# Portfolio Allocation using XCS Experts in Technical Analysis, Market Conditions and Options Market

Sor Ying (Byron) Wong
School of Informatics
University of Edinburgh
+852 8122 9630
sorying.wong@gmail.com

Sonia Schulenburg
Level E Limited, ETTC, The King's Buildings,
Mayfield Road, Edinburgh EH9 3JL, UK
+44 (0)131 472 4807
sonia@levelelimited.com

## ABSTRACT
Schulenburg [15] first proposed the idea to model different trader types by supplying different input information sets to a group of homogenous LCS agent. Gershoff [12] investigated this idea further with XCS agent. This paper takes an extra step to build a trading system that not only adopts the multi-XCS agent idea, but also utilizes knowledge from discretization theory, modern portfolio theory, options theory and methods of combining multiple models. In comparison to previous work, a wider range of input data were used including technical analysis, general market conditions and options market conditions. Secondly, quantization of continuous financial series was achieved using entropy-based discretization and histogram equalization. Thirdly, subtle investment strategies can now be generated as a result of taking stock price magnitude into account. Finally, multiple agents' predictions were combined using a variant of stacking. Empirical results show the best-performing XCS agents always outclass benchmark agents in every stock examined. Variance is reduced after combining predictions from multiple models. The technical analysis XCS agent was able to replicate a well known technical trading rule widely used in the 60s.

## Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning – *Concept learning, Parameter learning.*

## General Terms
Economics, Experimentation

## Keywords
Learning Classifier System, XCS, Reinforcement Learning, Computational Finance, Options, Economics, Finance.

## 1. INTRODUCTION
Chen [7-9], Gershoff [12], Schulenburg [15-17], Stone [20] etc have applied learning classifiers systems to different financial contexts such as the foreign exchanges market, the derivatives market and the equity market. While their results were promising, it is hypothesized that better results can be achieved by improving processes which surround the main XCS learning component.


*GECCO'07*, July 7–11, 2007, London, England, United Kingdom.


Previous work approached the problem in various ways, but they all share some common characteristics. First, technical indicators were primarily used as the input series. This immediately limits the input space to solely historical price and volume information. Is it possible to use a wider range of input data to avoid weak-form market efficiency? Second, instead of quantizing financial series using user pre-specified conversion rules, is there an automatic quantization process that could make the most of currently available data? Third, previous work typically only produces buy or sell prediction. Is it possible to predict expected price movement instead such that a continuum of trading strategy can be applied? Following from that, how should the reward structure change to provide more precise payoff? Finally, is there a good way to combine multiple predictions in on-line learning problem?

The goal of this paper is to re-investigate the XCS-based trading system by explicitly addressing these issues. Knowledge from discretization theory, modern portfolio theory, option theory and methods of combining multiple models was used to solve the above issues. Specifically, 4 technical indicators, 3 market condition indicators and 5 options market indicators were used as the input series, as opposed to only technical indicators. The quantization mechanism was based on histogram equalization and entropy-based discretization, rather than a set of pre-specified conversion rules. The target series agents predict was the expected price movement, instead of a buy or sell signal. A simple variant of stacking was adopted to combine multiple predictions into a meta-prediction.

This paper presents empirical results for XCS agents who learn switching strategies between a risky stock and Treasury Bills. These are compared against 4 benchmark agents: Buy & Hold, Bank, Price Trend and Random agents. The results demonstrate the presence of predictable structure in the US stock prices over a 10-year period. Best-performing XCS agents performed substantially better than benchmark agents, and a set of potentially profit-making trading rules were uncovered. One of the rules was found to be a famous technical trading rule widely used in Wall Street in the 60s.

## 2. BACKGROUND
Investment decisions can usually be broken down briefly into two steps. The first step is to select a list of companies which have growth potential or currently being undervalued. This process is called *portfolio selection*. The second step is to investigate the stock price movements of each selected company, and execute correct trading strategies at appropriate timing. Generative paradigms like Bayesian networks have been applied to portfolio selection problems in several contexts [21], [18], [19], [2].

Discriminative paradigms, by contrast, are usually used for price movement prediction [1, 7-9, 13, 15-17]. One possible explanation for such categorization is the challenges faced by Bayesian networks to estimate the true probability densities for non-Gaussian continuous variables. Discriminative paradigm models the input-output mapping directly, and has good records of modeling time-series data. Thus, it is widely used for solving price prediction problem.

Shenoy and Shenoy [18, 19] applied Bayesian networks to portfolio risk and return. They demonstrated how financial analysts can include their expert subjective judgments into the Bayesian network model. The output of their model is the posterior marginal probability distribution of the portfolio return. Therefore it can be used to obtain different risk-related quantities such as the expected return, return variance, and value-at-risk.

Beltrametti, Fiorentini, Marengo and Tamborini [3] were one of the earliest groups who applied LCS to financial markets. To investigate the behavior of the foreign currency market, they tested the data from Dollar-Mark and Dollar-Yen exchanges. The input of the LCS agent was derived endogenously from a macroeconomic model, rather than exogenous sources. The result was not very encouraging especially after taking transaction costs into account, but it did inspire the community to research agent-based method to model the financial markets.

