A Hybrid Neuro-Genetic Approach For Stock Forecasting

Yung-Keun Kwon [*] and Byung-Ro Moon [†]
School of Computer Science and Engineering
Seoul National University
Shilim-dong, Kwanak-gu
Seoul, 151-742 Korea

Abstract

In this paper, we propose a hybrid neuro-genetic system for stock trading. A recurrent neural network having one hidden layer is used for the prediction model. The input features are generated from a number of technical indicators being used by financial experts. The genetic algorithm optimizes the neural network's weights under a two-dimensional encoding and crossover. We devised a context-based ensemble method of neural networks which dynamically changes on the basis of the test day's context. To reduce the time in processing mass data, we parallelized the genetic algorithm on a Linux cluster system using message passing interface. We tested the proposed method with 36 companies in NYSE and NASDAQ for 13 years from 1992 to 2004. The neuro-genetic hybrid showed notable improvement on the average over the buy-and-hold strategy and the context-based ensemble further improved the results. We also observed that some companies were more predictable than others, which implies that the proposed neuro-genetic hybrid can be used for financial portfolio construction.

Index Terms– Stock prediction, parallel genetic algorithm, recurrent neural network, ensemble model, message passing interface

1 Introduction

It is a topic of practical interest to predict the trends of financial objects such as stocks, currencies, and options. It is not an easy job even for financial experts because they are

[*] E-mail: *kwon@soar.snu.ac.kr*
[†] E-mail: *moon@soar.snu.ac.kr*

mostly nonlinear, uncertain, and nonstationary. There is no consensus among experts as to how well, if possible, financial time series are predictable and how to predict them.

In this paper, we focus on the development of automatic stock trading systems based on artificial neural networks (ANNs) and genetic algorithms (GAs). Prior studies demonstrated that they might be efficient in stock prediction. For example, ANNs showed better performance than statistical regression models [30] and discriminant analysis [37]. In [31], three ANNs, time delay, recurrent, and probabilistic networks, were proven attractive. Support vector machine, another type of neural network, also showed good generalization [7]. On the other hand, genetic algorithms with tree-structured solutions were often used to produce understandable rules of an expert or an analyst. In [22], the authors examined the predictability of genetic programming and developed a single day-trading strategy. Genetic programming of polynomials showed another promising result [17].

Although ANNs are dominant so far in stock prediction as surveyed in [39], it is important how to find the most valuable input features and how to use them in applying them to stock prediction. In [38], they used not only quantitative variables but also qualitative information to more accurately forecast stock prices. Textual information in articles published on the web was also used in [33]. The most widely used inputs include daily transaction volume, interest rate, stock prices, moving average, rate of change, and so on [7] [31]. In addition, the optimization of weights is another important issue in ANNs [35]. The backpropagation algorithm is the most popular one for the supervised training of ANNs, but yet it is just a local gradient technique and there is room for further optimization. Genetic algorithms have been considered to have potential to reinforce the performance of ANNs. Various schemes for combining genetic algorithms and neural networks were proposed or compared, which were mostly designed to optimize the networks weights or to find a good topology (see survey papers [5] [34]). In the field of stock prediction, genetic algorithms using one-dimensional encoding were mostly used to train the network's weights and topologies [21] [15] [16].

In this paper, we try to predict the stock price using a hybrid genetic approach combined with recurrent neural networks. We describe a number of input variables that help the network to forecast the next day price. For the input variables, technical indicators or signals are used that were developed in deterministic trading techniques. The backpropagation algorithm is prone to get stuck in local minima and highly depends on the initial weights; a genetic algorithm is used for optimizing neural network's weights. The operators of genetic algorithm provide diverse initial weights for the network under two-dimensional encoding since it can reflect more geographical linkages of genes than one-dimensional encoding.

This paper seeks to find answers to several questions. One is whether the trading strategy based on a technical analysis is valid or not. Although many previous works showed successful results, most of them were tested on a small number of stocks or during a short period. In this paper, we test the proposed approach with 36 stocks for 13 years to validate the performance. A wide range of tests are important in that a goal of the stock prediction system is to choose some predictable stocks among a number of companies. Another question is how to implement the neuro-genetic hybrid system to make a trading decision within a practical response time. A demerit of such hybrid GAs is high computational cost. To reduce the time in processing mass data, we parallelized the genetic algorithm properly on a Linux cluster system using message passing interface.

The last question is about the performance measure. Previous works mostly evaluate the performance based on statistical measures such as mean squared/absolute error, prediction accuracy, false positive/negative errors, and so on. However, the most curious objective is how much money it makes. In addition, it is known to be hard to make profit with automatic stock trading systems when the transaction costs are considered. The difficulty was confirmed by a large body of literature on the effectiveness of various technical trading rules. The majority of them have found that such rules do not make money. For example, Alexander tests a number of filter rules, which advise a trader to buy if the price has risen by a fixed percentage and sell if the price has declined by the same percentage [1]. Although such rules appear to yield returns above the buy-and-hold strategy for the Dow Jones and Standard & Poor's stock indices, he concluded that they were not profitable when transaction costs were taken into account [2]. These conclusions were supported by the results of Fama and Blume, who found no evidence of profitable filter rules for the 30 Dow Jones stocks [11] [10], and many other results [3] [29] [27]. On the other hand, a few studies have reported positive profitability of technical trading strategies. In [32], for 11 of the 14 Dow Jones stocks that had earned profits in [11], a 0.5% filter rule produced annual mean returns after adjustment for 0.1% one-way transaction cost during 1970-82. Bessembinder and Chan showed that moving average-based trading rules can generate positive profits in Asian emerging markets such as Malaysia, Thailand, and Taiwan [4]. However, it was also observed that the profits were quite sensitive to transaction costs.

To overcome the problem, we use an evolutionary ensemble model. An ensemble learning is to aggregate multiple subsystems to solve a complex problem and expect stable performance, and since genetic algorithms produce many solutions it is natural to make an ensemble model with them. The basic evolutionary ensemble is one that chooses some best solutions and makes a decision by the opinion of the majority or the average output of the ensemble. In this paper, we propose a different ensemble model, a context-based ensemble. The model does not use the same class of members of ensemble for all test data. For each test data, it dynamically chooses a subset of members that perform well for the subset of training data being the most similar to the test data. In this model, the ensemble consists of the members performing best for the days with the most similar contexts to today's.

The rest of this paper is organized as follows. In Section 2, we explain the problem and present the objective. In Section 3, we describe our hybrid genetic algorithm and ensemble model for predicting the stock price. In Section 4, we provide our experimental results. Finally, conclusions and future works are given in Section 5.

2 Preliminaries

2.1 The Problem and Dataset

For automatic stock trading, there are a number of models such as intraday, daily, weekly, and monthly trading, depending on behavioral scopes. In this work, we consider the daily trading model.

