Adaptive Portfolio Trading Strategies for Foreign Exchange Portfolios

by
Arthur Rabatin, Rabatin Investment Technology Ltd
research@rabatin.com

http://www.rabatin.com

First Published by BNP Global Markets Research, December 1997

Abstract
Adaptive Portfolio Trading (APT) involves the development of self-learning, self-adapting intelligent trading models for portfolios of financial instruments. The framework uses *Genetic Algorithms* (GA), a parallel processing algorithm, to develop these trading models. Within this framework, we are able to develop models that perform successfully all functions a portfolio manager has to perform, while at the same time strictly observing risk thresholds or other defined constraints. This project attempts to use a new approach to systematic portfolio trading. It integrates *all* aspects of investment decisions, *market timing, portfolio allocation, risk management*, into one decision-making model, that is trained to develop trading behaviour that achieves consistent and predictable performance. Consistent performance is not the result of a single, optimised algorithm, or a single *best practice* rule. To create predictable performance, robust behaviour patterns must be developed, which are dependent on the details of the trading models implementation and learning algorithms. The design and results of such a trading model will be described using a US$-based portfolio that learns to trade a portfolio of different exchange rates.

1. Introduction
Genetic Algorithms are parallel processing algorithms that are used in a wide array of applications, like optimisation, scheduling and decision-making simulation. GAs have been actively researched since the early 1970's although their roots could be traced as early as 1940. Although the basic algorithms of GAs are comparatively simple, their *domain-independent flexibility*[1], combined with other artificial intelligence (AI) techniques (such as rule-based systems or *classifier systems*[2]) allows the design of flexible and powerful adaptive models.

GAs function in the way that an array of possible solutions to a problem (say, an array of objects that contain trading models) is evaluated and each solution is assigned a *fitness value*, i.e. a measurement of success or failure[3]. The performance benchmark used within the APT framework, the *Return Path Error* (RPE), is described below. Each of the possible solutions then competes against each other as individuals that form a *population*. This evolutionary concept is known as *survival of the fittest*. After the array of individual solutions, an entire population, has been evaluated (i.e. a fitness value has been assigned to each of them), a new generation of solutions is created by combining existing individual solutions. Here the term *genetic* refers to the method the algorithm is employing to encode information. Each possible solution (i.e. trading model) is encoded into a string of '0' and '1', resembling the structure of the genetic information as it is encoded in human cells[4].

New generations of this population are created by randomly cutting existing strings and combining these into new strings constituting the individuals of a new generation's population. By simulating the concept of survival of the fittest, individual solutions that are assigned higher fitness values (e.g. trading models with a higher Sharpe Ratio[5]) are being made more likely to be copied into the new generations, compared to lower fitness solutions. In this way each new generation of solutions consists of a combination of even fitter individuals.

The advantages of GAs for the design of complex decision-making models could be summarised as follows:

- The algorithm itself does not deal with the actual input data but with binary representations. This allows to put virtually any kind of input data into the control of the GA, as long as an *encoding*/*decoding* algorithm can be defined.
- The GA itself is a domain-independent algorithm, which can be applied to any kind of problem domain. There is therefore no restriction on the type of trading model which can be put under GA's control.
- GAs can deal simultaneously with a large number of input data and can create a large number of output results, allowing the analysis of complex, multi-dimensional problem matrices.
- The flexibility of GAs allows easy combination with other AI techniques to create *hybrid* models specifically suitable for a given task (like classifier systems or optimisation of neural network design).
- GAs allow implementation of constraints in various ways. This is important as the training of trading models must observe strict risk management and portfolio management thresholds.
- GAs are well-suited for parallel and distributed processing, as each population may consist of 100 or more individual models which can be evaluated simultaneously in parallel processes[6].

---

[1] Algorithms are *domain-independent* when the algorithm itself does not have any knowledge about the structure of the problem or the possible solution. For Genetic Algorithms, as long as a fitness value is available, the function of the GA is independent of the structure of the underlying problem.

[2] *Classifier Systems* (CS) are a set of rules that perform certain actions when the rule conditions are met. CS evaluate arrays of events occurring in the environment (e.g. financial markets) and learn to map certain patterns to certain kind of actions. Classifier systems are well-suited for the training of a decision-making process.

[3] Risk-adjusted return could be taken as such a measure.

[4] Generally, the encoding can be done over any type of finite alphabet. Using '0' and '1' is an efficient way because numerical values can be easily translated from decimal to binary form and vice versa.