Schulenburg and Ross [15-17] used multiple LCS agents to search different parts of the problem spaces by giving each individual agent different sets of information. They tested their model with 8 stocks from different industries, all showing promising results. In particular, with initial wealth of $10,000, some agents were able to generate more than 10 folds of return in 15 years.

Chen, Tseng, Cheng, Chang and Tsai [7-9] have applied XCS into several derivatives markets. Their first trial was a basic implementation of XCS applied to the Taiwan Future Exchange Market Index using standard technical indicators as input variables. They then applied the XCS model to the S&P Futures market using contrary sentiment indicators as input variables: volatility index, put-call ratio and trading index. In their later work, they applied XCS to intra-day data of Taiwan Index Option. The model predicted the price fluctuation of next 10 minutes. With the help of several empirical studies on relationships among several seemingly unrelated variables, the results showed that the average accuracy ratio of predicting the option trend for the next 10 minutes is more than 70% and the annual accumulative profit ratio is at least 50%.

# 3. SYSTEM ARCHITECTURE

A key quantity the trading system predicts is the percentage of current total wealth to invest in stock market. This is closely related to the expected change of tomorrow price. There are 3 types of XCS agents giving predictions of future price movements based on different sets of historical financial data. Their estimates are combined to form a more robust meta-estimate. This meta-estimate, together with the volatility of stock price and the user's risk preference, computes the optimal percentage to invest in stock using Modern Portfolio Theory.

The accuracy of agents' predictions depends largely on how well the problem is represented. The following sections will describe our system design.

## 3.1 Variable selection

Huge problem space is a classic problem in machine learning. A possible solution is to split the input series into smaller, *inter-independent but intra-dependent* subsets, allowing multiple XCS agents to learn *well* locally and form *better* combined prediction globally. Gershoff [12] demonstrated that this multi-agent approach was far better than a single agent approach under a clean room experimental setting where artificial data were used. Schulenburg [15-17] also showed promising results when applying multi-agent method to LCS agents.

Dividing input variables into smaller group implicitly assumes inter-group relationships are independent. While this is hard to achieve precisely, a good split can generally be done by using prior expert knowledge. In our case, input series were classified into three main categories:

*Technical Analysis (TA)* consists of Rate of Change (ROC), Relative Strength Index (RSI), Ultimate Oscillator (ULTOSC) and On Balance Volume (OBV).

*General Market conditions (Mkt)* consists of daily return % of S&P 500 Index, daily S&P 500 Index volume, daily 10 year T-note bond yield and daily 3 month T-bill bond yield.

*Options Market conditions (Options)* consists of Delta, which measures sensitivity of option value to underlying stock price; Gamma, which measures second order sensitivity of option value to underlying stock price; Vega, which measures sensitivity of option value to stock price volatility; Theta, which measures sensitivity of option value to passage of time; Implied Volatility, which is a stock volatility estimate derived from the Black-Scholes formula[1].

## 3.2 Discretization

The implementation of system was built on top of Butz's XCSJava [6], which is based on the original XCS framework proposed by Wilson in 1995 [22]. The agent interacts with the environment through a state binary vector that represents *discrete* environment states. To apply this to the highly continuous-value based financial domain, the system must first quantize the input continuous series into discrete intervals.

Previous work usually selected the conversion mechanism manually i.e. choosing several discrete cut points simply by prior knowledge. This causes two problems: first, it poses the risk of losing important information during conversion if the cut points are not chosen appropriately. For example, it probably does not help to choose a cut point at 0 when all data points are positive. Second, in a highly dynamic environment like the stock market, the range of the continuous series changes as time progresses. Thus, even a set of good discrete cut points have been chosen initially, as the range of continuous series shifts, this set of cut points becomes obsolete. To address these problems, an automatic discretization process must be adopted. The process should ensure information is being preserved during conversion, and discrete cut points should get updated as the market regime changes.

[1] The formula is very complex, and so it's not reproduced here. Interested reader should visit Black-Sholes at Wikipedia.

In the absence of class label information, the target series (daily return of stock price) can only be quantized using unsupervised discretization. A desired number of discrete intervals must first be chosen. By optimizing the number of bins using leave-one-out cross-validation, the number of discrete cut points was set to be 9. *Histogram equalization* was used to quantize the target series.

After quantizing the target series, each continuous input series now has a corresponding class label. So, supervised discretization like entropy-based discretization can be applied. *Entropy-based discretization* is a method that quantizes a continuous series by repeatedly splitting the series into intervals which maximizes the information gain.

For complete details of the two discretization processes, please refer to Dougherty, Kohavi and Sahami's paper [10]. The trading system utilizes discretization library from Weka [23].