In the problem, each record of dataset includes daily information which consists of the closing price, the highest price, the lowest price, and the trading volume. We name them at day t as $x(t)$, $x_h(t)$, $x_l(t)$, and $v(t)$, respectively. The trading strategy is based on the series of $x(t)$; if we expect $x(t+1)$ is notably higher than $x(t)$ we buy the stocks; if lower, we sell them; otherwise, we do not take any action. The problem is a kind of time-series data prediction that can be usually first tried with delay coordinates as follows:

$$x(t+1) = f(x(t), x_h(t), x_l(t), x(t-1), x_h(t-1), x_l(t-1), \ldots).$$

But, it is a simple and weak model. We transform the original time series to another that is more suitable for neural networks. First, $\frac{x(t+1)-x(t)}{x(t)}$ is used instead of $x(t+1)$ as the target

```
if ( signal is $SELL$ ) {
    $C_{t+1} \leftarrow C_t + min(B, S_t) \times (1 - T)$
    $S_{t+1} \leftarrow S_t - min(B, S_t)$
}
if ( signal is $BUY$ ) {
    $C_{t+1} \leftarrow C_t - min(B, C_t)$
    $S_{t+1} \leftarrow S_t + min(B, C_t)$
}
$S_{t+1} \leftarrow S_t \times \frac{x_{t+1}}{x_t}$
```

Figure 1: Investing strategy and change of the property

variable. For the input variables, we use technical indicators or signals that were developed in deterministic trading techniques. To achieve it, we construct the model as follows:

$$\frac{x(t+1) - x(t)}{x(t)} = f(g_1, g_2, \ldots, g_m).$$

where g_k $(k = 1, \ldots, m)$ is a technical indicator or signal.

2.2 The Objective

There can be a number of measures to evaluate the performance of the trading system. In our problem, we simulate the daily trading as close as possible to the actual situation and evaluate the profit. We have a cash balance C_t and stock S_t at day t $(t = 1, \ldots, N)$. We start with C, i.e, $C_1 = C$ and $S_1 = 0$. Figure 1 shows the investing strategy and change of property at day $t+1$ according to the signal at day t of the trading system. In the strategy, the constant B is the upper bound of stock trade per day and T is the one-way transaction cost. We have the final property ratio P as follows:

$$P = \frac{C_N + S_N}{C_1 + S_1}.$$

3 The Suggested System

3.1 Processing Data

As mentioned, we have four daily data, x, x_h, x_l, and v, but we do not use them for the input variables as they are. We utilize a number of technical indicators being used by financial experts [23]. We describe some of them in the following:

- Moving average (MA)
  - The numerical average value of the stock prices over a period of time.
  - MA_S, MA_L : short-term and long-term moving average, respectively.
- Golden-cross and dead-cross
  - States that MA_S crosses MA_L upward and downward, respectively.
- Moving average convergence and divergence ($MACD$)
  - A momentum indicator that shows the relationship between MA_S and MA_L.
  - $MACD = MA_S - MA_L$
- Relative strength index (RSI)
  - An oscillator that indicates the internal strength of a single stock.
  - $RSI = 100 - \frac{100}{1 + U/D}$,
    U, D : An average of upward and downward price changes, respectively.
- Stochastics
  - An indicator that compares where a stock price closed relative to its price range over a given time period.
  - $\%K = \frac{x(t) - L}{H - L} \times 100$,
    $H, L =$ The highest and the lowest price in a given time period.
  - $\%D =$ Moving average of $\%K$

We generate 75 input variables using the technical indicators[1]. Figure 2 shows some representative variables. The others not shown in this figure include variables in the same forms as the above that use trading volumes in place of the prices. After generating the new variables, we normalize them by dividing by the maximum value of each variable. It helps the neural network to learn efficiently.

[1]At first, we generated 116 input variables and eliminated 41 variables which are lowly correlated with the target variable or mutually highly correlated with another input variable.

$$
\begin{aligned}
X_1 &= (x(t) - x(t-1))/x(t-1) \\
X_2 &= (x(t) - x_l(t))/x_h(t) - x_l(t) \\
X_3 &= (MA(t) - MA(t-1))/MA(t-1) \\
X_4 &= (MA_S(t) - MA_L(t))/MA_L(t) \\
X_5 &= (x(t) - MA(t))/MA(t) \\
X_6 &= x(t) - min(x(t), x(t-1), \ldots, x(t-5)) \\
X_7 &= x(t) - max(x(t), x(t-1), \ldots, x(t-5)) \\
X_8 &= \text{\# of days since the last golden-cross} \\
X_9 &= \text{\# of days since the last dead-cross} \\
X_{10} &= \text{the profit while the stock has risen or fallen continuously} \\
X_{11} &= \text{\# of days for which the stock has risen or fallen continuously} \\
X_{12} &= MACD(t) \\
X_{13} &= RSI(t) \\
X_{14} &= \%K(t) \\
X_{15} &= \%D(t) \\
X_{16} &= \%K(t) - \%K(t-1) \\
X_{17} &= \%D(t) - \%D(t-1)
\end{aligned}
$$

Figure 2: Some examples of input variables

3.2 Artificial Neural Networks

We use a recurrent neural network architecture which is a variant of Elman's network [9]. It consists of input, hidden, and output layers as shown in Figure 3. Each hidden unit is connected to itself and also connected to all the other hidden units. The network is trained by a backpropagation-based algorithm.

It has 75 nodes in the input layer corresponding to the variables described in Section 3.1. Only one node exists in the output layer for $\frac{x(t+1)-x(t)}{x(t)}$, the change rate of the next day's closing price over today's. In the testing phase, the trading decision generates one of the three signals, *BUY*, *SELL*, and *KEEP* [2], based on the predicted output value $\hat{y}$. If $\hat{y}$ is greater than a positive threshold, the network produces the signal *BUY*. On the other hand, $\hat{y}$ is less than a negative threshold, it produces the signal *SELL*. Otherwise, it produces the signal *KEEP*. In this paper, the thresholds were set to 0.005 and -0.005, respectively.

[2]The signal *KEEP* means no action.

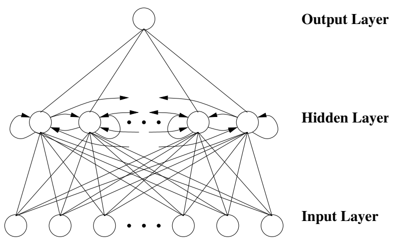


Figure 3: The recurrent neural network architecture

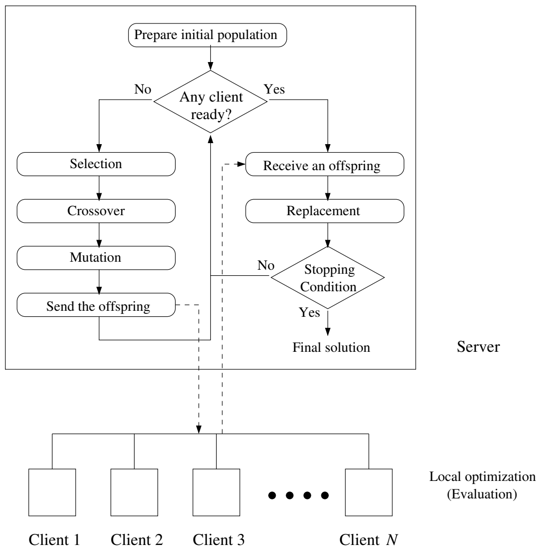


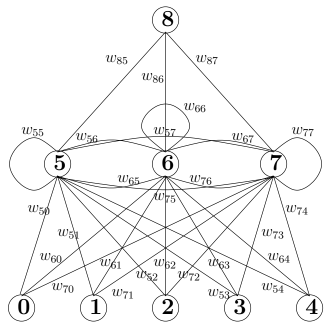
Figure 4: The framework of the parallel genetic algorithm

3.3 Distribution of Loads by Parallelization

There have been many attempts to optimize the architectures or weights of ANNs. In this paper, we use a GA to optimize the weights. Especially, we parallelize the genetic algorithm since it takes much time to handle data from a long period of time. The structure of the parallel GA is shown in Figure 4.