[5] Sharpe Ratio (named after W. Sharpe) is a common measurement for risk-adjusted return. It measures the ratio between the annualised yield (adjusted by a risk-free rate) and the annualised standard deviation of monthly returns.

[6] A *parallel process*, in computing terms, allows the use of multiple processors to solve a particular problem. This can only be done if the underlying algorithm-as the GA does-allows to *split* the problem into separate components, which can be evaluated independently. This particular implementation of

2. Application of Artificial Intelligence Concepts in Trading and Design Principles of the APT Framework

The application of AI concepts in financial trading is mostly associated with time series forecasting. However successful such applications may be, translating such forecasting results into a portfolio trading strategy requires a number of decisions to be taken, such as portfolio allocation, risk management, that significantly influence the trading performance, but can in no way be derived from the market forecast. Unless every market forecast is extremely accurate[7], even small changes in portfolio allocation or risk management can turn a theoretically profitable forecasting strategy into an unpredictable distribution of profits and losses[8] (see also Toulson and Toulson (1997)).

Multi-currency and multi-instrument portfolio trading is different from theoretically trading a single market because the effect of portfolio allocation decisions, shifts in exchange rates, shifts in financing costs influence the decision made for an individual instrument. In a diversified portfolio, constraints are defined for the entire portfolio, but affect the trading decision in each individual instrument. Therefore, a new approach is required for systematic trading models. It is necessary to model the entire daily *decision-making process* of a portfolio manager, applying the same conditions and restrictions in the system's learning process as applied to the real-time execution of such a strategy. As it will be shown in this article, the flexibility of Genetic Algorithms, combined with concepts of classifier systems and rule-based systems, are especially suitable for this task.

The design of trading models developed within the APT system is based on the following requirements with the aim being not to maximise predictability of market prices, but to maximise consistency and predictability of trading performance. Namely,

- integrating all aspects of the trading/investment decision into one complete decision-making model incorporating:
  - Market Selection Decision
  - Portfolio Allocation Decision
  - Buy/Sell Decision
  - Market Price Risk Analysis
  - Portfolio Risk Analysis and Portfolio Risk Management Decision
- application of real-time constraints, such as user-defined risk thresholds, allocation restrictions, defined by the trading manager, throughout the entire training process,
- designing trading models as distributable objects that can be executed across a network and allowing for performance to be replicated on several locations (but performing the training process centrally),
- creating adaptive models that can learn and adapt without human interference.

3. Adaptive Trading Models Applied to a Leveraged Foreign Exchange Trading Portfolio

Rather than describing the theoretical aspects of the underlying concepts, this article will describe the design and training process of a leveraged foreign exchange portfolio that has been created and trained. The model results have been obtained with the current *beta version* of the training application, which indicate profitable models with acceptable risk/return ratios.

3.1 Portfolio Performance Measurement: Return Path Error (RPE) Optimisation

For an adaptive learning process, performance measurement is especially important, because the learning process 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.

During an automated learning process the performance benchmark is used by the strategy model to evaluate the results of the learning process. Choosing a precise performance benchmark is necessary to avoid the risk of creating *overoptimised* rule sets during the learning period, which are likely to fail when applied to new data. For GAs, a performance measurement or a fitness value is necessary to implement the survival of the fittest strategy.

The performance benchmark used in the APT model is based on a user-defined *Return Path (RP)*[9]. The RP is either the quarterly or monthly range of expected percentage returns which constitute the optimum level of performance. The benchmark used in the APT system is the deviation of the actual returns from this path, i.e. by how much the actual monthly/quarterly return exceeds these RP limits to the upside or downside. This results in the monthly/quarterly Return Path Error (RPE) value, which the learning process seeks to minimise (an RPE of zero indicates performance is completely within the desired range)[10].

GA in the APT framework also allows the training of trading models in a *distributed process*, which is a parallel process that can be executed on more than one system, i.e. across a network of workstations.

[7] According to Sharpe, attempts to time the market are not likely to produce incremental returns of more than four percent per year over the long run. Moreover, unless a manager can predict whether the market will be good or bad each year with considerable accuracy (e.g. be right at least seven times out of ten), he should probably avoid attempts to time the market altogether (Bauer (1994)).

[8] According to some authors (influenced by the development of modern portfolio theory), the allocation and risk decision is far more relevant than the attempt to predict prices, a decision that could as well be made randomly.

