# Designing safe, profitable automated stock trading agents using evolutionary algorithms

Harish Subramanian, Subramanian Ramamoorthy, Peter Stone, Benjamin J. Kuipers
Artificial Intelligence Laboratory, Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712, USA
harish.subramanian@alumni.utexas.net, s.ramamoorthy@mail.utexas.edu,
(pstone,kuipers)@cs.utexas.edu

## ABSTRACT

Trading rules are widely used by practitioners as an effective means to mechanize aspects of their reasoning about stock price trends. However, due to the simplicity of these rules, each rule is susceptible to poor behavior in specific types of adverse market conditions. Naive combinations of such rules are not very effective in mitigating the weaknesses of component rules. We demonstrate that sophisticated approaches to combining these trading rules enable us to overcome these problems and gainfully utilize them in autonomous agents. We achieve this combination through the use of genetic algorithms and genetic programs. Further, we show that it is possible to use qualitative characterizations of stochastic dynamics to improve the performance of these agents by delineating safe, or feasible, regions. We present the results of experiments conducted within the Penn-Lehman Automated Trading project. In this way we are able to demonstrate that autonomous agents can achieve consistent profitability in a variety of market conditions, in ways that are human competitive.

## Categories and Subject Descriptors

[**Real World Applications**]: finance, intelligent agents, automated trading

## General Terms

Algorithms, Design, Experimentation, Performance

## Keywords

Genetic Algorithms, Genetic Programming, Finance, Application, Fitness Evaluation

## 1. INTRODUCTION

Recent developments in the automation of exchanges and stock trading mechanisms have generated substantial interest and activity within the machine learning community [13, 17], including the evolutionary algorithms community [3, 4, 5, 8].


*GECCO* '06 Seattle, WA USA


When building autonomous agents, it is desirable to structure them in a modular fashion - as combinations of simple interpretable pieces. Such a structure could enable the identification of specific transferable features in the strategy that contribute to the observed success. This is crucial if these algorithms are to be adopted in the real world of banks and investments, where the stakes are high and practitioners need some assurance of safety and, at least statistical, bounds on profitability. Correspondingly, when a strategy is found to be unprofitable, it can often be improved if its failure can be attributed to specific causes that can then be eliminated.

One way to achieve such modular design is through the use of *technical trading rules*. Such rules are widely used by practitioners. For instance, a survey of foreign exchange traders in London [19] estimates that up to 90% of them use some form of trading rules in daily practice. As such, the popularity of the rules flies in the face of theoretical objections arising from the efficient markets hypothesis, [6, 12], that asserts that stock markets are random processes lacking in systematic trends that may be exploited by mechanized rules. However, with the emergence of high frequency finance [7] it has been demonstrated that the efficient market hypothesis is not entirely valid in the short time scales at which autonomous agents are able to operate. Moreover, recent evidence [1, 11] has suggested that the profitability of trading rules can be rigorously characterized and the simplicity of these rules does not necessarily imply a lack of theoretical justification for their use. As stated in [1], there is ample evidence to suggest that trading rules are very well suited to autonomous agents operating in electronic markets.

A more pressing concern, from the perspective of agent design, is that each trading rule is profitable in certain environments or market conditions and there are corresponding market conditions under which it could be unprofitable. This dichotomy suggests that no rule can be used in isolation for any reasonable period of time. It also implies that the emphasis of design needs to be on composite rules - as noted in [4], we need to move beyond the "technical trading rule" and consider the "technical trader". In fact, when people use trading rules, they observe several indicators and use their judgement to arrive at a composite decision. Naive combinations of trading rules are rarely, if ever, useful and prudent. So, the question arises as to how the human judgement involved in combining these rules can be emulated in an automated strategy acting within a computer program? We will address this question in this paper.

Another key aspect of human judgement is the way we handle asymmetry between upside and downside risks. Traders might welcome the occasional windfall profits but they certainly want to avoid bankruptcy at all costs; and they act in such a way that these preferences are satisfied. For an automated agent emulating

such behavior, we seek some assurance that worst-case losses are bounded below at an acceptable level. In an uncertain trading environment, and with limited information, are there mechanisms that enable us to arrive at such bounds? It seems unlikely that we can establish these bounds in an absolute sense - after all, this is a stochastic environment notorious for its inherent risks. However, we do seek a statistical characterization that assures us that our profits are consistent in an acceptable way. This is what we mean by the term *safe agent design*.

In this paper, we present an approach to autonomous agent design that explicitly addresses these issues. We show that it is possible to achieve safety by qualitatively characterizing the domain of profitability of each rule and devising multiple model strategies that attempt to navigate away from unsafe regions of the state space. We establish the soundness of this concept by discussing the properties of a simple hand-constructed agent that won an open multiagent competition. We then show that through the use of evolutionary algorithms acting within this framework of safe agent design, we can construct fairly sophisticated agents that achieve levels of performance that are at least comparable to, and often superior to, what humans are usually able to achieve. We present two different designs based on evolutionary algorithms. A genetic algorithm is used to optimize a weighted combination of technical rules. Each generation of candidate solutions is optimized to increase the average returns and reduce the volatility of returns - over a fixed period of training days. Simulations are run to tune the weights to trade robustly over varying market conditions. A similar design is also implemented with a genetic program that evolves complex rules from a pool of constituent rules and boolean operators to combine them. We evaluate various fitness functions and performance measures including a simple measure of risk adjusted return and one that penalized only downside volatility. We compare their merits and effects on the agents, and evaluate their performance when used as a tuning parameter for the GA.