Parallel genetic algorithms can be categorized into three classes [6]: (i) global single-population master-slave GAs, (ii) single-population fine-grained, and (iii) multiple-population coarse-grained GAs. We take the first model for this work. In this neuro-genetic hybrid approach, the fitness evaluation is dominant in running time. Evaluating an offspring (a network), the backpropagation-based algorithm trains the network with the training data.

In a measurement with *gprof*, the evaluation part took about 95% of the total running



A recurrent neural network

$$\begin{array}{c|ccc}
 & \overbrace{0 \quad 1 \quad 2 \quad 3 \quad 4}^{\textit{input}} & \overbrace{5 \quad 6 \quad 7}^{\textit{hidden}} & \overbrace{8}^{\textit{output}} \\ \hline
5 & w_{50} \; w_{51} \; w_{52} \; w_{53} \; w_{54} & w_{55} \; w_{56} \; w_{57} & w_{85} \\
6 & w_{60} \; w_{61} \; w_{62} \; w_{63} \; w_{64} & w_{65} \; w_{66} \; w_{67} & w_{86} \\
7 & w_{70} \; w_{71} \; w_{72} \; w_{73} \; w_{74} & w_{75} \; w_{76} \; w_{77} & w_{87}
\end{array}$$

Corresponding 2-D chromosome

Figure 5: Encoding in the GA

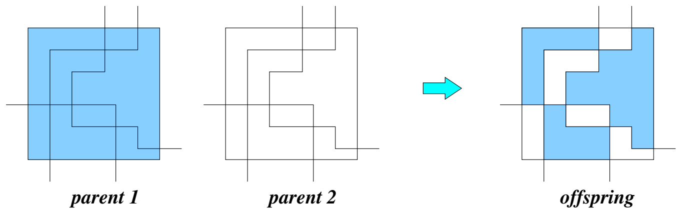


Figure 6: An example of 2-D geographical crossover

time. We distribute the load of evaluation over the clients (slaves) of a Linux cluster system. The main genetic parts locate in the server (master). When a new ANN is created by crossover and mutation, the GA passes it to one of the clients. When the evaluation is completed in the client, it sends the result back to the server. The server communicates with the clients in an asynchronous mode. This eliminates the need to synchronize every generation and can maintain a high level of processor utilization, even if the slave processors operate at different speeds. This is possible because we use a steady-state GA which does not wait until a set of offspring is generated. All these are achieved with the help of MPI (Message Passing Interface), a popular interface specification for programming distributed memory systems. In this work, we used a Linux cluster system with 46 CPUs.

As shown in Figure 4, the process in the server is a traditional steady-state GA. In the following, we describe each part of the GA.

- *Representation:* Most GAs used linear encodings in optimizing ANN's weights [14] [25]. Recently, a two-dimensional encoding has proven to perform favorably [24]. We represent a chromosome by a two-dimensional weight matrix as shown in Figure 5. In the matrix, each row corresponds to a hidden unit and each column corresponds to an input, hidden, or output unit. A chromosome is represented by $p \times (n + p + q)$ where n, p, and q are the numbers of input, hidden, output units, respectively. In this work, the matrix size is $20 \times (75 + 20 + 1)$.

- *Selection, crossover, and mutation:* Two parent chromosomes are selected with probabilities that are proportional to their fitness values. The fitness values are normalized in such a way that the best chromosome is chosen with a probability four times higher than that of the worst chromosome. This is a general practice in the GA community [12]. The normalized fitness value of a chromosome in the population is computed as follows:

$$F_k = P_k - P_w + (P_b - P_w)/3$$

where

F_k : fitness of chromosome k

P_k : final property ratio of chromosome k (defined in Section 2.2)

b, w : the indices of the best and the worst chromosomes in the population, respectively.

The offspring is produced through geographic two-dimensional crossover [20]. It is known to create diverse new schemata and reflect well the geographical relationships among genes. It chooses a number of lines, divides the chromosomal domain into two equivalent classes, and alternately copies the genes from the two parents as shown in Figure 6. The GA then perturbs the solution with the following mutation operator. It generates a random number in the range [0, 1] for each gene (entry) of the offspring; if the random number for the gene is smaller than a preset probability, it is replaced with an arbitrary number in the preset range. All these three operators are performed in the server.

- *Local optimization:* After an offspring is modified by a mutation operator, it is locally optimized by backpropagation which helps the GA fine-tune around local optima. Local optimization is mingled with quality evaluation. As mentioned, it is performed in the client and the result is sent to the server.

- *Replacement and stopping criterion:* The offspring first attempts to replace the more similar parent to it. If it fails, it attempts to replace the other parent and the most inferior member of the population in order. Replacement is done only when the offspring is better than the replacee. The GA stops if it does not find an improved solution for a fixed number generations.

3.4 Instance-based Ensemble of Neural Networks

An ensemble learning is to aggregate multiple subsystems to solve a complex problem. A number of approaches have been developed for ensemble learning [28] [18] [36] [13] [8]. The method is based on the fact that a solution with the smallest training error does not necessarily guarantee the most generalized one.

It is usual to select the best individual as the final solution in genetic algorithms. However, there is room for improving the performance with help of other individuals in the population. Evolutionary ensemble approaches select a subset of the population as ensemble. It consists of some best individuals or representative ones from the whole population. In the latter, a clustering algorithm such as k-means algorithm [26] is used and a representative solution for each cluster is selected. In this paper, we devised an instance-based ensemble which is different from traditional ensembles. Traditional ensemble models do not consider the relationship or difference between data; the members of the ensemble are chosen with respect to a fixed set of instances. The basic idea of our instance-based ensemble is that it does not fix the members of ensemble but dynamically chooses the members that perform well for the days with similar contexts to today's. The instance-based ensembles are determined as follows:

i) We obtain a set of NNs by the genetic algorithm.

ii) For each test day, we select a set S of instances among the training days that are the most similar to the test day in terms of Euclidean distance. The distance is computed from the normalized 75-variable vector for each day which is described in Section 3.1.

iii) We construct a subset of NNs, as an ensemble, that predict relatively well on the days in S.