[9] Formally, the RPE is defined as $RPE = \sqrt{\sum_{n=0}^{N-1} e_n^2} \Big/ N$, where N is the number of periods (either calendar quarters or calendar months), n is the n^{th} calendar period (indexed between 0 and N-1) and e_n is the actual tracking error for the n^{th} period. This last term 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).

[10] As far as we are aware of, this is indeed a new concept for evaluating portfolio performance.

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.

3.2 Database Setup and Portfolio Specification

The following currency pairs have been evaluated: GBP/USD, USD/CHF, USD/CAD, AUD/USD and the cross rates of GBP/DEM, DEM/JPY and DEM/CHF. The selection of currency pairs reflected the requirement to create an automated trading approach for a portfolio of markets which exhibit different volatility and trend patterns. The purpose is to induce *diversification* to trading strategies which rely on markets exhibiting longer term trends, such as USD/DEM and USD/JPY.

The database used in the portfolio consists of daily open/high/low/close data (closing prices as of 6:00 PM NY Time) for the period from 9 May 1990 to 17 September 1997. Additionally, historical US$ exchange rates for various currencies are available to perform a daily *mark-to-market*[11] of the portfolio during both the testing and evaluation periods.

In the following table we present some additional details regarding transaction costs, individual and global portfolio constraints.

Table 1: 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 porfolio 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[12]. 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.

3.3 Adaptive Trading Model: Learning Process Design

This section describes how the actual learning process is designed. A difference is being made between the learning method for the buy/sell decision and for other aspects of the trading decision. The system has more freedom in finding decision criteria for buying or selling than it has for creating risk management or portfolio management rules. This is partly due to the fact that the user of such a system will wish to have greater control over the risk and portfolio management system used (e.g. to implement specific constraints), than it is necessary on the buy/sell decision.

The difference in terms of *expert knowledge* between these two areas is that regarding market forecasting, we do not know if we are able to correctly predict the direction of the market, whereas in risk management, we know certain rules which can be applied (like restriction on the overall portfolio risk), but we do not know *which* of the rules (i.e. which parameters) or combination of rules, we should best employ for a specific trading or investment strategy. In microeconomic terms, the buy/sell decision is a decision under risk, whereas the risk management decisions are decisions under uncertainty.

3.3.1 Learning Market Timing (Buy/Sell) Decisions

Rather than defining a market forecast and deciding to buy or sell a particular market, the APT system uses an *event-driven* strategy to learn which market (environment) conditions should lead to which kind of behaviour. *Behaviour* is defined as an action to buy a particular instrument, sell that instrument or do nothing, i.e. not changing the current state[13]. The market timing model is based on the system learning to map patterns in environment data to trading decisions. This is done by creating layers of objects which retrieve information from the environment (market prices or other data), apply mathematical, statistical and logical operations, and return an array of logical "true/false" values, which are interpreted by the system to create a decision to buy or sell a certain market instrument, or to do nothing. The purpose of this process is to build an evolving pattern of events in the market environment, which the system can interpret to develop its trading behaviour. More specifically, the trading model learns to analyse relationships within data using *calculation node objects (CNOs)* and then translates these numerical results into a decision-making strategy by mapping resulting numerical patterns to a decision pattern using *event node objects (ENOs).*

Calculation node objects principally retrieve input data, apply a mathematical, logical operator or a statistical function and calculate output data. The GA-based learning algorithm (i.e. the optimisation process) selects which input data are used and which mathematical, statistical or logical operator is applied to calculate the output data.

Input data can be either environment data (i.e. data retrieved from an external database, like market prices), or output data from other CNOs. The n^{th} CNO can use the output data of (CNO[0]...CNO[n-1]), for n > 0. The calculations of the CNOs are aware of the time series nature of data, i.e. calculations need not only use latest data, but can go back into the history of the price series. The output data are the result of the calculation, where results of logical operator functions are represented through integer values (0 (false), 1 (true)). Every time the trading model is evaluated with new data, the calculation function of these CNOs is called by the program to calculate each CNO's output data.

---

[11] *Mark-To-Market* describes the process of evaluating open positions and available cash balances using current market prices and current exchange rates. In this portfolio, the result is the US$-value of the portfolio, including losses or gains on any open position.

[12] 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).

[13] Note that today most financial instruments can also be sold *short*.