## 3.3 From discrete intervals to state vector

After quantizing data into discrete values, building the binary vector should be straight forward. For an attribute that has *n* possible intervals, a binary vector of length *n-1* is needed. The first *i-1* digits are set to false whenever the $i^{th}$ interval is present in the data, and true otherwise. The remaining attributes are set to false [23]. In other words, the $(i-1)^{th}$ binary attribute represents whether the attribute is less than or equal to interval *i*. For example,

**Table 1: discrete to binary vector using n-1 encoding**

| Interval | → | Binary |
|---|---|---|
| 0-50 | → | 0000 |
| 50-53 | → | 1000 |
| 50-53 | → | 1100 |
| 57-60 | → | 1110 |
| 60-infinity | → | 1111 |

Mutation and crossover in the XCS Genetic Algorithms component were modified to prevent illegal binary string like 0111. Whenever invalid strings are produced, they will be discarded and new strings will be re-generated.

## 3.4 XCS Agent

Three XCS agent types are defined based on the information set described in section 3.1. They are the only learning / reinforcement components. All agents have the same goal: try their best to predict the price movement of tomorrow stock price using their available input information set. They first train themselves by *exploring* all available *historical* data. During this phase, no exploitation is performed.

After this initial training phase, agents enter the trading phase and are asked to apply what they have learnt to *unseen* real world scenarios --- they are asked to give predictions on tomorrow stock price movement based on input information of today. These predictions are combined to form a single estimate using the result combiner (section 3.7), and this single estimate is sent to the Portfolio Optimizer (section 3.8).

The next day, the actual stock price movement is observed. Agents receive their rewards based on their prediction accuracies, and these rewards update the internal parameters of agents' reinforcement components. At this point every day, the cumulative performance of the meta-agent is evaluated. If the cumulative prediction accuracy is lower than a pre-specified threshold level, *all* agents, including the meta-gent itself, will be killed.

If that's the case, a new set of agents along with a new discretization process will be launched. The new set of discrete cut points are based on a small set of most recent data points (currently this is set to be the last 10 days). All previously seen data are then quantized using this new set of discrete cut points. All new agents start their training phases by *exploring* themselves into this new training environment. The hope is to create a new set of agents such that their collective intelligence is better than the old, suicide meta-agent. Once all new agents finish training, they will be placed back to the current real world environment and the above process iterates.

If the meta-agent is performing well (the cumulative prediction accuracy is greater than the pre-specified threshold level), all agents will be asked to predict the *next* stock price movement. The above process then repeats.

Only exploration is done during training phase, whereas in trading phase, exploitation is mainly performed. Occasionally, agents explore with probability $\frac{0.5}{\sqrt{iteration+1}}+0.05$ in trading phase, however resulting predictions are not presented to the Portfolio Optimizer (section 3.8), i.e. only agents' internal parameters are updated but no actual trading is executed. This is consistent to our intuitions --- a person can always guess and learn from the stock price movement without actually spending money trading it.

Figure 1 to Figure 4 shows an agent's life span graphically.

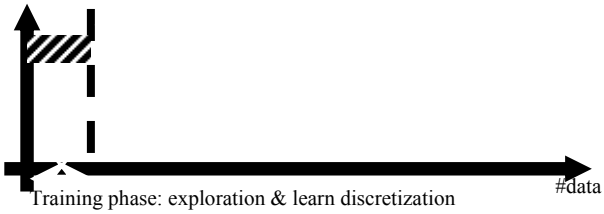


**Figure 1. Agent is in its initial training phase, where exploration is mainly performed**


**Figure 2. Agent finishes training phase and is now in trading phase. Agent mainly exploits but occasionally it explores.**

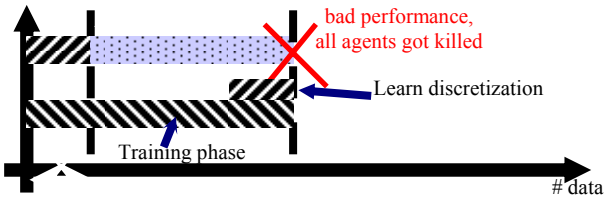


**Figure 3. Collective performance is evaluated at every time step. If its accuracy is lower than a pre-specified threshold level, all agents will be killed. A new set of agents is created. A small set of most recent data were used to learn discrete cut points. All historical data were then quantized using this set of new cut points, which serve as the training set for new agents.**

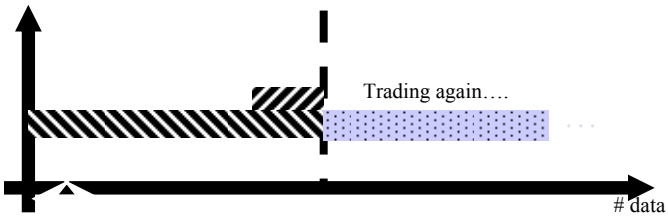


**Figure 4. After the new training phase, agent starts trading again… process repeats.**

## 3.5 Dominant Class problem

As mentioned earlier in section 3.2, change of market regime deteriorates the quality of the discrete cut points. A set of intervals which is perfect for some data is not necessarily well suited for another. The consequence of this is: a majority of new data will be converted to only a few discrete intervals. We call this the dominant class problem.