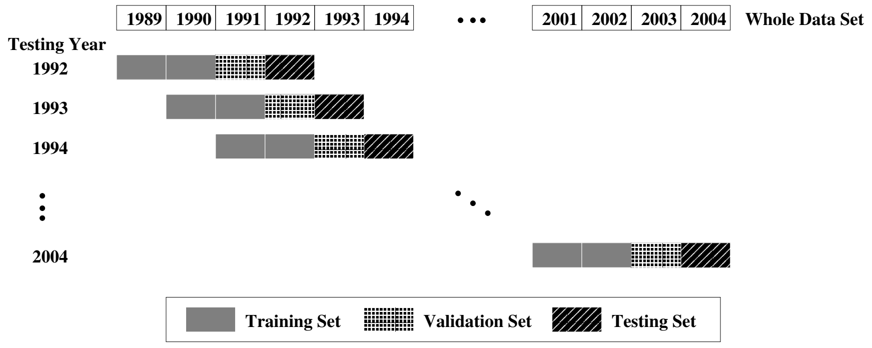


Figure 7: Testing scheme used in this paper

The trading decision depends on the majority decision in the ensemble. The final decision generates one of the following three signals: *BUY*, *SELL*, and *KEEP*. In the course, we extract three opinions from the ensemble, D_1 and D_2. D_1 is about the direction of the price at the next day. D_2 is about the direction of the price at the day after the next day. The ensemble produces the signal *BUY* when both D_1 and D_2 are "*up*," and produces the signal *SELL* when both D_1 and D_2 are "*down*." Otherwise, it produces the signal *KEEP*. By this strategy, the trading becomes more conservative and too prompt an action can be avoided.

4 Experimental Results

We tested our approaches with 36 companies' stocks in NYSE and NASDAQ. We got the whole data from 1989 to 2004 from Yahoo (http://quote.yahoo.com). We divided the whole data set into 13 overlapping training-validation-testing sets as shown in Figure 7; it is the walk-forward testing routine which is commonly used in evaluating the prediction performance of time-series data [7] [19]. To test a stock price of year Y, the GA was trained with the two consecutive years of data from year $Y - 3$ to year $Y - 2$, and validated with the data of year $Y - 1$; in other words, only the three most recent years of data were used to predict one year. In this paper, the testing year Y was shifted year by year from 1992 to 2004.

Table 1: Relative performance of *RNN* and *GA* over the buy-and-hold (36 companies and $T = 0$)

Symbols	Market State			*RNN*		*GA*	
	Up	*Down*	*NC*	*Better*	*Worse*	*Better*	*Worse*
AA	8	3	2	5	4	**8**	**1**
AXP	10	1	2	**2**	**3**	2	5
AYP	8	4	1	4	2	**5**	**2**
BA	8	5	0	3	4	**4**	**4**
C	8	3	2	**4**	**3**	**4**	**3**
CAT	9	1	3	**4**	**5**	**4**	**5**
DD	7	3	3	6	2	**8**	**0**
DELL	10	2	1	3	2	**7**	**3**
DIS	7	4	2	**5**	**2**	**6**	**3**
EK	7	6	0	**9**	**0**	8	0
GE	9	3	1	4	5	**4**	**4**
GM	8	5	0	7	1	**8**	**1**
HD	8	3	2	**5**	**4**	**5**	**4**
HON	9	4	0	4	3	**6**	**4**
HWP	9	4	0	4	2	**8**	**2**
IBM	10	3	0	3	4	**4**	**4**
INTC	9	3	1	4	4	**4**	**3**
IP	5	4	4	**8**	**2**	**8**	**2**
JNJ	9	4	0	**2**	**1**	**4**	**3**
JPM	6	1	2	2	1	**3**	**0**
KO	7	4	2	**4**	**2**	**4**	**2**
MCD	7	3	3	**6**	**2**	6	3
MMM	8	1	4	2	3	**4**	**4**
MO	7	3	3	3	2	**6**	**2**
MRK	6	6	1	5	2	**6**	**2**
MSFT	8	3	2	4	3	**5**	**2**
NMSB	9	3	1	**12**	**0**	**12**	**0**
ORCL	8	4	1	2	4	**3**	**4**
PG	11	1	1	**3**	**1**	2	1
RYFL	4	7	0	**11**	**0**	**11**	**0**
SBC	6	5	2	7	2	**8**	**1**
SUNW	9	4	0	**7**	**4**	6	4
T	5	6	2	**5**	**5**	3	5
UTX	9	2	2	**3**	**2**	2	3
WMT	6	4	3	**5**	**2**	**6**	**3**
XOM	8	2	3	**9**	**1**	8	1
Up	282			68	83	**93**	**83**
Down		124		79	3	79	5
NC			56	29	3	30	2
Total	282	124	56	176	89	**202**	**90**

4.1 Property evaluation

Firstly, we tested our approaches without consideration of transaction cost ($T = 0$ in Figure 1). Table 1 shows the summary of the experimental results. The values mean the number of cases (years). In the Market State part of the table, *Up*, *Down*, and *NC* represent the states of the stock market in each year. The *Up* and *Down* mean that the closing price of the year has risen or fallen against the year's starting price by 5% or more, respectively. *NC* means no notable difference. For example, eight years have risen and three years have fallen among the 13 years in case of AA (Alcoa Inc.). *RNN* and *GA* are the average results by the recurrent neural network (20 trials) and the hybrid genetic algorithm

Table 2: Relative performance of ensemble approaches over the buy-and-hold (36 companies and $T = 0.003$)

Symbols	market state			*Winer*		*M-Ensemble*		*A-Ensemble*		*I-Ensemble*	
	Up	*Down*	*NC*	*Better*	*Worse*	*Better*	*Worse*	*Better*	*Worse*	*Better*	*Worse*
AA	8	3	2	**7**	**3**	5	3	4	4	3	2
AXP	10	1	2	4	5	0	4	3	5	**5**	**3**
AYP	8	4	1	3	4	1	3	**3**	**3**	**2**	**2**
BA	8	5	0	5	5	2	5	4	5	**3**	**2**
C	8	3	2	4	5	2	3	**4**	**3**	2	2
CAT	9	1	3	4	6	4	5	4	6	**6**	**3**
DD	7	3	3	6	3	**6**	**1**	8	4	**7**	**2**
DELL	10	2	1	4	7	4	3	4	6	**5**	**3**
DIS	7	4	2	**6**	**3**	3	4	6	4	**5**	**2**
EK	7	6	0	7	2	6	2	7	1	**8**	**1**
GE	9	3	1	2	5	2	5	3	4	**3**	**3**
GM	8	5	0	8	3	6	2	**8**	**2**	**8**	**2**
HD	8	3	2	**2**	**4**	0	4	1	5	0	4
HON	9	4	0	**5**	**4**	4	4	**5**	**4**	2	4
HWP	9	4	0	7	3	7	2	5	3	**9**	**1**
IBM	10	3	0	3	7	3	5	4	7	**4**	**1**
INTC	9	3	1	3	6	3	6	3	7	**4**	**2**
IP	5	4	4	7	3	6	2	6	3	**7**	**0**
JNJ	9	4	0	2	5	2	2	3	4	**2**	**1**
JPM	6	1	2	0	5	0	2	0	6	**2**	**2**
KO	7	4	2	4	4	2	3	**5**	**3**	3	2
MCD	7	3	3	4	5	**3**	**3**	**4**	**4**	**2**	**2**
MMM	8	1	4	2	5	1	5	2	5	**4**	**3**
MO	7	3	3	4	6	3	5	2	4	**4**	**5**
MRK	6	6	1	**7**	**2**	4	3	6	3	5	1
MSFT	8	3	2	4	4	**4**	**3**	4	4	**3**	**2**
NMSB	9	3	1	10	2	10	2	**10**	**1**	9	2
ORCL	8	4	1	4	6	5	3	3	3	**6**	**1**
PG	11	1	1	1	4	1	5	2	3	**1**	**1**
RYFL	4	7	0	**11**	**0**	**11**	**0**	**11**	**0**	**11**	**0**
SBC	6	5	2	**7**	**2**	**8**	**3**	**7**	**2**	6	2
SUNW	9	4	0	**8**	**3**	5	3	6	4	4	2
T	5	6	2	4	5	2	5	3	5	**6**	**3**
UTX	9	2	2	2	6	1	4	**2**	**3**	**2**	**3**
WMT	6	4	3	4	3	4	3	4	2	**6**	**2**
XOM	8	2	3	5	3	**7**	**2**	5	3	5	1
Up	282			57	134	52	102	62	122	**80**	**64**
Down		124		88	5	66	11	77	8	62	6
NC			56	25	9	19	6	22	5	22	4
Total	282	124	56	170	148	137	119	161	135	**164**	**74**