A second layer of objects, ENOs, translates the numerical result of CNOs into a signal that represents a possible trading decision to buy, sell or to keep the existing position unchanged. Finding mapping modes to translate numerical results into decision signals is the responsibility of the optimisation process. The structure of these mapping modes is important because they make sure the conditions for buy and sell decisions are defined symmetrically: for every rule to buy a market must also be an intepretation when to sell a market using this rule. Otherwise, the system could easily develop a bias towards a market direction, in other words be over-optimised based on a certain *market direction* during training, rather than optimise on the decision *rules*[14].

As a last step, the array of ENOs is evaluated by finding a pattern to select ENOs, of which then one action, that appears most often in that pattern, is retrieved as a *final* decision signal. The result of the entire calculation is then either a buy, a sell or a do nothing decision. The advantage of this setup from an AI technology point of view is that :

- The flexible arrays of CNOs and ENOs overcome the restrictions of fixed length Genetic Algorithms.
- Similar to Koza's *Genetic Programming* concept, a complex structure can be represented by combining *simple* operators and creating a linked list of calculation objects (Koza (1992)).

The selection of the actual decision, which results from the calculation, is similar to the way a classifier system selects decisions. However, here the competition between the nodes is performed by the GA system and only evaluated by the fitness function, rather than within the classifier system itself.

3.3.2 Learning Portfolio Allocation, Market Selection and Risk Management

Of the components of the trading decision, the market timing model is implemented differently than risk analysis, portfolio allocation and risk management. Price risk analysis, portfolio allocation and portfolio risk management are implemented as a rule-based learning process. Risk-related decisions face stricter constraints than the market-timing model. Rather than leaving the system with any freedom to chose its own risk behaviour, the system is able to chose from different given rules, and a limited range of mathematical and statistical operators to combine these rules. These rules mainly define risk and allocation strategies, which the trading model is free to combine within the overall portfolio constraints that the trading manager has assigned.

3.3.2.1 Price Risk Analysis

Price Risk defines the anticipated risk in the price change and the worst-case price level at which an open position will be squared. In other words, this analysis represents the level of risk the trading model is prepared to take once a buy/sell decision has been initiated. To calculate this risk, a number of input variables are available to the trading model, such as:

- Price Volatility (historical average price changes over N-days)
- Price Volatility (Standard Deviation of price changes)
- Trading Range High/Low (over N-days)
- Maximum Price Changes (over N-periods)
- Price Risk calculations of other instruments within the portfolio
- Correlation of volatility of this instrument to other instruments within the portfolio.

All input data are calculated over time frames, which are part of the optimisation process. These data are fed into the learning algorithm, which is trained to decide about the input parameters for these rules and the combination of rules using mathematical and logical operators.

Price risk is often estimated as the standard deviation of prices. The trading model may also create very different estimates of risk. The trading model is free to develop expressions using each data input in multiple ways. Although essentially any estimate of market risk must use previous price behaviour as input in any form, the trading model can draw upon a number of data to improve the risk estimate. Because any estimate of market risk also depends on the investment time horizon, it is important that these elements are evaluated simultaneously with the type of trading strategy chosen by the system.

The result of these calculations is a value that represents the anticipated price risk that is accepted as an exit criterion in a worst-case scenario. Such a worst-case exit criterion is a *Stop/Loss* level. Stop/Loss levels are often neglected in market forecasting models, because they are seen as not relevant to the success or failure of the forecast. This approach, however, cannot be applied to portfolio trading. The Stop/Loss level is a risk threshold at which the original forecast is accepted to be wrong and the taken position must be corrected. If this is to be neglected, then the portfolio is actually assuming unlimited risk. Using the standard deviation as a proxy for price risk instead does not serve the purpose because the actual distribution of price changes can, especially within shorter time frames, not be approximated correctly by a bell-shaped normal distribution.

Evaluation of these rules is repeated every time the trading model receives new data. This enables the model to adjust its risk calculation to changes in the market environment. The only constraint that is put onto this system within the portfolio described here is that the system is never allowed to increase the accepted price risk beyond the initially established value.

3.3.2.2 Portfolio Allocation and Market Selection

Portfolio Allocation is based on rules which input data for each market relative to other markets within the portfolio. During the learning process, the model learns rules as to how the portfolio should be allocated to each market[15], using as input data:

- Price data (and calculations applied to these data, such as market volatility)
- Calculated data (cross-correlation of markets and relative values (similar to stock market "beta"))
- Relative value of each market's previous performance within the portfolio

The calculations applied to one price data are for portfolio allocation purposes always interpreted relative to the entire portfolio. Let us assume historical market volatility was the only data input the system decided to use as an allocation criteria. The

[14] For many financial instruments (especially foreign exchange), "buy" or "sell" is no different type of transaction. The focus of trading model behaviour in the APT framework, is *when is a new position initiated*, and *when will the trading model exit from this position*. Literature dealing with asset investing strategies often makes the assumption that "buying" equals "investing" and "selling" equals investment liquidation. This, though, is not necessarily the case with most financial instruments available today.