Trading on real world electronic markets, that fulfil orders in an automated fashion, is a costly exercise. To allow us the freedom of experimenting in a limited, but realistic electronic exchange, we use a simulated environment as part of the Penn-Lehman automated trading (PLAT) project [9]. It uses real-world, real-time stock data and incorporates complete order book information; simulating the effects of matching orders in the market. It also allows the participants to program their own strategies with ease within the client-server mechanism, and trade with other agents as well as the external market.

## 2. THE PENN-LEHMAN AUTOMATED TRADING PROJECT

The Penn-Lehman Automated Trading (PLAT) project is a broad investigation of algorithms and strategies for automated trading in financial markets. The agents described in this paper were constructed as a part of this research effort. All agents implemented within this project trade using the Penn Exchange Simulator (PXS). PXS is a software simulator for automated trading that merges automated client limit orders with real world, real time order data available via modern Electronic Crossing Networks (ECN). The simulator periodically queries the ECN to get order book information, which it then uses to match buy and sell orders. The order books from the Island external market and the internal market created by PLAT agents are merged, i.e., the simulator tries to match the agent orders against actual Island trades and if no such match is available, it tries to match against another agent's order.

Some of the agents described in this paper were initially designed to participate in an open multi-university competition. The participants in the competitions were researchers from several major research universities in the USA and UK. The contests each consisted of 10 days of live trading. As mentioned earlier, the market clearing mechanism combined the order books received from NASDAQ and an order book that was internal to the simulated environment. The rules of this competition include the following clauses that define the trading task: (1) The task is that of intraday trading. Each agent begins with no money and no shares at the start of each trading day. The agent may take long or short positions during the day in an attempt to improve net worth. (2) There are no limits on the number of shares that can be held at any given time. However, every agent must liquidate its position at the end of the trading day. Any shares held at the end of the day will be valued at 0. Any shares borrowed from the simulator will need to be purchased at a cash value of twice the current price. (3) PXS runs in transaction cost/rebate mode for the competition. Every time PXS executes a trade, one side of the order must have already been sitting in one of the order books, and the other side of the order must have been the "incoming" order. For each share executed by PXS, the party whose order was already in the books receives a rebate of $0.002, and the party that was the incoming order pays a transaction fee of $0.003. This is exactly the policy used by Island ECN. (4) The performance measure for the contests is a form of the Sharpe ratio [16]. Additionally, for our experiments, we employ a modified risk adjusted return as a measure based on the policy that we are not perturbed by the possibility of volatile return structure provided the strategy is mostly profitable. Sortino ratio [15], is a modification of the Sharpe ratio that differentiates 'harmful volatility' from volatility in general, using a value for downside deviation only.

The discussion in this paper will be framed in terms of the following variables and functions:

- Price, $p_n$, is a random variable (where *n* is a timestep)
- Volume of shares, $v_n = f(v_{n-1}, ..., v_0, p_{n-1}, ..., p_0)$
- Cash in hand, $c_n = g(c_{n-1}, \Delta v_n, p_n)$
- Net Worth, or Value, $V_n = p_n \cdot v_n + c_n$
- The goal, for a single day of trading, is to maximize $V_n$.
- The performance of the agent is evaluated by Sharpe and Modified Sortino ratios based on $V_n$ at the end of several days of trading.
- The goal of agent design is to design the function $f$ to maximize the performance measure.

## 3. QUALITATIVE BEHAVIORS OF TRADING RULES

Composite rules need to be constructed from simpler rules. What should the basis set of simpler rules consist of, using which we compose the more complex rules? The intuitive answer is that they must complement each other. For instance, given a particular set of trends during a trading day, if one rule were unprofitable then another rule(s), adapted to the specific conditions of the day, should be able to take over. As such the space of trends is infinite, so we need a more compact language to characterize rules in a basis set. In this Section, we present one way to approach this issue. Trading rules behave as predictive filters [14]. Profitability is a function of the dynamic properties of the predictive filter and the dynamics of the stock price. In this section, we consider the qualitative behavior

of one rule, the contrarian strategy, but this analysis can be extended to other rules as well.

Consider the Contrarian strategy, defined in terms of sign relationships. As long as an upward or downward trend is detected, the volume is corrected downward or upward respectively.

$$v_n = \begin{cases} v_{n-1} - \gamma & : \quad \Delta p_n > \delta \\ v_{n-1} + \gamma & : \quad \Delta p_n < -\delta \\ v_{n-1} & : \quad -\delta < \Delta p_n < \delta \end{cases} \tag{1}$$

Often, the decision $\Delta p_n > \delta$ can not be made instantaneously [14]. In practice, this is approximated by using moving averages. An arithmetic moving average rule, based on a moving window can be used. A more useful form of this approximation can be achieved by comparing two windows of different lengths.

$$\sum_{j=0}^{m_s-1} p_{n-j}/m_s \geq \sum_{j=0}^{m_l-1} p_{n-j}/m_l + \delta \Rightarrow \Delta p_n > \delta \tag{2}$$

where $m_l > m_s$.

In order to understand the behavior of this rule, we need a characterization of the dynamics of the price. Consider a price process that is a mean reverting log random walk (at any given time, the returns can be assumed to be randomly chosen from a lognormal distribution) with drift,

$$\begin{aligned} x_n &= \ln(p_n) - \ln(p_{n-1}) \\ x_n &= \alpha + \beta x_{n-1} + \epsilon_n \end{aligned} \tag{3}$$

where $\epsilon_n \sim iid(0, \sigma^2)$.