(20 trials), respectively. *Better* and *Worse* represent the relative performance of *RNN* and *GA* over the buy-and-hold. *Better* and *Worse* mean that the final property ratio of the learned strategy is at least 5% higher or lower than that of the buy-and-hold, respectively. Since there are 36 companies tested for 13 years, we have 462 cases except six cases with deficient data (four cases in JPM and two cases in RYFL). The GA performed better than the buy-and-hold in 202 cases, worse in 90 cases, and comparable in 170 cases. *RNN* and *GA* showed a significant performance improvement over the buy-and-hold on the average. The *GA* considerably improved the performance of *RNN* in the *Up* cases.

Table 3: Relative performance of ensemble approaches over the buy-and-hold (13 years and $T = 0.003$)

Year	market state			*Winer*		*M-Ensemble*		*A-Ensemble*		*I-Ensemble*	
	Up	*Down*	*NC*	*Better*	*Worse*	*Better*	*Worse*	*Better*	*Worse*	*Better*	*Worse*
1992	22	9	5	10	11	7	10	12	10	**10**	**5**
1993	21	9	6	12	12	9	11	10	7	**13**	**5**
1994	19	8	9	10	8	10	6	10	8	**9**	**3**
1995	34	0	2	5	25	4	19	6	23	**9**	**11**
1996	29	3	4	9	12	**11**	**5**	11	9	**10**	**4**
1997	32	3	1	9	15	6	11	8	14	**9**	**5**
1998	24	8	4	11	10	7	8	10	9	**10**	**6**
1999	26	6	4	16	10	14	7	15	12	**20**	**4**
2000	12	22	2	19	8	11	10	15	10	**17**	**3**
2001	12	18	5	**19**	**5**	18	5	18	5	15	4
2002	4	26	5	24	6	18	4	**22**	**3**	21	5
2003	27	5	2	11	19	12	16	12	17	**12**	**14**
2004	20	7	7	**15**	**7**	10	7	12	8	9	5
Total	282	124	56	170	148	137	119	161	135	**164**	**74**

4.2 Neural Networks Ensemble

We tested our ensemble approach under the consideration of transaction cost ($T = 0.003$ in Figure 1). Table 2 and Table 3 show the experimental results by companies and years, respectively. (The detailed result is listed in Appendix A.) *I-Ensemble* is the instance-based ensemble described in Section 3.4 and *Winner* is the version that uses the best solution. *A-Ensemble* is the version that selects a set of best NNs in the population and makes the decision from the average output of them, and *M-Ensemble* is the version that selects a set of best NNs in the same way as *A-Ensemble* and makes the decision by the the majority opinion of them. They are average results over 20 trials. The *I-Ensemble* performed better than the buy-and-hold in 164 cases, worse in 74 cases, and comparable in 224 cases. More specifically, *I-Ensemble* performed best in 26 companies among 36 and during 10 years among 13. *Winner* uses the best NN and it is an approach with no ensemble. It did not show good performance when considering the transaction cost, primarily due to too often trades. *I-Ensemble* showed a significant performance improvement over not only *Winner* but also the other ensemble models on the average. In addition, we can observe that some companies are more predictable than others. Practically, one can choose the stocks of companies that turned out to be more stably predictable by the GA.

Table 4: The average accuracy improvement of GA over "Base" (%)

Symbols	BUY	SELL
AA	2.96	14.92
AXP	0.99	5.50
AYP	1.30	9.01
BA	1.24	4.49
C	0.49	3.00
CAT	1.89	6.95
DD	1.43	18.27
DELL	0.60	14.67
DIS	1.04	9.26
EK	1.18	10.24
GE	0.76	17.35
GM	6.31	11.62
HD	1.97	-0.02
HON	1.21	4.64
HWP	2.70	22.62
IBM	0.57	12.26
INTC	0.47	6.06
IP	1.11	7.40
JNJ	1.94	5.14
JPM	2.87	10.67
KO	1.14	8.51
MCD	1.77	5.36
MMM	2.65	8.98
MO	1.64	7.86
MRK	1.26	13.64
MSFT	1.29	9.06
NMSB	28.82	38.42
ORCL	0.30	2.53
PG	1.83	12.37
RYFL	43.47	42.18
SBC	1.58	10.72
SUNW	0.34	8.13
T	1.30	12.57
UTX	-0.17	5.58
WMT	6.84	8.92
XOM	1.02	18.62

4.3 Accuracy evaluation

We analyze the frequency of correct prediction in *I-Ensemble* model. Figure 8 shows the results. There are three charts for each company, "BUY", "SELL", and "TOTAL"; "BUY" and "SELL" charts are about the accuracies of *BUY* and *SELL* signals, respectively, and "TOTAL" is about the accuracies of both the two signals. In "BUY" and "SELL" charts, "Base" lines mean the percentage of the number of days when the price has risen and fallen over the number of the whole days, respectively. We exclude the days at which the price change is too small (0.1%). Table 4 shows the average accuracy improvements over the base accuracy for each company. It is an interesting phenomenon that it has a larger improvement in *SELL* than that in *BUY*.

4.4 Time Reduction of Parallel Genetic Algorithm

As mentioned, the genetic algorithm is parallelized on a Linux cluster system with 46 CPUs. Its objective is to reduce the processing time without loss of performance. The parallel GA and the single-CPU GA showed little difference in terms of the final property ratio. On the other hand, the 46-CPU parallel GA was about 39 times faster than the single-CPU GA on the average.