[15] The system always calculates an allocation for a market with respect to the entire portfolio. This may, for instance, result in one existing position being reduced because another position is being entered into. This allows to continuously balance the portfolio to keep the highest possible level of diversification, which is in line with the target to develop a consistent trading performance, based on the defined Return Path target. As a restriction, the currently implemented trading models are not allowed to increase the risk of an open position, only to decrease it.

system would then calculate historical volatility for each market in the portfolio, and calculate the share each market has in the total sum of the portfolio. This would translate into a percentage value which the system can use to calculate allocation for every market available to the portfolio, thus possibly re-allocating the portfolio to achieve a more stable portfolio (See also Appendices I and II).

Using as input, the trading model's own historical performance on a given market within the portfolio greatly improves the system's ability to simulate a human decision-making process. Only by using an integrated framework for creating trading decision models, these data can actually be used as an input into the learning process. By applying mathematical and logical operators for the combination of rules, we eliminate having the actual allocation decision solely dependent upon one single rule. The result of these calculations is a percentage value that for each market is being made available within the portfolio. This share of the portfolio is calculated every time the trading model is evaluated with new data (e.g. after each trading day). This allows the system to dynamically adjust the portfolio to new market conditions (should, say, relative performance, relative volatility or cross-correlation change).

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 is not tradable.

3.3.2.3 Portfolio Risk Management

The Portfolio Risk Management is a *high level* analysis, as it uses the output generated by all other calculations to derive the value of the portfolio at risk for a given position. As risk is always the result of open positions in the market, it is dependent on both the anticipated *price risk* and *the amount of equity allocated* to a position, and uses calculations from the previous rule sets as input. Portfolio risk management is generally concerned with measuring risk after risk has been taken. Monitoring risk has indeed become an essential consideration nowadays. In the context of a trading strategy, however, risk management can be far more precise, because the trading model can simultaneously analyse all input variables to the risk calculation, as well the rules used for deciding on risk exposure, before a decision is actually taken.

Within this system, the focus of risk management is therefore not the *ex-post* measurement of risk, but the *ex-ante* decision on how much *risk must be taken*, strictly observing any limits that are put on the system by the trading manager[16]. Input data for these rules include:

- anticipated per-unit price risk of the particular market,
- portfolio allocation decided by the trading model,
- portfolio or risk constraints defined externally,
- input data used by both the price risk and the portfolio allocation calculation rule set.

Like other rule-based learning algorithms, calculation rules use input data and the model applies mathematical and logical operators to combine these rules for the calculation of a single value. The result of portfolio risk calculations is therefore the precise position size a trading position should have if a trade is to be entered into, expressed in terms of units as specified for each financial instrument.

The most important function of the risk management component is to balance the required risk that has to be taken with any trading decision, with the desired exposure. Because the trading model can take both price risk, as well as portfolio allocation into account, the system can exactly evaluate if a certain type of trading decision is compatible with its aim to achieve consistent performance (see also Appendices I and II). Since the calculation of the correct position size is repeated every time the model receives new input data, the model not only calculates the position size for new transactions, but also adjusts the position size for any existing open positions, should either portfolio allocation or risk management rules demand such adjustment. As external portfolio constraints are part of the portfolio risk calculation, the resulting value will always be within those constraints. The effect of such constraints (e.g. relative to the anticipated performance expressed in the Return Path definition) can therefore be precisely measured during the entire learning process.

3.4 Learning Process: Setup

3.4.1 Cross-Validation and Adaptive Systems

Cross Validation is the method of dividing a time series into a *training* set and a *test* set (or several of those), whereby the system's learning process is performed on the training set alone. The resulting parameters are then applied to the test data, which are new to the system. This should give an indication of how successful the application of such rules will be in the future.

Adaptive systems constantly use previous data as training sets to apply the resulting, optimised parameter sets to new data. The frequency of such adjustments depends on the problem domain and the decision has to be made by the system supervisor[17]. In terms of measuring the learning success, adaptive systems offer the advantage that the learning process itself can be tested and more data are available to compare training with evaluation results.