If dominant class exists, a learning algorithm can easily achieve high accuracy by blindly guessing the majority class. As an extreme example, assume there are only 2 discrete intervals (0-50 and 50-100). If 99% of the data are classified as 50-100, then even the simplest learning algorithm can achieve 99% accuracy by repeatedly classifying data to 50-100. Hence, even the prediction accuracy is extremely high, the prediction itself isn't informative.

A solution to this problem is to check if dominant class exists in the target variable series at every time step. If such class exists, the current set of discrete intervals should be replaced by a new set of intervals which based on more recent data. After modifying the discrete cut points, however, all rules previously learnt by agents are no longer valid. Hence, modifying the current set of discrete interval inevitably kills the current set of agents.

A class is defined to be dominant when there are more than (2)(# of data)/(# of intervals) data points in one discrete interval.

## 3.6 Reward function

The usual way to define reward function is to assign 0 for incorrect prediction and a positive value (say 1000) for correct prediction. In this system, however, agents predict tomorrow price movement. Hence, a new reward structure is introduced to capture the quality of predictions. First, *the difference between the true and predicted value* should be taken into account. To capture this effect, reward is defined to be the negation of the squared error.

Second, *the importance of the prediction* should be taken into account. A wrong prediction when the true price change is -10% definitely costs more than a wrong prediction when the true price change is -0.1%! To incorporate this notion of critical prediction, reward is doubled or halved (corresponding to a correct or incorrect guess) when the prediction is considered crucial, which is defined to be true if actual price change is greater than 3%.

## 3.7 Result Combiner

In order to reduce the variance of the result, it is very important to combine predictions from technical analysis agent, general market condition agent and the options market agent. In fact, 5 identical agents for each agent type were created, each having different initial random seeds. The intention here is: by introducing more agents with different random seeds, their collective predictions can hopefully be more robust and reliable.

There are several methods to combine multiple models' outputs to a final one, such as bagging [5], boosting [11] and stacking [24]. Here, a variant of stacking was used. Combining predictions is essentially asking the question: "Which agent should we trust more?" An intuitive answer is to trust agents who have been performing well historically. One way to measure historical performance is to examine the *total wealth* of agent at this point. Thus,

$$collective\_prediction_t = \sum_i \frac{wealth_{t-1}(i)}{\sum_j wealth_{t-1}(j)} \times prediction_t(i)$$

## 3.8 Portfolio Optimizer

The result combiner produces an estimate of tomorrow expected price return, what could the trading system do to utilize this quantity? From the Modern Portfolio Theory [4], if we wish to allocate wealth between a risky and a risk free investment, the optimal ratio of holding the risky investment, $w^*$, is give by

$$w^* = \frac{E(R_{risky}) - R_{free}}{0.01 A \sigma^2_{risky}}$$

where $E(R_{risky})$ is the expected return of risky investment; $R_{free}$ is the return of risk free investment; $\sigma^2_{risky}$ is the variance of risky investment; $A$ is the investor's revealed preference for risk. Treating XCS agents' combined prediction as an estimate of $E(R_{risky})$, 3 month T-bill rate as the risk free investment rate $R_{free}$, implied volatility from the options market data as $\sigma_{risky}$, $w^*$ can then be computed. Hence, by applying Modern Portfolio Theory, the system recommends a percentage of the total wealth which should be invested in the stock market every day.

## 3.9 Summary of system flow

Figure 5 shows a summary of the system flow.

Three XCS agent types were used in this system: technical analysis, general market condition and options market condition. They differ from one another in their input information set. Every agent's goal is to predict tomorrow's stock price movement, i.e. the percentage change of the stock price at time $t+1$.

Continuous input series must be quantized in order to construct the environmental state vector for XCS. Histogram equalization is applied to the target series, whereas entropy-based supervised discretization is applied to all input series.

Every XCS agent has to go through both training and trading phases. During trading phases, agents kill themselves if their collective performance is lower than a pre-specified threshold level. A new set of agents as well as a new set of discrete intervals are created. Only exploration is performed during learning phase, whereas in trading phase, exploitation is mainly executed.

Predictions from multiple agents are combined to one single meta-prediction using weightings derived from agents' historical performance. This meta-prediction is then transmitted to the Portfolio Optimizer. The optimizer computes the optimal percentage of an investor's total wealth to invest in stock market. Following the system recommendation, the investor's portfolio is adjusted accordingly.