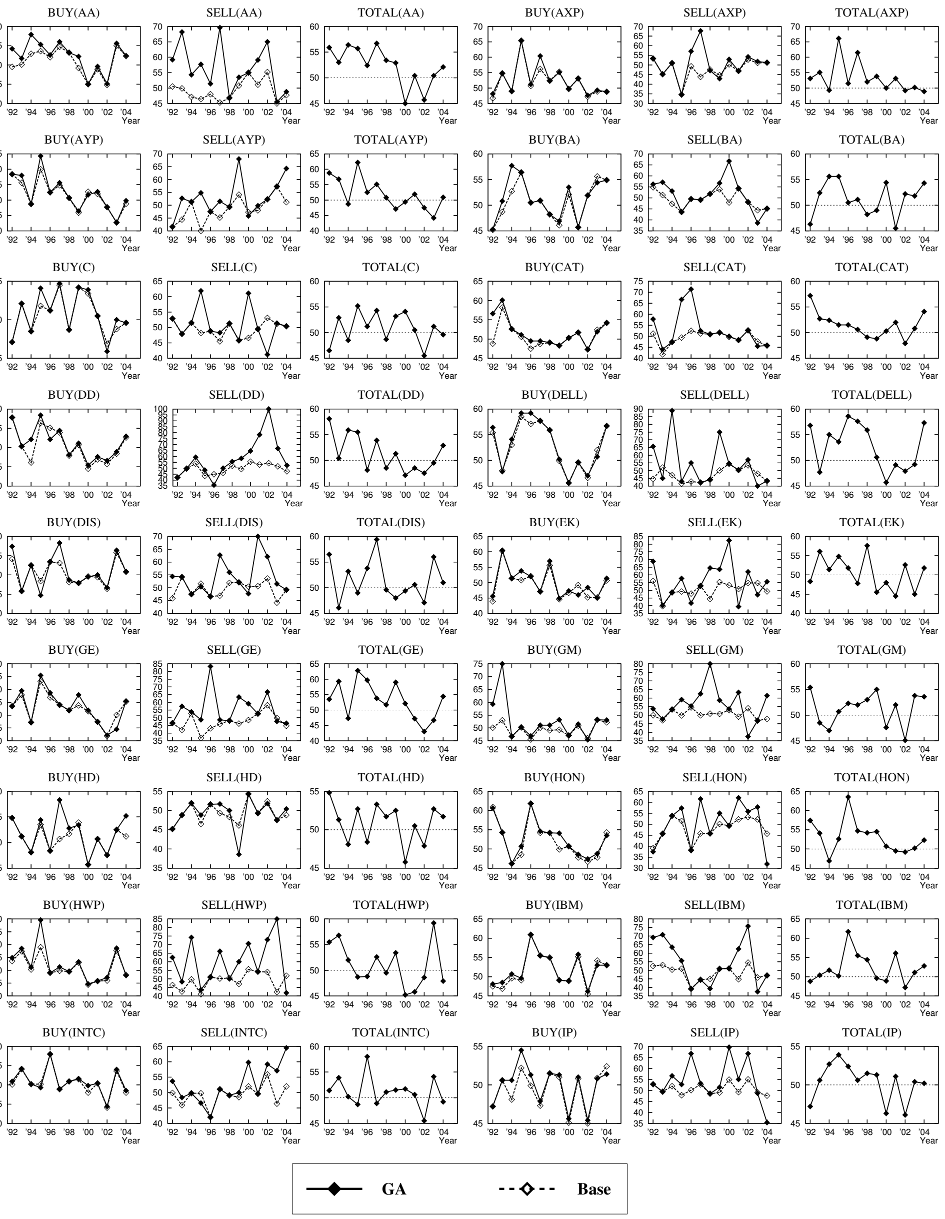


Figure 8: Accuracies

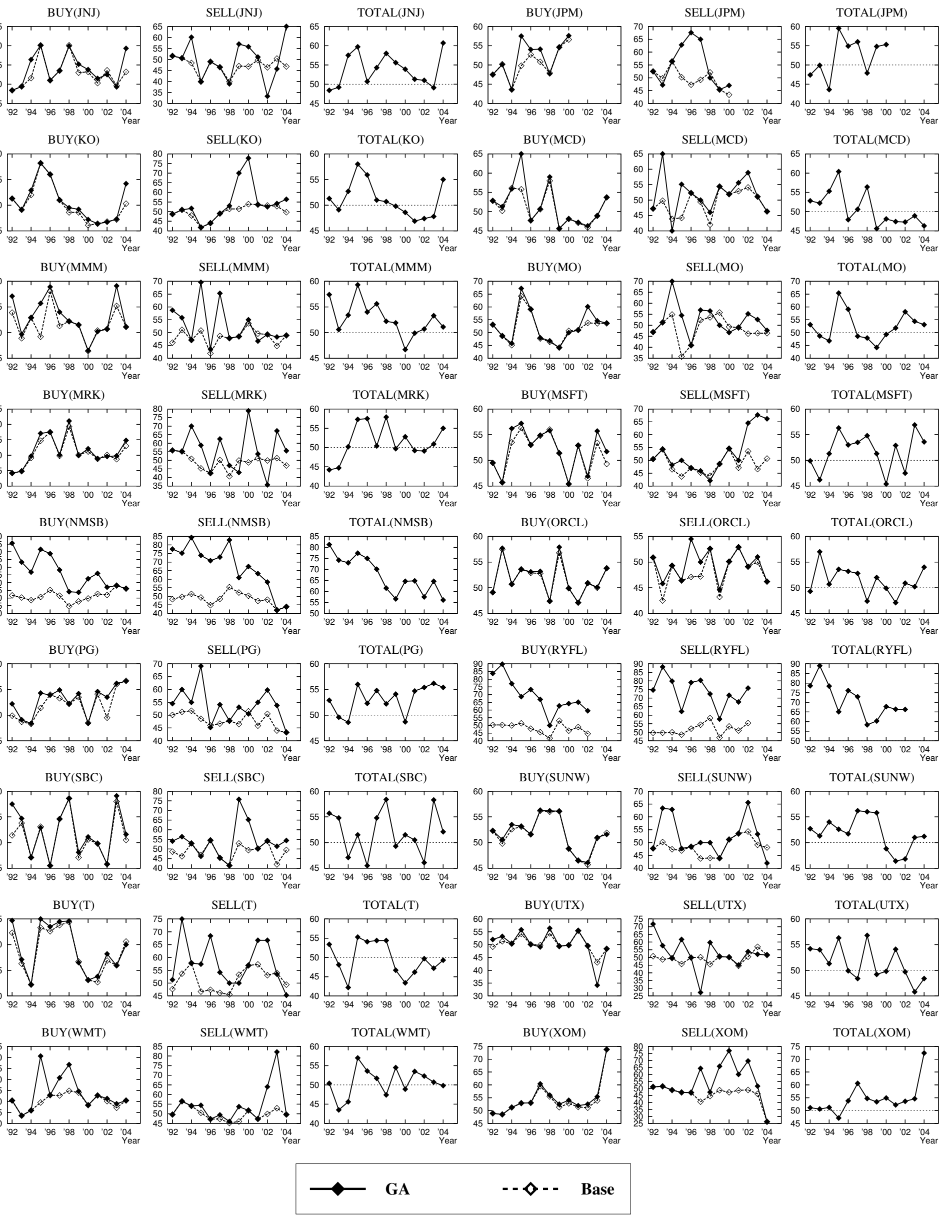


Figure 8: Continued

4.5 Comparison with Other Approaches

We compare our approach with two other approaches: genetic programming and support vector machine. We explain them briefly in the following.

- Genetic programming (GP) : GP which represents a solution using a tree has been often used to predict stock price. In this paper, the GP in [22] was compared. In the GP, $+$, $-$, $\times$, $\div$, exp, sqrt, ln, sin, and cos were used for function node set, and 28 variables consisting of $x(t), \ldots, x(t-6), x_h(t), \ldots, x_h(t-6), x_l(t), \ldots, x_l(t-6), v(t-1), \ldots, v(t-6)$ were used for terminal node set. Other GP parameters were also determined as in [22].