The trading models of the APT system implement adaptive behaviour in two ways:

- no use of static parameters to rules and no use of static expert knowledge rules except for implementing desired constraints (like portfolio risk limitations). All actual parameters passed on to decision-making rules are being made dependent on environment data (e.g. market prices or accounting data).
- Periodic re-learning using previous data, from both the environment and previous performance results.

[16] The APT framework allows implementation of any kind of constraints. Typical constraints are a restriction of the relative size of any position within the portfolio (e.g. to enforce diversification). Other constraints may restricts trading size in certain option contracts due to limited market liquidity. The FX trading model demonstrated here does not use any restriction for individual positions, but uses only a global constraint on the entire portfolio, as defined by the maximum leverage of 3 times the marked-to-market portfolio value.

[17] See Holland (1975): "It follows (…) that an adaptive plan cannot be considered good simply because it will eventually produce fit structures for the environment confronting it; it must do so in a reasonable time span. What a "reasonable time span" is depends strongly on the environments (problems) under consideration (…)" (see table 1, Portfolio Specification).

The structure of the trading model to use dynamically calculated parameters (rather than static, optimised) is necessary to create a system that learns behaviour rules, rather than simply trying to repeat previous behaviour patterns. By focusing on learning behaviour rules, the system behaviour and its success become more consistent and robust, even in a changing environment. It is the second aspect, the re-training, which is the core of a Genetic Algorithm based learning process.

In the previous paragraphs, we have shown how the trading model is designed to process input data to create dynamic decision-making rules. The following paragraphs will describe how the periodic re-learning of the trading model allows the system to adapt to new environments by including new data in the learning process. All data refer to the configuration of the learning process that is used in the Forex trading model, described below.

3.4.2 FX Trading Strategy: Data Setup

We have decided to adjust the strategy approximately once every year and half for the period from 7 June 1993 to 17 September 1997, resulting in three test periods of approximately 370 trading days (see the following table).

Table 2: Data

	Training (Learning)		**Testing (Application)**	
	From	*To*	*From*	*To*
Period A	9-5-1990	4-6-1993	7-6-1993	8-11-1994
Period B	9-5-1990	8-11-1994	9-11-1994	12-4-1996
Period C	5-9-1991	12-4-1996	15-4-1996	17-9-1997

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.

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. Choosing a suitable performance benchmark is critical to the success of the adaptive process, because this benchmark decides on the rule set that is selected. Using benchmarks (fitness values) that easily lead to over-optimisation will lead to drastically reduced performance during the testing 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, 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.

After the training is completed, the trading model uses an internal threshold to find a number 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. The benefit of consistency increases even more when more diverse markets are used in the trading portfolio.

3.4.4 FX Trading Strategy: Creating a Continuous, Adaptive Trading Strategy

The division of the database into several training/application periods is necessary to create an adaptive learning process, but still does not correctly reflect how the system would be applied in a real-time environment. 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. Measuring each training/application period independently does therefore not correctly reflect the 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. We regard the ability to create continuous, adaptive performance simulations as one of the main aspects of the APT development framework.

3.5 Portfolio Performance Results: Overview

The portfolio performance measures are presented in Table 3.

Table 3: FX Portfolio Performance Measures

Fitness Value	Reciprocal value of the Quarterly Return Path Error (see above)
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)

3.5.1 Adaptive FX Trading Model Using Closest Fitness Rule Set

The following table describes the overall performance of the Forex Portfolio trading model during each of the testing periods (A-C) by using the closest fitness rule. The overall comment is that it is less successful during training, but the performance during the on-line evaluation is more successful and more consistent.

Table 5: Closest Fitness Rule Set Performance Overview

	Period Index	Fitness	Yield	YieldDD	Profit Months	Profit Quarters	No. Trades
Training	A	210.97	14.52%	2.95	70.27%	100.00%	91
Evaluation	A	40.25	3.64%	0.64	70.59%	66.67%	31
Training	B	112.07	12.64%	1.74	72.22%	88.89%	56
Evaluation	B	38.66	5.69%	1.20	52.94%	50.00%	22
Training	C	58.25	7.54%	0.99	63.64%	73.68%	86
Evaluation	C	43.16	6.66%	1.00	52.94%	60.00%	18
Continuous	All	42.12	5.11%	0.76	58.82%	64.71%	86

The continuous performance report most accurately reflects the performance of the system, when adapting new behaviour patterns, while still managing existing open position acquired during previous testing periods. Although an annual compound yield of slightly over 5% may not, with hindsight, represent highly successful behaviour, it must be noted that this result is generated relatively consistently during all application periods, using realistic assumptions for portfolio limits and transaction costs. Also important to note is a relatively good ratio between the worst performance drawdown and the yield, expressed at a Yield/Drawdown ratio of 0.76. This, we believe, is due to the integration of risk management controls into the trading decision, as implemented in the rules available to the trading model.

