributed, parallel processing version of the learning process which is scalable from a single processor system to a network of any number of workstations. This allows a larger number of simulations to increase the frequency of re-training and adaptation, eventually leading to a continuos adaptive process. Further research is conducted in the application of Genetic Programming systems, which use the same evolutionary programming concept (“survival of the fittest”), but do not use the same encoding of information in binary strings as Genetic Algorithms.