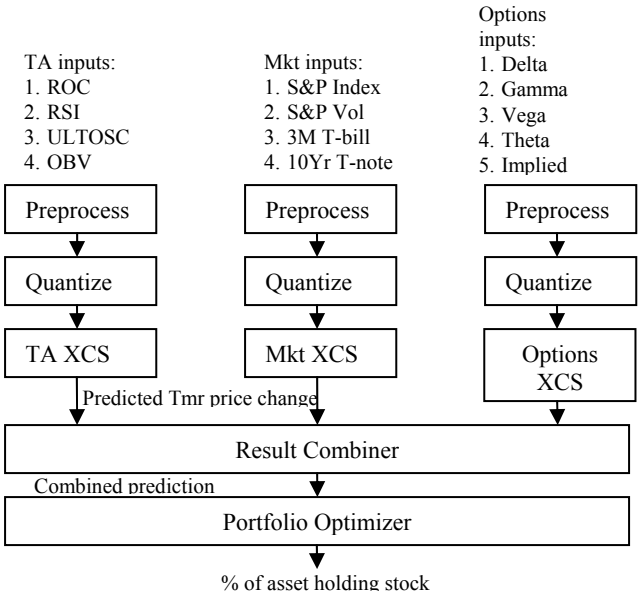


**Figure 5. The complete system flow**

# 4. EXPERIMENTAL RESULTS

## 4.1 Setting

To assess the system performance under a full range of market phenomena such as bubbles, crashes, drastic trading volume change etc, 5 stocks from different industries were used. They are Citigroup (C), IBM (IBM), General Motors (GM), Eastman Kodak (EK) and Exxon Mobil (XOM). Data were collected from 4 January 1996 to 28 April 2006. Initial cash was set to be $10,000. Commission was set to be 0.5% of transaction value. The risk free interest rate varies according to real world interest rate movements.

Four benchmark agents were implemented. Buy and Hold agent invests all its money into stocks on day 0, holds the shares and sells them at the end of the period. Bank agent deposits all its money into the bank on day 0, leaves it aside and withdraws both principal and interests at the end. Random agent makes random buying or selling decision everyday. Price Trend agent forms its prediction of tomorrow stock price movement from yesterday stock price movement.

## 4.2 Results

Table 2 shows final wealth of all best-performing agents.

**Table 2: final wealth of all best-performing agents**

| | XCS agents | | | | Benchmark agents | | | |
|---|---|---|---|---|---|---|---|---|
| | TA (k) | Mkt (k) | Options (k) | Meta (k) | Rdm (k) | B&H (k) | Bank (k) | Price Trend (k) |
| C | 89 | 134 | 130 | 98 | 30 | 64 | 13 | 60 |
| IBM | 70 | 52 | 44 | 38 | 51 | 35 | 13 | 23 |
| GM | 36 | 46 | 39 | 14 | 27 | 9 | 13 | 9 |
| EK | 34 | 29 | 27 | 9 | 2 | 5 | 13 | 15 |
| XOM | 41 | 63 | 55 | 45 | 40 | 40 | 13 | 8 |

**Table 3: percentage of sample population which each agent type was able to outperform the Buy and Hold agent.**

| | TA (%) | Mkt (%) | Options (%) | Meta (%) | Rdm (%) |
|---|---|---|---|---|---|
| C | 11.2 | 15.2 | 6.4 | 28 | 8 |
| IBM | 29.6 | 26.4 | 21.6 | 24 | 20 |
| GM | 58.4 | 64.8 | 61.6 | 40 | 64 |
| EK | 86.4 | 82.4 | 78.4 | 40 | 84 |
| XOM | 1.6 | 4.8 | 4 | 4 | 8 |

Results of best-performing agents in 125 runs are shown instead of average values. This counter-intuitive presentation layout is more appropriate because the most important output from the system is a set of if-then rules that generates high returns. This piece of information can only be obtained from the knowledgebase of the best-performing agent. That is, as long as one of the agents is Warren Buffet, no matter how awful other agents perform (and how terrible the average performance is), the system has produced the most important piece of information --- Buffet's brain.

Table 3 shows the percentage of the sample 125 experimental trails in which each *non-deterministic* agent type was able to outperform the Buy and Hold agent. For example, (11.2%)(125) = 14 technical analysis agents outperformed the Buy and Hold agent in 125 trails. The higher the number, the better these 125 technical analysis agents performed. The figure gives a rough idea of the number of experimental trails required to produce a representative set of population. For instance, if an agent type has a low percentage value (say 1%), it would not make sense to regard the best-performing agent in *a few experimental trails* (say 10) as "the true" best-performing agent. That is, this percentage figure gives a sense of how representative the best-performing agent of our 125 trails is with respect to "the true" best-performing agent of this type. Of course, it also gives an idea on how consistent this agent type outperformed the Buy and Hold agent.

Figure 6 (on page 8) shows agents' wealth against time for Citigroup's stock. The horizontal axis shows the number of trading days so far and the vertical axis shows the current wealth in dollars. The linear rising line at the bottom represents the

wealth of bank agent. The brown line represents the wealth of the Buy and Hold agent, which can also be viewed as Citigroup's stock price movement in a scaled fashion.

Random agent performed poorly, probably because the stock price exhibited a general upward momentum rather than volatile ups and downs. Thus, there was a higher probability for price change to be positive than negative, making strategy which predicts the ups and downs with equal probability very hard to survive.

Price trend agent performed satisfactorily, thanks to the general upward tendency of Citigroup's stock price. It has outperformed Buy and Hold agent for most of the time except at the end.