3.5.2 FX Portfolio Performance: Performance Consistency Across Training-Testing Sets

The following tables show each of the above performance measurements together with the ratio of each training set compared to the evaluation set. The ratio is calculated as [Test Result]/[Training Result]. In other words, the closer the ratio is to 1.0, the closer the testing result compared to the training result. To demonstrate how the selection of behaviour patterns affects the real-time performance, we have included tables for the closest-fitness rule set, compared to the top-fitness rule set selection. As it can be seen from the tables 6 and 7 below, choosing the closest-fitness rule set does indeed result in a much higher performance consistency compared to selecting the highest optimised rule set. If the top-fitness rule set had been selected for trading, period C would have resulted in a net loss. Comparing the performance consistency using all available performance benchmarks in the given table, the average ratio between testing/training set for the closest fitness rule set is 0.62, whereas the average ratio for the top fitness rule set is only 0.48!

Table 6: Performance Consistency of Top Fitness Rule Set

Fitness		**Period A**	**Period B**	**Period C**	**Average**
	Training	245.74	112.08	60.89	139.57
	Evaluation	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**
	Training	14.19%	12.62%	7.96%	0.12
	Evaluation	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	Training	72.97%	72.22%	67.27%	0.71
	Evaluation	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	Training	100.00%	88.89%	84.21%	0.91
	Evaluation	66.67%	50.00%	40.00%	0.52
	Ratio	*0.67*	*0.56*	*0.48*	*0.57*

Table 7: Performance Consistency of Closest Fitness Rule Set

Fitness		**Period A**	**Period B**	**Period C**	**Average**
	Training	210.97	112.07	58.25	127.10
	Evaluation	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**
	Training	14.52%	12.64%	7.54%	0.12
	Evaluation	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	Training	70.27%	72.22%	63.64%	0.69
	Evaluation	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	Training	100.00%	88.89%	73.68%	0.88
	Evaluation	66.67%	50.00%	60.00%	0.59
	Ratio	*0.67*	*0.56*	*0.81*	*0.68*

Hence, this trading model has successfully created a profitable, self-learning and self-adapting portfolio trading strategy over a diverse portfolio of currency pairs. The system has also exhibited relatively high consistency in performance, while adapting to new behaviour patterns. It is important to recognise however, that this development is still at a relatively early stage compared to what the technology is capable of. In absolute terms, neither return on investment nor drawdown risk are at a level which would be satisfactory for an investor or trading manager.

At the current stage of development, the actual performance is still not satisfactory as it does not follow the desired return path consistently enough. Next stages of development will test this trading model over a larger number of markets, using a more frequent re-training, to further test the system's capability to adapt to new behaviour patterns during trading.

During previous tests, we have seen the most significant increases in performance and performance consistency, when the system had been given more machine time to perform the learning process. Optimising such a large number of variables to fit a given return path involves searching a potential solution space of around 10^{18} possible trading models! The APT framework itself is not restricted to foreign exchange portfolios, but can be tested on any instrument or combination of instruments for which sufficient historical data are available.

4. Conclusion

In this note, we showed that adaptive, intelligent agents can indeed perform all functions that a human trader has to perform when managing a multi-currency and multi-instrument portfolio. The AI system further allows more precise measurement and prediction of trading behaviour and allows strict implementation of risk and portfolio thresholds.

Given adequate computational resources, this approach is scalable to include any number of instruments a trader or fund manager may wish to include in a portfolio. Thus, the ability of such a system, to monitor and decide upon investments, by far exceeds a human trader's ability to monitor large number of markets.

The results presented here are part of the beta testing of trading models implemented in the APT development framework. During our development we have seen the trading model performance dramatically increased when the system is given more resources to fully exploit its ability to create complex behaviour patterns. We are therefore convinced that further research and development into this area will create opportunities for new competitive global, systematic investment strategies.