We may characterize the conditions required for profitable operation (with the trading rule of equations 1,2, window sizes $m_s, m_l$ and the price process of equation 3) in terms of the probability distribution of daily returns. In specific cases, the conditions can be derived analytically. For instance, [2] presents analytical conditions for profitability, with $m_s = 1, m_l = 2$, and [1] includes some generalizations. However, in general, this becomes somewhat involved. An alternate approach to gaining a similar understanding is through the use of Monte Carlo simulation. We adopt the following procedure to perform a simulation study:

1. Select models for the price process and trading rules
2. Generate a large number of sample paths of the price process and apply the trading rule. A typical setting might consist of 3000 sample paths, each path corresponding to a day's trading, with trading opportunities separated by 1 minute intervals.
3. Collect sample statistics of the value at the end of the sample paths

Performing such an experiment for the Contrarian strategy yields the following characterization:

- The rule is profitable, in a statistically significant sense, in a mean-reverting log random walk market.
- The rule results in zero expected profits in a simple log random walk market.
- The rule is potentially loss-making in a log random walk market with drift, i.e., $\alpha \neq 0$.

If this rule were to serve as a component of a composite strategy, then the primary requirement is that whenever $\alpha \neq 0$, we should avoid this rule and prefer another rule that is profitable under that condition. Many trading rules can be characterized in this way. Even when such a formal characterization is hard to come by, it stands to reason that if we have a sufficiently diverse 'bag' of rules, we can compensate the weaknesses of each rule by substituting another rule that is of a complementary nature. In this way, the composition is significantly more robust than each individual rule. The above characterization is qualitative to the extent that we are able to make statements based just on the sign of key quantities, e.g., $\alpha \neq 0$ or $\beta \neq 0$. This ensures that a composition based on these rules can be expressed compactly and without depending on detailed identification of market conditions. In situations where the rules do not admit such a qualitative representation, we can still use the same approach to partitioning state space, but we will need more detailed quantitative information about the price process.

# 4. COMPOSING MULTIPLE TRADING RULES

In the previous Section, we arrived at the rationale for when a particular rule should be invoked. How do we make this decision in real time? Characterizing statistical distributions in real-time is hard. In this Section, we propose an alternate, surrogate, test that allows us to make this decision. The state of a trading agent can be represented by two important variables - cash in hand, $c_n$, and volume of shares, $v_n$. In a phase space defined by these two variables, depicted in Figure 1, the result of a day's trading defines a trajectory. Ideally, the agent follows a path along the direction of increasing $c_n$ while staying close to $v_n = 0$, thus minimizing risk.

Now, we wish to identify, without fitting statistical distributions, when the agent is in one of the unprofitable regions of state space. Intuitively, the agent is in an unprofitable region when its performance falls below that of a simple random walk market. In the $c_n - v_n$ phase space, the random walk market yields a line that has the property that a trading rule that is neutrally profitable would stay on this line, exchanging cash for shares and vice versa, but not making any profit. A profitable agent will follow a trajectory that lies on one of the half planes defined by this line. An unprofitable agent enters the other half plane. By observing the trajectory of the agent in real time, we can detect unprofitability by monitoring entry into this undesirable half-plane.

In addition, for a day trading agent such as the agents participating in the PLAT competitions, one of the requirements is that it must unwind its position by the end of the day or risk the possibility of great losses overnight (due to market activity outside trading hours). This acts as a boundary value constraint that implicitly restricts the maximum position taken during the day. Combining these two requirements, we specify that a trading rule, e.g., the Contrarian strategy, should be invoked only as long as $(V_n - p_0 \cdot v_n + c_n > 0) \wedge (c_n > c_{nmin}) \wedge (v_{nmin} < v_n < v_{nmax})$. This test encodes the intuitive idea of "falling below" the performance of the simple random walk market and hence acts as a surrogate test for safety, from the notation introduced in Section 2 for net value and cash/share positions.

Our approach to safe composition consists of ensuring that we have a collection of trading rules that have complementary directional properties and that we monitor their progress using the mechanism described above in order to make changes. A simple agent that embodies this approach is the the following: If $(V_n - p_0 \cdot v_n + c_n > 0) \wedge (c_n > c_{nmin}) \wedge (v_{nmin} < v_n < v_{nmax})$ then invoke the Contrarian strategy, else Divest (i.e., reduce share position) using a Safe Contrarian strategy (a modified version of the original rule).

This rule is kept deliberately simple to illustrate ideas. In later sections, we will discuss more complex combinations. However, we

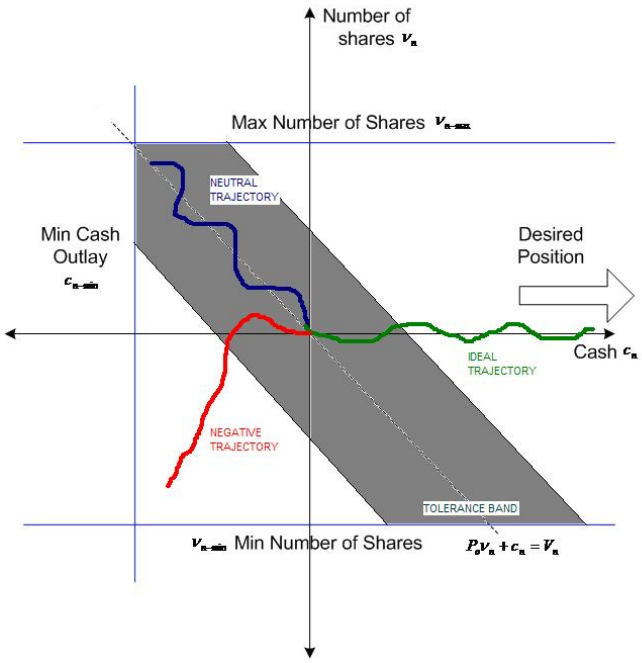


**Figure 1: Agent behavior represented along the cash-share axes**

note that even such a simple rule, constructed using the principles outlined above, is capable of impressive results. This strategy functions such that the Contrarian strategy is invoked and the agent is guaranteed non-negative returns. In a drifting market, the Contrarian rule is unprofitable and eventually the divestment strategy is invoked, which brings the agent back towards the desirable region.