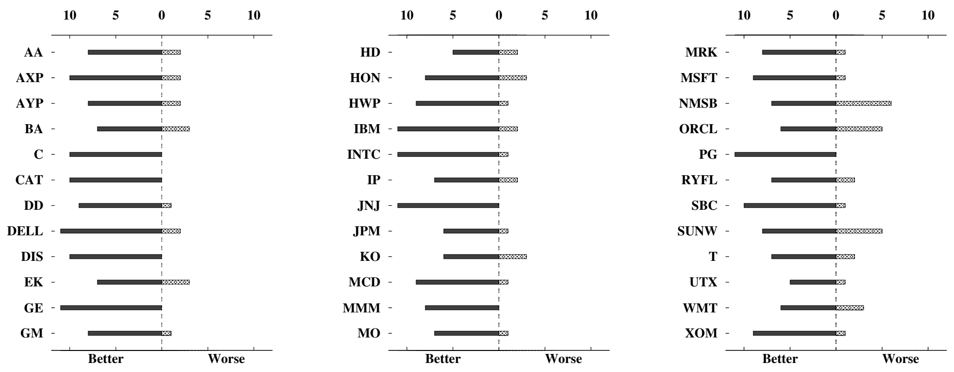
- Support vector machine (SVM) : SVM developed by Vapnik has been receiving increasing attention in pattern recognition. We refer to the model used in [7]. Independent variables consist of $x(t) - EMA_{100}(t)$, $\frac{x(t)-x(t-5)}{x(t-5)}$, $\frac{x(t-5)-x(t-10)}{x(t-10)}$, $\frac{x(t-10)-x(t-15)}{x(t-15)}$, and $\frac{x(t-15)-x(t-20)}{x(t-20)}$ and dependent variable is $\frac{EMA_3(t+5)-EMA_3(t)}{EMA_3(t)}$ where $EMA_n(t)$ is the n-day exponential moving average at day t. In this experiment, the Gaussian function is used as the kernel function, and the validation set is used to select the optimal parameter of the kernel function.

Figure 9 and Figure 10 show the comparison results between our approach and other ones by companies and years, respectively. In the figures, "Better" and "Worse" mean the number of cases where the final property of our GA is higher or lower than that of the corresponding approach by 5% or more, respectively. Our GA showed consistently better performance; it predicted better in 34 companies among 36 than both GP and SVM. Although it also performed better for all tested years, the gaps against the other approaches were greater when the market was strong (in the state *Up*).[3]

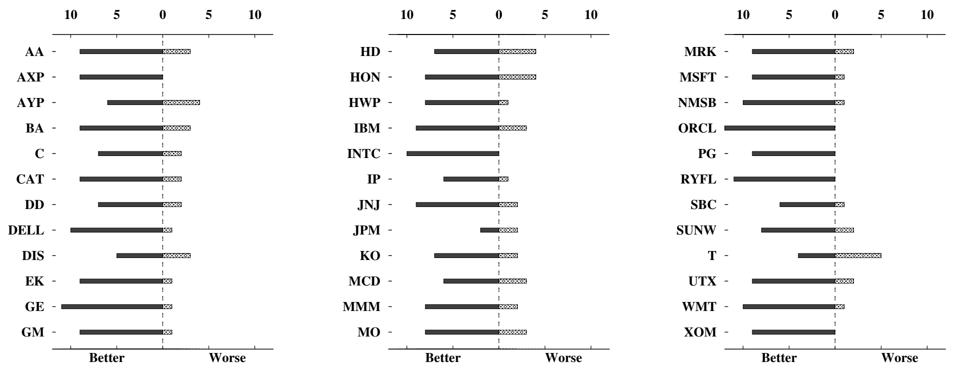
4.6 Application to Portfolio Construction

More accurate stock prediction is relevant to investors in that it is a starting step in constructing an optimal portfolio. We do not handle the portfolio optimization problem here

[3]As shown in Table 3, the market's state was *Down* in from 2000 through 2002 and it was *Up* in the other years.



GA vs. GP

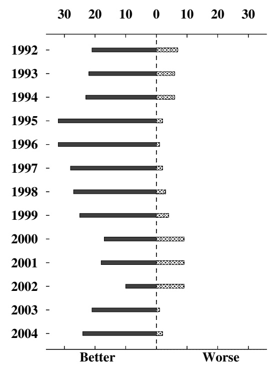


GA vs. SVM

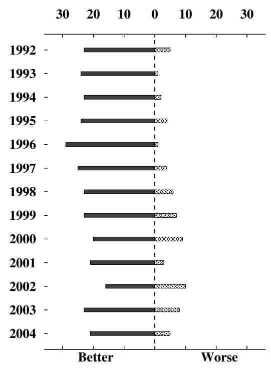
Figure 9: Comparison of our approach with other approaches by companies

but examine whether the proposed system can be utilized for it or not. In portfolio construction based on a stock prediction system, it chooses a set of stocks which are expected to rise. However, no system can predict well on all companies or for all periods. The most tractable scenario to construct the next year's portfolio is that it chooses the companies which have been predicted relatively better than other companies for the recent years. Therefore, it is desirable that prediction performance is consistent in as many companies as possible.

We investigated performance consistency in Figure 11. In the figure, X-axis is the number of consecutive recent years that our GA performs better than the buy-and-hold strategy and left Y-axis denotes the number of two cases, "Consistency" and "Not" which mean it predicts consistently better than the buy-and-hold or not, respectively, in the next



GA vs. GP



GA vs. SVM

Figure 10: Comparison of our approach with other approaches by years

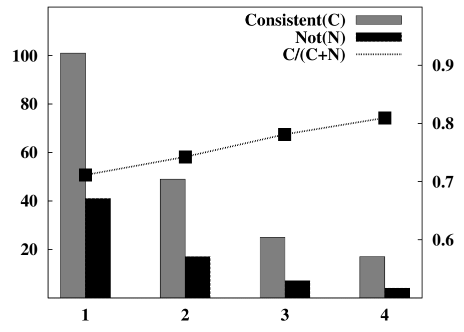


Figure 11: Performance consistency

year. Right Y-axis is the ratio of "Consistency" cases over the total number of cases. Our approach shows a considerable promising result; the number of "Consistency" cases is two times or more larger than that of "Not" cases. The probability that our GA consistently beats the buy-and-hold in the next year is over 0.8 when the former has performed better than the latter in each of the four most recent consecutive years.

5 Conclusion and Future Work

In this paper, we proposed a genetic algorithm combined with a recurrent neural network having one hidden layer for the daily stock trading. The proposed method was tested with 36 companies in NYSE and NASDAQ for 13 years from 1992 to 2004 and showed significantly better performance than the "buy-and-hold" strategy. For the input nodes of the neural network, we transform the stock prices and the volumes into a large number of technical indicators or signals. The GA optimizes the neural network's weights under a two-dimensional encoding and crossover. This turned out to contribute to the performance improvement. We parallelized the GA since it takes much time to process mass data from a long period of time. It was implemented on a Linux cluster system with 46 CPUs using message passing interface. It reduced the response time about 39 times without loss of performance. We devised a new evolutionary ensemble model, a context-based ensemble method of neural networks which dynamically changes on the basis of the test day's context; the model proved to be profitable even when considering the transaction costs. That was possible as it made the decision of buying or selling more conservative and effective.