APPENDIX I: Portfolio Allocation Calculation: An Example

Assume a trading model decided to base its portfolio allocation decision only on the relative volatility of markets. The allocation would then be calculated as $A_i = v_i \Big/ \sum_{n=1}^{N} v_n$, where v_i is the volatility measurement for the i^{th} market in the portfolio, N is the number of markets within the portfolio and A_i is the portfolio allocation for the i^{th} market in the portfolio. It is up to the trading model to decide whether variance of prices, historical trading ranges, or indeed other methods to describe a market's behaviour, are being used for calculation. Mathematical operators can be applied by the system to the right-hand side of the equation (replacing the simple volatility measurement of this example with a more complex expression), or a limited number can be applied to the result of the calculation itself.

For instance, using a given market's N-days historical trading range (TR_N), N′-days historical variance of prices $(V_{N'})$, and N″-days historical cross-correlation $(C_{N''})$,the system could build an allocation model as follows

$$F_i = d\min(eTR_{N_i}, j \max(jV_{N_i'}, gC_{N_i''}))$$

with $A_i = F_i \Big/ \sum_{n=1}^{N} F_n$ where F_i is the functional expression used for the calculation and δ, ε, ϕ, γ are limited range parameters that can be optimised to adjust the portfolio allocation to better fit within the absolute level of a defined return path. The benefit of this concept is that portfolio allocation can be at any time re-calculated for every single component of the portfolio, when a new position is being entered into, or when the allocation is re-evaluated during a mark-to-market process. The number of input variables can be expanded, so it would be possible to use this concept, e.g. to re-allocate trading lines between individual funds or individual traders using a measurement already used for monitoring these traders, such as Value-At-Risk calculation. This is a concept that translates the analysis into a measureable, cross-validated, decision-making concept. A more consistent and diversified portfolio is the result.

APPENDIX II: Risk Management Calculation

The adaptive trading model system uses the result of both the price risk calculation and the portfolio allocation, to define the position size to be taken, for either a new trading decision or to adjust a given market position. The position size is the actual number of units that should be bought or sold for a given market (the term unit may mean: number of shares, number of futures contracts, or US$ mln for foreign exchange). The link between the input data and the calculation of position size, is the actual risk the system is prepared to allocate to each trading decision. Although this could be pre-defined by the trading manager, we have decided that this Forex trading model should also decide what the appropriate level of risk would be, restricting ourselves to only defining the global portfolio parameters.

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 portfolio risk is the percentage of the portfolio that the system puts at risk for every single position. Why should this be left to the trading model and not be decided by the trading manager? An investor or trading manager normally has a very clear idea of the kind of portfolio performance which is expected, and the amount of risk that she is prepared to take in return. This, however, is expressed in global terms for the entire portfolio. It is not easily possible to translate this general constraint into a limit for each individual position, because the appropriate risk depends on the frequency of trading, on market conditions, and on the performance of the entire portfolio itself (accepting the consistency and predictability of performance as priority). It is therefore sensible to let the trading model, through its own learning process, find ways to calculate the required risk in such a way that it best meets the portfolio risk/performance requirement defined in the return path (and other portfolio specifications, see table 1).

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 comptible 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.

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.

Bibliography

Bauer R. J. (1994), *Genetic Algorithms and Investment Strategies,* John Wiley & Sons, New York.
Bundy A. (1996), *Artificial Intelligence Techniques,* Springer, New York.
Goldberg D. E. (1989), *Genetic Algorithms,* Addison Wesley, New York.
Holland J. (1995), *Adaptation in Natural and Artificial Systems,* 4th edition, MIT Press, Cambridge.
http://alife.santafe.edu
http://alife.santafe.edu/alife/topics/gp/
http://www.rabatin.com
Internet Newsgroups *comp.ai.**
Koza J. R. (1992), *Genetic Programming,* MIT Press, Cambridge.
Ladd S. (1995), *C++ Simulations and Cellular Automata,* M&T Books.
Michalewicz Z. (1992), *Genetic Algorithms + Data Structures = Evolution Programs,* Springer, New York.

Rosenberg M. (1996), *Currency Forecasting,* Irwin.
Toulson D. L. and S. P. Toulson (1997), *A Trading System for FTSE-100 Futures Using Neural Networks and Wavelets*, BNP Working Papers in Financial Economics, no.3.
Vince R. (1992), *Mathematics of Money Management,* John Wiley & Sons, New York.
Vince R. (1995), *The New Money Management,* John Wiley & Sons, New York.