All XCS agents outperformed all benchmark agents. In particular, Options and Market agents have both beaten the market by large margins, i.e. they made twice as much as Buy and Hold agent did. TA agents also outperformed Buy and Hold agent for more than 40%. All XCS agents surpassed all benchmark agents significantly from day 700 to day 1200 when Citigroup's stock price is increasing a lot. Similar phenomenon is observed from day 1800 onwards.

The meta-agent performed better than all benchmark agents, but worse than both Market and Options agents. However, it has the highest percentage of sample population that beat the Buy and Hold agent. Meta-agent in general performs moderately but consistently. It did reduce the standard deviation of the result. For example, in the 125 experimental trails of Citigroup's stock, the standard deviations of non-deterministic agents are:

| TA ($) | Mkt ($) | Options ($) | Meta ($) | Rdm ($) |
|---|---|---|---|---|
| 22,372 | 25,831 | 23,362 | 17,847 | 25,052 |

Meta-agent has the lowest standard deviation among all other non-deterministic agents. If our goal is to create an artificial stock market with agents having similar capabilities, the meta-agent will be a good choice to start with.

In general, combining different models' predictions is not an easy task. Indeed, this is essentially another learning problem in an upper level --- meta-agent has to decide which lower level agent should it trusts and by how much. Most well-known solutions which combine multiple models only apply to offline learning algorithms in which a clear notion of training set and validation set is present. For online learning algorithms such as this one, it is still unclear what the best way to approach this problem is.

In volatile market condition, random agent generally performs a lot better than other benchmark agents, probably due to the non-deterministic nature of the agent type. Stock price, to some extent, follows a random walk. Hence, it is entirely possible for random strategy to make "easy" money when stock market exhibits highly volatile behavior. Nonetheless, stock price does have its rational component that depends on non-random factors. Discovering these factors is not easy but doesn't mean it is impossible. The fact that *all* XCS agents outperformed *all* random agents in every experiment is a solid proof of the existence of stock price's rational component[2]. If the stock market were truly random, consistent victories are almost impossible.

Another encouraging fact is that a rule XCS agents discovered[3] when learning IBM stock pattern turned out to be a famous technical trading rule widely used in Wall Street in the 60s [14].

## 4.3 Results for a declining stock

Citigroup's stock has been increasing over the past 10 years. It's worth investigating how the system performs under a bad investing period. Figure 7 (on page 8) shows the experimental results of a declining stock, Eastman Kodak.

With the emergence and widespread adoption of digital technologies in the photography business, Kodak's traditional line of products such as cameras, films etc are being marginalized. The company has undergone several restructuring in the past 10 years, causing significant decline in its stock price. In particular, Kodak's stock price has been decreasing since day 626 and was almost halved at the end of period. This big price drop killed the Buy and Hold agent whose wealth is directly proportional to stock price.

Price trend agent performed satisfactorily during this bad time, thanks to the clear downward tendency of Kodak's stock.

Random agent, who typically performs badly in trendy period, surprisingly performed well this time. At one point (day 920), it was the best-performing agent among all.

Results of all XCS agents are encouraging. After day 646, they all started increasing their wealth significantly even Kodak's stock price kept decreasing. Although later movements seemed to be a bit jerky, they all tended upwards. It is worth noting the great performance of XCS agents when the stock prices have obvious downward trends (as in EK and GM). Agents are usually able to spot the downward trends at early stage and make right investment decisions.

# 5. CONCLUSION

This paper starts with the multi-trader modeling approach discussed by Schulenburg [15] and Gershoff [12]. By incorporating techniques from other domains to address the 5 issues mentioned in introduction, XCS agents are able to perform better than benchmark agents. Variance is also reduced after combining predictions from multiple models.

It's worth mentioning the internal knowledgebase of all best-performing agents captures some form of potentially profit-making classifiers. Some of them are just false patterns, and some of them do not make sense economically even they fit the data well. Nevertheless, it is a good start for financial analyst to discover true market pattern from this reduced set. In our experiment, one of the rules was found to be a famous technical trading rule widely used in Wall Street in the 60s.

[2] Only 0.00003% would this observation happen by chance.

[3] "If ultimate oscillator > 70 and previous stock price change is within 2 to 3%, then tomorrow stock price will be in -2.5 to -3.5%"

## 6. REFERENCES


[1] G. Armano, A. Murru, and F. Roli, "Stock market prediction by a mixture of genetic-neural experts," *IJPRAI,* vol. 16, pp. 501-526, 2002.

[2] A. F. Atiya and M. Magdon-Ismai, "A Bayesian Approach to Estimating Mutual Fund Returns," in *Computational Finance 1999*, Y. S. Abu-Mostafa, B. LeBaron, A. W. Lo, and A. S. Weigand, Eds. Cambridge, MA: The MIT Press, 2000.

[3] L. Beltrametti, R. Fiorentini, L. Marengo, and Tamborini, "A learning-to-forecast experiment on the foreign exchange market with a classifier system," *Journal of Economic Dynamics and Control,* vol. 21, pp. 1543-1575, June 1997 1997.