This agent was submitted to the April 2004 PLAT competition. The experiment consisted of an economy of six agents that included the proposed agent, called SMM, four other competing agents with the same trading objective as ours, and a VWAP agent intended to provide liquidity. We used Sharpe ratio, a measure of risk adjusted returns, as a means of evaluating the performance of these agents. The results of the live competition show that the SMM achieved consistent profitability - as reflected by the Sharpe ratio that was several multiples higher than that of other agents.

| Test Set | **SMM** | #1 | #2 | #3 | #4 |
|---|---|---|---|---|---|
| Avg. Profit | **2519** | 239 | 4725 | 1057 | 148 |
| Std. Dev. | **1009** | 316 | 6551 | 1829 | 3413 |
| *Sharpe ratio* | **2.49** | 0.76 | 0.72 | 0.58 | 0.04 |

The primary benefit of the proposed monitoring mechanism is that the trading rule is able to adjust its risk by divestment, when market conditions are unfavorable, in order to avoid excessive losses. In later sections, we will discuss combinations of several rules, some of them more sophisticated than the ones described above. However, the foregoing discussion provides us reason to believe that our approach and monitoring mechanism will allow us to achieve a measure of robustness. With this encouraging evidence, we may use evolutionary optimization to aggressively search for higher levels of performance. This will be the focus of the remainder of this paper.

# 5. EVOLUTIONARY AGENT DESIGN

Having understood the basic principles of composite agent design, we apply them to the design of an evolutionary learning algorithm that uses optimization to determine the best weighting between a set of rules. If each component rule yields a decision, $D_i \in \{-1, 0, 1\}$ corresponding to {*Sell*, *Do Nothing*, *Buy*}, then a combined decision would be $D_{comb} = \sum \omega_i \cdot D_i$. The goal of the evolutionary optimization algorithm is to determine the best set of weight vectors, $\omega_i$ that would optimize the fitness function in an effort to maximize the performance of the trading agent. The aim of our design is to maximize profitability. In addition, we incorporate our test for safety to bias the search algorithm towards safe and feasible regions. The evolutionary algorithm learns both the trading decision and the trade volume simultaneously.

The evolutionary learning process identifies the best strategy by searching among the weights, to identify the right composite trading rule. However, in practice, the competing objectives of maximizing profits and managing risk to avoid penalties makes this learning problem hard. To evaluate the benefit of bootstrapping such learning agents by providing the qualitative characterization and allowing the exploration to focus on identifying quantitative optima, we performed several experiments. These, in turn, isolate the various elements of the agent design. We use two variations of the general agent architecture. The first experiment is based on a genetic algorithm agent, called GAA. In this agent, $\omega_i$ is encoded as binary strings and hence quantized to discrete values of volume. A variation of this agent called GAAMM is defined as,
If $(V_n - p_0 \cdot v_n + c_n > 0) \wedge (c_n > c_{nmin}) \wedge (v_{nmin} < v_n < v_{nmax})$ then invoke the decision $D_{comb} = \sum \omega_i \cdot D_i$, else $D_{comb} = 0$.
The second experiment is based on an agent that uses genetic programming [10], called GPA. This agent uses a decision tree to combine the individual decisions. In our notation, this implies that $\omega_i$ could take on finer resolution values than in the case of GAA. The modified version of GPA, called GPAMM, is similar to GAAMM in the use of the multiple model safety mechanism (differing from GAAMM in the representation of $\omega_i$).

The design is based on a representative set of days including a variety of different market conditions. The PLAT domain is configured to run with historical as well as live data. For the purpose of our experiments, we use a single stock (MSFT), and data used for training and testing is selected to include various market conditions (increasing, decreasing, mean reverting, etc.) The data, selected from over 6 months (archived in the PLAT project), was used as a baseline. After pruning for incomplete or corrupted data, we tried to include representatives of a number of different stock behaviors over a training day were included. Training in a non-volatile market (or other homogeneous market patterns) alone could yield a set of weights that would be disastrous in a volatile market, and attempts to eliminate this over-fitting effect have been made. A non-homogeneous subset of the selected trading days, set aside as test day for competitive tests, is substantially different from the training data - both in terms of trading days or market behavior patterns.

# 6. PERFORMANCE EVALUATION AND FITNESS FUNCTIONS

The criteria for evaluating the performance of our agents in the experiments we performed, are measures of risk-adjusted investment returns. The reason these are good metrics, in addition to the profits is that consistency is rewarded and volatile trading patterns are not. Common measures within this class are the *Sharpe* and *Modified Sortino* ratios. These measures reward consistency as

well as profitability. Since these are the two tenets of our design, they are ideally suited for evaluation of their performance.