[4] Z. Bodie, A. Kane, and A. J. Marcus, *Investments*, 5th ed. New Delhi: Tata/McGraw-Hill, 2002.

[5] L. Breiman, "Bagging predictors," *Machine Learning,* vol. 24, pp. 123-140, 1996.

[6] M. V. Butz, "XCSJava 1.0: An implementation of the XCS classifier system in Java," Illinois Genetic Algorithms Laboratory 2000.

[7] A. P. Chen and Y. H. Chang, "Using extended classifier system to forecast S&P futures based on contrary sentiment indicators," *Evolutionary Computation,* vol. 3, pp. 2084 - 2090, 2005.

[8] A. P. Chen, Y. C. Chen, and W. C. Tseng, "Applying Extending Classifier System to Develop an Option-Operation Suggestion Model of Intraday Trading-An Example of Taiwan Index Option," *Lecture Notes in AI,* vol. 2681, pp. 27-33, January 2005 2005.

[9] T. W. Cheng, W. C. Tsai, and C. A..P., "Strategy of futures trading mechanism using extended classifier system," in *The 2nd International IEEE Conference on Intelligent Systems*, 2004, pp. 503-507.

[10] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and Unsupervised Discretization of Continuous Features," in *International Conference on Machine Learning*, 1995, pp. 194-202.

[11] Y. Freund and R. R. Schapire., "Experiments with a new boosting algorithm," in *Proc. Thirteenth International Conference on Machine Learning* Bari, Italy: Morgan Kaufmann, 1996.

[12] M. Gershoff, "An Investigation of HXCS Traders," in *School of Informatics*. vol. Master of Sciences Edinburgh: University of Edinburgh, 2006.

[13] G. Kendall and Y. Su, "Learning with Imperfections – A Multi-Agent Neural-Genetic Trading System with Differing Levels of Social Learning," in *2004 IEEE Conference on Cybernetics and Intelligen Systems*, Singapore, 2004.

[14] MHP, "The Ultimate Oscillator - Technical Analysis."

[15] S. Schulenburg and P. Ross, "An Adaptive Agent Based Economic Model," in *Learning Classifier Systems: From foundations to Applications, Lecture Notes in Artificial Intelligence*. vol. 1813, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, Eds. Berlin: Springer-Verlag, 2000.

[16] S. Schulenburg and P. Ross, "Strength and Money: An LCS Approach to Increasing Returns," in *Learning Classifier Systems: From foundations to Applications*. vol. 1996 of Lecture Notes in Artificial Intelligence, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, Eds. Berlin: Springer-Verlag, 2001.

[17] S. Schulenburg and P. Ross, "Explorations in LCS Models of Stock Trading," in *Learning Classifier Systems: From foundations to Applications*. vol. 2321 of Lecture Notes in Artificial Intelligence, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, Eds. Berlin: Springer-Verlag, 2002.

[18] C. Shenoy and P. P. Shenoy, "Bayesian Network Models of Portfolio Risk and Return," in *Computational Finance 1999*, Y. S. Abu-Mostafa, B. LeBaron, A. W. Lo, and A. S. Weigand, Eds. Cambridge, MA: The MIT Press, 2000.

[19] C. Shenoy and P. P. Shenoy, "Modeling Financial Portfolios Using Belief Functions," in *Belief Functions in Business Decisions*, R. P. Srivastava and T. J. Mock, Eds. Heidelberg: Physica-Verlag, 2002.

[20] C. Stone and L. Bull, "Foreign Exchange Trading using a Learning Classifier System," University of the West of England Bristol, Bristol United Kingdom 2004.

[21] C. C. Tseng, P. J. Gmytrasiewicz, and C. Ching, "Refining Influence Diagram For Stock Portfolio Selection," Society for Computational Economics, Computing in Economics and Finance 2001 2003.

[22] S. W. Wilson, "Classifier Fitness Based on Accuracy," *Evolutionary Computation,* vol. 3, 1995.

[23] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[24] D. H. Wolpert, "Stacked generalization," *Neural Networks,* vol. 5, pp. 241-259, 1992.

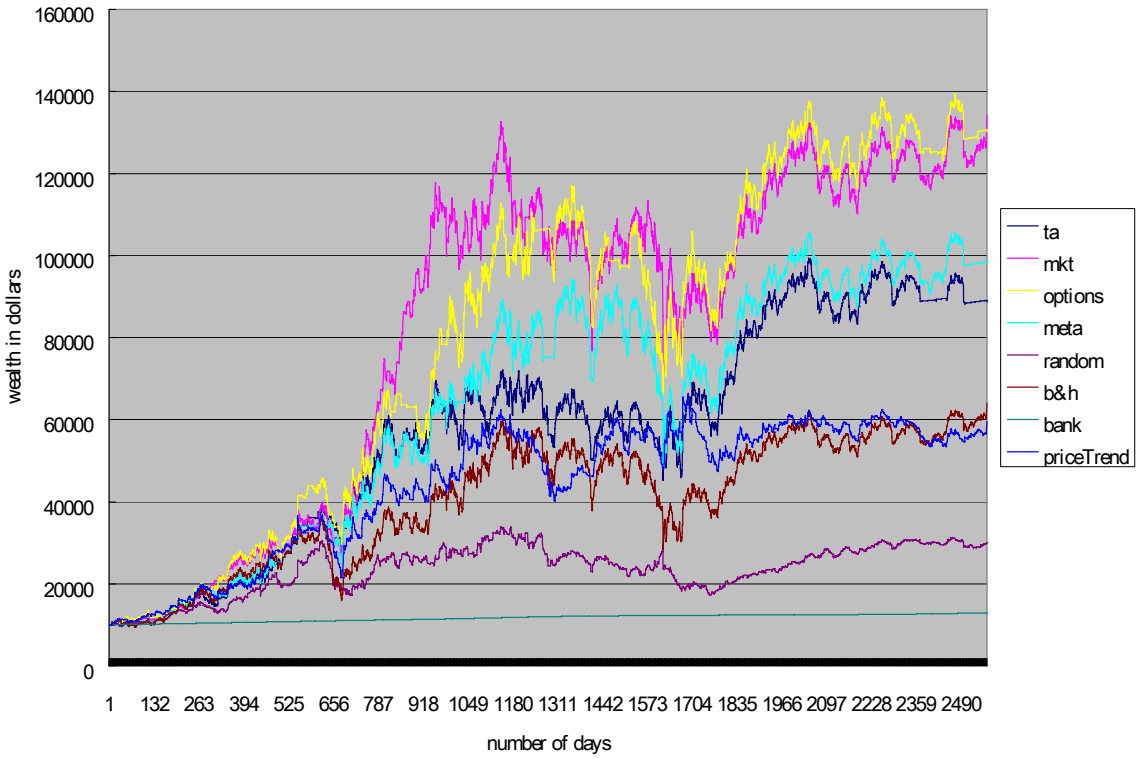


**Figure 6. agents' wealth against time for Citigroup's stock**

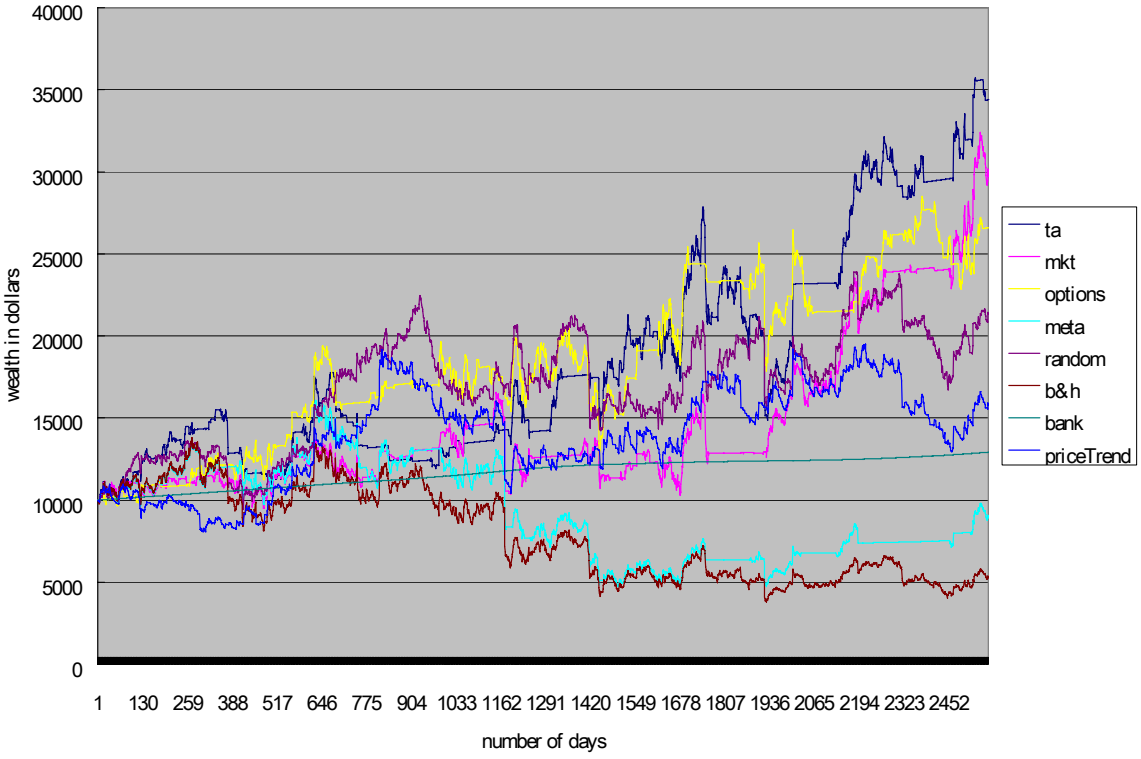


**Figure 7. agents' wealth against time for Kodak's stock**