We use both these ratios, in turn, as fitness functions and as evaluation measures. If $\rho_i$ is the daily return or the profit/loss of the agent at the end of trading day $i$ (after adjusting for transaction fees and penalties), $\mu$ is the mean value of all $\rho_i$ and $\sigma$ is the standard deviation, then the Sharpe ratio (SR) is, $S = \mu/\sigma$. Despite the common use of Sharpe ratio in the field of financial performance evaluation, we are not perturbed by the possibility of volatile return structure provided the strategy is mostly profitable. Sortino ratio, is a modification of the Sharpe ratio that penalizes only the harmful volatility, i.e., downside deviation. The form of this ratio that we will use in following sections is *Modified Sortino Ratio*, which is $\mu/\sigma_n$, where $\sigma_n$ is the standard deviation of negative returns only. The evolutionary algorithms in our design determine optimal weights to maximize the measures of performance. Owing to the iterative nature of simulations over the complete trading cycle, the fitness functions for the optimization of the genetic algorithms are the performance evaluation criteria themselves. We optimize the weights of the algorithm by simulating trades over a sample period of trading days. On evaluating the Sharpe and Modified Sortino ratios at each generation, we evaluate the fitness of the candidate solutions. The performance evaluation measure that tests how well the agent has done while trading in a test set of days, acts as the fitness function (the parameter to be optimized) in a unique training set.

# 7. IMPLEMENTATION OF GENETIC ALGORITHM AGENT

In the previous section, we described the general principles of designing an agent where several rules are combined and the combination weighted using an evolutionary algorithm. Let us examine in greater detail, the specifics of these designs. We have it set up such that the final output of the agent is a trading decision (deciding to abstain from trading is also an action) produced at frequent samples of a trading day when the books are updated and fresh data is available. In addition to designing an automated strategy that is intuitively appealing, the generation of effective strategies using complete, comprehensible indicator strategies also helps in the understanding of these component strategies, their effects and limitations. Additionally, it allows for substitution, addition and deletion of component rules from the overall rule set. The encompassing safety mechanism (listed in the figure below as Multiple Model Control) determines the mode of operation and is analogous to a faucet. It acts as the final regulator on the trading decision and volume, allowing only those trades that move the agent towards increased profitability.

- Each of the weights, represented as discrete bits in a string, include a sign bit that helps evaluate the suggested action and the strength of the suggestion in a single string. This representation is particularly useful in possibly expanding the search space available to the GA as well as in compensating for some misrepresentations of the component (indicator) strategies, allowing the agent to tune itself for better performance.

- To ease the burden of computation time, we used a small number of finite, discrete values for the weights at the cost of increased granularity in weight values. Longer strings would make the search space of possible solutions much larger, owing to the increased number of unique combination of bits and this would make it unreasonably computation intensive.

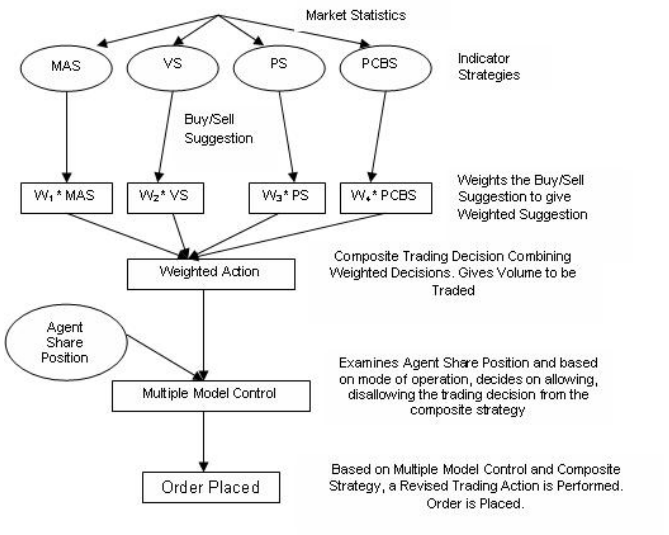


**Figure 2: Mechanics of working of GAA agent**

- We split the strings into buy and sell components to accommodate for the asymmetry in their trending behavior.

- The genetic algorithm iterates through generations of population, fitness calculation, crossover and mutation stages until convergence is achieved. In view of the heavy computation time associated with each generation, we set a maximum number of iterations at which to stop iteratively computing the strings.

- A population of ten strings for each of the buy and sell rules, initialized randomly, allow for sufficient exploration of the search space.

- An elitist approach to the selection of candidates for subsequent generations encouraged convergence.

- Each generation is populated as a result of crossover and mutation operations performed on the selected strings.

- The fitness of candidate strings was calculated in each generation using the Sharpe or Modified Sortino ratio, with the goal of the evolution process to maximize the fitness functions (and find the combination of weights that achieves this).

- The components (indicator strategies) were simple adaptations of *moving average crossover, price channel breakout, order book volume imbalance* and *simple price trend*, denoted by MAS, PCBS, VS and PS respectively.

The bit strings consisting of 2 bits corresponding to each weight, can take on 7 discrete values (including 3 non-zero values for each of the positive and negative signs prefixing them). For a pool of four indicator strategies, that gives us a bitstring of 12 bits. The addition of rules will necessitate retraining the agent with the longer string or by replacing an insignificant component rule. In the very first generation, the population is initialized using uniform pseudo-random integers which are translated to strings, and constitute the initial population. The top k percent of strategies are used for crossover, and bit strings are crossed over tail to head. The probability of being selected for crossover is higher for strategies with

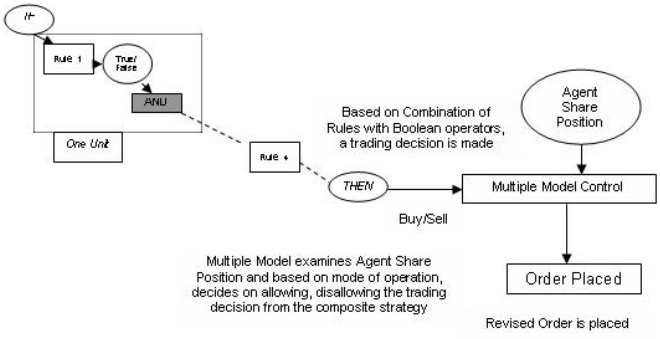


**Figure 3: Mechanics of working of GPA agent**

higher fitness. In order to preserve strings with high fitness in our search for local as well as global maxima, we maintain an elitist model since the top 2 strings (from each of the buy and sell sides) are spared mutation. After checking to see if the resulting strings are unique, we use a relatively high mutation rate of 10 percent, to enable exploration of the search space inspite of the small population size. Convergence is said to occur when mean fitness changes by no more than 10 percent and maximum fitness changes by no more than 1 percent between the previous and current generation.

# 8. IMPLEMENTATION OF GENETIC PROGRAM AGENT

We now present a design, using a genetic program, where we combine various rules using boolean operators in place of the weighted majority approach in the previous section. This allows us to grow, using a conventional tree structure, composite rules of varying length and varying complexity of interaction between the component rules. The design of this agent is very similar to the GAA in the use of bit-strings. In place of using a sign operator bit, we use bits for boolean connectors. The binary string is an effective representation because complex statements (including Boolean logic and numerical values of parameters) can be represented in this form. The primary difference between the GP Agent (GPA) and the GAA are:

- the allowance for growth of trading strategies based on combination of rules and thus a variable number of component rules
- the use of Boolean operators as combining elements in place of the weights in the GAA.

The use of GP allows for optimized strategies based on single rules as well as a fixed number of chosen indicators. In a sense, the solution space of a GA is a subset of that of a GP. Trading strategies are constructed by allowing the genetic selection engine to combine the component indicator rules with Boolean operators. Strategies are once again split into buy and sell rules. In each case, the GP has the potential to choose from buy and sell rules based on multiple technical indicators. The Boolean operators AND, OR and XOR are used to compose these buy and sell rules. Finally, as a cash and share position management, we use the multiple model control mechanism, like in GAA. The order price is similar to the price used in GAA and the order volume in GPA was constant - the value determined empirically. Order volume is determined by frequency. For this purpose, we assume n consecutive orders of m shares each to be close in value and fulfillment ability as an order of n.m shares. The exclusion of volume as a tuning factor in this experiment was made after problems with convergence of the algorithm. We hypothesize that as the search space in this form of a GP is large (much larger than the GAA representation), the inclusion of volume makes the search space so intractably large that convergence took a very long time. Apart from Boolean connectors, a difference in design from the GAA is in the use of rule in use bits that act as flags. Each generation of evolving strategies in the GP follows the same steps (population initialization, fitness calculation, crossover and mutation) as that of the GAA, and the mechanics of these steps remain the same.

# 9. EVALUATION OF AGENTS - DEVELOPMENT

So far, we have examined various building blocks that make up our agent, and how they are composed to make a composite trading agent. We now turn our attention to evaluating the agents in tests of their performance. Specifically, we aim to show that:

- the combination of rules is superior to each of the components
- a weighted composition of these rules is better than a naive combination
- a regulatory multiple model mechanism provides improved performance by allowing the agent to switch between various modes of varying risk aversion
- the composite trading agent designs perform well in a simulated, realistic economy with competing agent designs.

In this section, we use the Sharpe ratio and raw profits as measures of evaluation. In the following section, we will examine the use of the Modified Sortino ratio as a fitness function as well as a measure of performance. We first compare the agents' performance against the component strategies running alone. If the components exhibit similar or superior profitability, there would be no benefit in combining them with others. We test both the GAA and GPA in two separate experiments with different out-of-sample test sets of 15 trading days each, in a competition with the component strategies and various control strategies.

| Test Set 1 | **GAA** | MAS | VS | PS | PCBS |
|---|---|---|---|---|---|
| Avg. Profit | **714** | 95 | -307 | 10 | -747 |
| *Sharpe ratio* | **0.7** | 0.095 | -0.177 | 0.006 | 0.548 |

| Test Set 1 | **GPA** | MAS | VS | PS | PCBS |
|---|---|---|---|---|---|
| Avg. Profit | **319** | 96 | 12 | 43 | 268 |
| *Sharpe ratio* | **0.285** | 0.105 | 0.011 | 0.041 | 0.371 |

| Test Set 2 | **GAA** | MAS | VS | PS | PCBS |
|---|---|---|---|---|---|
| Avg. Profit | **430** | 232 | 170 | -253 | -314 |
| *Sharpe ratio* | **0.411** | 0.201 | 0.114 | -0.181 | -0.333 |

| Test Set 2 | **GPA** | MAS | VS | PS | PCBS |
|---|---|---|---|---|---|
| Avg. Profit | **325** | 210 | 12 | 80 | -256 |
| *Sharpe ratio* | **0.346** | 0.24 | 0.013 | 0.082 | -0.217 |

The results show that GAA and GPA outperformed each of the component strategies in both test cases - an important result regarding the benefit of the composite strategy. Even within the component strategies, we can see that there is substantial variation between the various strategies' performance, indicating that it may be useful to include different weights for the various indicator suggestions. This variation can be attributed to a combination of different test sets of data (market economies), their interaction with varying

performances of the other agents that made up the economy, and general variance.

This supports our initial hypothesis that the indicators, acting on the same data set and when used in combination, do not lend suggestions of equal power. This leads us to believe that a weighted majority schema may be more appropriate, as opposed to a simplistic combination. In order to verify this, we take the case of the GAA, and compare the use of tuned weights against fixed and equal weights, for two different test sets of data.

| Test Set 1 | **GAA** | Equal Weights |
|---|---|---|
| Avg. Profit | **715** | 178 |
| *Sharpe ratio* | **0.701** | 0.18 |
| Test Set 2 | **GAA** | Equal Weights |
| Avg. Profit | **431** | -12 |
| *Sharpe ratio* | **0.411** | -0.016 |

The results confirmed our hypothesis. We venture to explain the advantage that the tuned agent has with the following:

- Equal weights would imply that each of these indicators were equally confident when they gave a buy or sell recommendation at each time step. This is probably not the case. For example, a price breakout may be more of a 'sure bet' in suggesting a trend than a simple price check. So, the weights should be allowed to lend varying suggestive powers to the components accordingly.
- Different weights for buy and sell sides accommodate the contingency that an indicator may be more confident of a buy signal as opposed to a sell signal, or vice versa.

To fit the final piece of the agent design puzzle, we proceed to verify the benefit of bootstrapping the agent with qualitative information to provide a multiple model control mechanism. The results show that the evolutionary agents outperformed the ones without multiple models. The results of all of the experiments so far, clearly illustrate that the agents that were bootstrapped with qualitative information perform significantly better.

| Test Set 2 | Sharpe | Test Set | Sharpe |
|---|---|---|---|
| GAA | 0.701 | GPA | 0.133 |
| No MM | 0.255 | No MM | -0.101 |

In the absence of the multiple model component, the strategy tends to trade lower volumes and allots higher weights to indicator strategies that are conservative (that have higher thresholds). This can be attributed to the inherently safe regions that encapsulate the trading agents' exploration. The agent is willing to take a riskier position, with the confidence that it will be led back to safety. There are two ways of optimizing the performance ratios - increasing profit everyday, or decreasing variance. Without qualitative information, the optimization leads to a lower variance and lower profit. With the multiple model component and higher profits (from higher volume of trades), the variance is also higher. The agent, in the presence of the MM component, trades much closer to the boundaries of safety since it knows where they are.

## 10. EVALUATION OF AGENTS - COMPETITION

When agents are trading in a market, the price dynamics are affected by competing agents, each with a view to maximizing its own profit. Agent interaction and the uncertainty it causes is often a cause of worry for designers of automated strategies. In this

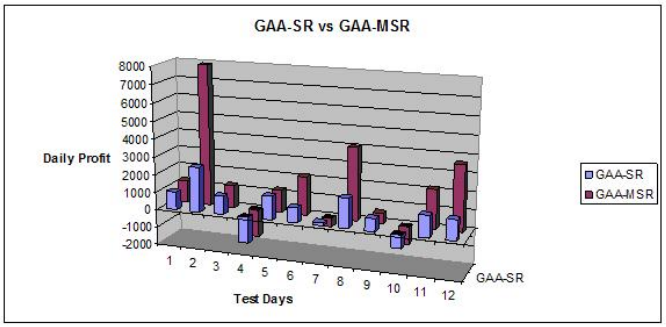


**Figure 4: Performance Comparison of GAA tuned with Sharpe and MSR**

section, we test the agents in open competition with other agents that have different strategies, but want to be as profitable as possible. In the process, we test the agents' performance in a more diverse environment. These agents are competitive and are all aiming to be profitable. In addition to GAA and GPA, we use successful agents from open competitions on the PLAT environment, including agents SOBI [18], Market Maker(MM) [17] and Volume-based agent (VBA). Earlier, we had hypothesized that fitness function is a key component in deciding the effectiveness of the agent. We proposed the use of *Sharpe ratio* (SR) and *Modified Sortino Ratio* (MSR) as performance measures, but also as a means of evaluating fitness of candidates in the evolutionary algorithms. We expect that penalizing negative volatility alone would result in an agent with higher trading profitability as there is no limit on positive spikes or windfalls that the agent may make on particular days. The results of our experiments confirm our hypotheses.

| Fitness function used | Sharpe | Sharpe | MSR | MSR |
|---|---|---|---|---|
| Agent design | GAA | GPA | GAA | GPA |
| Avg. Profit | 797 | 415 | 1774 | 660 |
| Sharpe | 0.752 | 0.431 | 0.691 | 0.430 |
| MSR | 1.762 | 0.964 | 3.54 | 1.07 |
| MSR/Sharpe | 2.345 | 2.233 | 5.122 | 2.49 |

Finally, we test the agents' performance in competitive tests. We use all four agents (GAA and GPA, each tuned with Sharpe ratio and MSR). Competing agents include a market maker (MM) and Simple Multiple Model Agent (SMM) which were the winning agents in the live PLAT competitions in December 2003 and April 2004. These are used to benchmark the performance of our evolutionary algorithms against competitive agents, in tests of 15 trading days each. The results are encouraging and suggest that the evolutionary agents are profitable in a competitive environment, and are in fact most successful when the fitness function and other tuning parameters are chosen carefully to optimize the right parameters.

| Test Set | Avg. Profit | Sharpe | MSR |
|---|---|---|---|
| GAASR | 725 | 0.552 | 1.518 |
| GPASR | 360 | 0.335 | 0.659 |
| **GAAMSR** | **2231** | **0.85** | **5.007** |
| GPAMSR | 710 | 0.509 | 2.196 |
| MM | 212 | 0.84 | 2.574 |
| SMM | 1561 | 0.807 | 2.67 |

Overall, the MSR tuned versions of the evolutionary agents perform very well and are extremely competitive with the other leading agents. The GAA strategy is consistently more profitable than

the GPA, potentially owing to the design of GPA that needs a larger pool of indicators to choose from. From the rigorous tests in this section, we conclude that our choices of design parameters for this agent are justified. We have achieved the elusive combination of consistency and high Sharpe and Sortino ratios using these automated agents, and in the event, these results demonstrate a small but significant result for automated trading agents.

## 11. DISCUSSION AND FUTURE WORK

The motivation for this work is twofold, (1) to use genetic algorithms to optimally design composite trading rules that are robust in varying market conditions and (2) to search for principles to guide the design of safe, composite rules for autonomous trading agents. Some key observations that emerged from our work are listed below:

1. Not only can trading rules be combined together to achieve profitable trading, but, in fact, the composite trading rule can be structured in such a way that it becomes possible to reason about safety and performance. This is something that our proposed methodology adds to the existing literature on trading agent design using evolutionary algorithms.

2. The fitness function used to tune an optimization algorithm has a great effect on the profitability of the agent. When we introduced the modified Sortino ratio (and thereby allowed for unbounded positive volatility), profit margins increased, and trading performance increased. This is also in accordance with observed preferences of human traders, as noted earlier.

3. Since the fitness function and the performance criteria were the same in our designs, the choice of performance criteria to match our objectives, is crucial to the development of agents with increased performance.

A potential challenge to easily extensible models based on the design presented above, is the computation time. Training the agent involves simulating numerous, complete trading days. The time taken for a training run for as few as 12 generations was 2 hours and 30 minutes on a Pentium-4, 1 GHz machine with 256 MB of RAM. Further complexity will require training the agents for longer periods of time. Possible solutions to this problem include limiting the interaction of training data and the agent to quickly simulate the trading activity, and modeling this interaction numerically in such a way that the training activity can be carried out without necessitating complete trading simulations. The tests in this work are confined to a single market. The use of further sophisticated fitness functions as well as better representations of the candidate solutions will only work to steadily improve the performance of agents designed using the proposed principles, and the techniques introduced here may be extended for tests in multiple markets.

## 12. CONCLUSION

Human trading behavior involves a fair amount of flexibility and adaptability to the uncertainty of market conditions. Practicing traders are able to use sophisticated combinations of simple rules to arrive at robust decisions. We have tried to emulate this behavior in our agent designs. Our approach to solving the problem of profitable automated stock trading achieves this using a combination of genetic algorithms to optimize and re-adjust the trading strategy; and qualitative information to provide safety guarantees regarding the agents' behavior. Although the ideas are presented and evaluated in the context of specific agent architectures, the principles we have adopted are fairly general and they are easily extensible to other designs. Any complex trading rule or set of rules, if it can be qualitatively characterized, can be substituted into our evolutionary agent design. This provides us with a sound platform from which to produce further, more sophisticated strategies that are ultimately more profitable.

## 13. ACKNOWLEDGEMENT

This research was supported in part by NSF CAREER award IIS-0237699.

## 14. REFERENCES

[1] E. Acar and S. Satchell. Advanced Trading Rules. *Butterworth-Heinemann*, 2002.

[2] E. Acar and S. Satchell. A theoretical analysis of trading rules: an application to the moving average case with markovian returns. *Appl. Math. Finance*, 4:165–180, 1997.

[3] F. Allen and R. Karjalainen. Using genetic algorithms to find technical trading rules. *J. of Financial Economics*, 51:245–271, 1999.

[4] M. Dempster and C. Jones. A real-time adaptive trading system using genetic programming. *Quantitative Finance*, 1:397–413, 2001.

[5] M. Dempster and C. Jones and Y. Romahi and G. Thompson. Computational Learning Techniques for Intraday FX Trading Using Popular Technical Indicators. *IEEE Transactions on Neural Networks*, 12(4), 2001.

[6] E. Fama. Efficient Capital Markets: A review of theory and empirical work. *J. Finance*, 25(2):383–417, 1970.

[7] R. Gencay. An introduction to High-Frequency Finance. *Academic Press*, 2001.

[8] T. Hellstrom and K. Holmstrom. Parameter tuning in trading algorithms using ASTA. *Computational Finance*, 1:343–357, 1999.

[9] M. Kearns and L. Ortiz. The Penn-Lehman automated trading project. *IEEE Intelligent Systems*, 18(6):22–31, 2003.

[10] J. Koza. Genetic Programming: On the programming of computers by means of natural selection. *MIT Press*, 1992.

[11] A. Lo and A. Craig MacKinlay. A Non-Random walk down Wall Street. *Princeton University Press*, 1999.

[12] B. Malkiel. A Random Walk down Wall Street. *WW Norton*, 1996.

[13] J. Moody and M. Saffell. Learning to trade via direct reinforcement. *IEEE Trans. Neural Networks*, 12(4):875–889, 2001.

[14] S. Neftci. Naive trading rules in financial markets and wiener-kolmogorov prediction theory: A study of"technical analysis". *J. Business*, 64(4):549-571, 1991.

[15] C. Pendersen. Derivatives and downside risk. In *Derivatives Use, Trading and Regulation*, 2001.

[16] W. Sharpe. The sharpe ratio. *J. Portfolio Management*, 21(1):49–58, 1994.

[17] A. Sherstov and P. Stone. Three automated stock-trading agents: A comparative study. *AAMAS 2004 Workshop on Agent Mediated Electronic Commerce VI*, 2004.

[18] H. Subramanian. *Evolutionary algorithms in optimization of technical rules for automated stock trading*. M.S. Thesis, University of Texas, Austin, 2004.

[19] M. Taylor and H. Allen. The use of technical analysis in foreign exchange markets. *J. Int. Money Finance*, 11:304–314, 1992.