Future study will include portfolio optimization based on the proposed prediction model. In the experiments, the proposed GA predicted better for some companies than for other companies and the performance was considerably consistent. It implies that this work can be useful in portfolio optimization. However, further researches such as how many stocks to choose and how many shares of each stock to purchase are necessary. Feature selection/extraction will be also included in future work. Appropriately selected input variables avoids the curse of dimensionality, improves the generalization, and reduces the computational cost, but it is not easy to find an optimal subset of input variables. In particular, it is an urgent task in that investigating more companies for longer periods is necessary to construct a more stable portfolio.

References

[1] S. S. Alexander. Price movements in speculative markets: trends or random walks. *Industrial Management Review*, 2:7–26, 1961.

[2] S. S. Alexander. *The Random Character of Stock Market Prices*. MIT Press, 1964.

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

[4] H. Bessembinder and K. Chan. The profitability of technical trading rules in the asian stock markets. *Pacific-Basin Finance Journal*, 3:257–284, 1995.

[5] J. Branke. *Evolutionary Algorithms for Neural Network Design and Training.* Technical Report No.322, University of Karlsruhe, Institute AIFB., 1995.

[6] E. Cantu-Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles*, 10(2):141–171, 1998.

[7] L. J. Cao and F. E. H. Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14:1506–1518, 2003.

[8] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001, 1990.

[9] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[10] E. F. Fama. Efficient capital markets: a review of theory an empirical work. *Journal of Finance*, 25:383–417, 1970.

[11] E. F. Fama and M. E. Blume. Filter rules and stock market trading. *Journal of Business*, 39:226–241, 1966.

[12] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[13] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001, 1990.

[14] S. A. Harp, T. Samad, and A. Guha. Towards the genetic synthesis of neural networks. In *International Conference on Genetic Algorithm*, pages 360–369, 1989.

[15] P. G. Harrald and M. Kamstra. Evolving artificial neural networks to combine financial forecasts. *IEEE Transactions on Evolutionary Computation*, 1(1):40–52, 1997.

[16] S. Hayward. Simulating profitable stock trading strategies with an evolutionary artificial neural network. In *International Workshop on Intelligent Finance*, pages 108–117, 2004.

[17] H. Iba and N. Nikolaev. Genetic programming polynomial models of financial data series. In *Proceedings of the Congress on Evolutionary Computation*, pages 1459–1466, 2000.

[18] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[19] I. Kaastra and M. Boyd. Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 10:215–236, 1996.

[20] A. B. Kahng and B. R. Moon. Toward more powerful recombinations. In *International Conference on Genetic Algorithms*, pages 96–103, 1995.

[21] F. Kai and X. Wenhua. Training neural network with genetic algorithms for forecasting the stock price index. In *IEEE International Conference on Intelligent Processing Systems*, pages 401–403, 1997.

[22] M. A. Kanoudan. Genetic programming prediction of stock prices. *Computational Economics*, 16:207–236, 2000.

[23] P. J. Kaufman. *Trading Systems and Methods.* John Wiley & Sons, 1998.

[24] J. H. Kim and B. R. Moon. Neuron reordering for better neuro-genetic hybrids. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 407–414, 2002.

[25] C. T. Lin and C. P. Jou. Controlling chaos by ga-based reinforcement learning neural network. *IEEE Transactions on Neural Networks*, 10(4):846–869, 1999.

[26] J. MacQueen. Some methods for classification and analysis of multivariate observation. In *Proceedings of the 5th Berkley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[27] C. J. Neely. Risk-adjusted, ex ante, optimal technical trading rules in equity markets. *International Review of Economics and Finance*, 12(1):69–87, 2003.

[28] N. J. Nilsson. *Learning Machines: Foundations of Trainable pattern-classifying systems.* New York: McGraw Hill, 1965.

[29] M. J. Ready. Profits from technical trading rules. *Financial Management*, 31:43–61, 2002.

[30] A. N. Refenes, A. Zapranis, and G. Francis. Stock performance modeling using neural networks: A comparative study with regression models. *Neural Networks*, 7:375–388, 1994.

[31] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch. Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks*, 9:1456–1470, 1998.

[32] R. J. Sweeney. Some new filter rule test: Methods and results. *Journal of Financial and Quantitative Analysis*, 23:285–300, 1988.

[33] B Wuthrich, V. Cho, S. Leung, D. Permunetilleke, K. Sankaran, and J. Zhang. Daily stock market forecast from textual web data. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 2720–2725, 1999.

[34] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87:1423–1447, 1999.

[35] X. Yao and Y. Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8:694–713, 1997.

[36] X. Yao and Y. Liu. Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics–Part:B*, 28:417–425, 1998.

[37] Y. Yoon, G. Swales Jr., and T. M. Margavio. A comparison of discriminant analysis versus artificial neural networks. *Journal of the Operational Research Society*, 44:51–60, 1993.

[38] Y. Yoon and G. Swales. Predicting stock price performance: a neural network approach. In *Proc. 24th Annual Hawaii International conference on System Sciences*, pages 156–162, 1991.

[39] D. Zhang and L. Zhou. Discovering golden nuggets: Data mining in financial application. *IEEE Transactions on Systems, Man, and Cybernetics–Part:C*, 34:513–522, 2004.

Appendix A: P values ($T = 0.003$)

Symbols	Strategies	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004
AA	*I-Ensemble*	1.191	1.069	1.154	1.232	1.217	1.165	1.077	1.719	0.797	1.195	0.677	1.035	0.864
	Winner	1.052	1.067	1.057	1.271	1.218	1.161	1.239	1.176	0.902	1.056	0.773	1.795	0.918
	A-Ensemble	1.020	1.046	1.058	1.172	1.234	1.129	1.284	1.201	0.797	1.127	0.817	1.797	0.849
	M-Ensemble	1.163	1.063	1.103	1.151	1.236	1.082	1.121	1.255	0.795	1.223	0.790	1.715	0.865
AXP	*I-Ensemble*	1.028	1.320	1.133	1.413	1.353	1.712	1.203	1.660	1.065	0.690	0.961	1.211	1.181
	Winner	0.981	1.028	1.067	1.200	1.298	1.384	1.232	1.638	1.091	0.693	1.046	1.191	1.132
	A-Ensemble	0.966	1.008	1.072	1.080	1.284	1.425	1.231	1.584	1.054	0.692	1.061	1.040	1.211
	M-Ensemble	0.966	1.014	1.075	1.398	1.336	1.542	1.183	1.548	1.034	0.690	0.974	1.180	1.146
AYP	*I-Ensemble*	1.076	1.159	0.825	1.325	1.039	1.054	1.071	0.760	1.316	0.861	0.230	1.625	1.441
	Winner	1.061	1.113	0.813	1.245	1.014	0.990	1.066	0.780	1.255	0.926	0.417	1.764	1.154
	A-Ensemble	1.073	1.116	0.813	1.280	1.029	0.969	1.028	0.760	1.268	0.993	0.334	1.856	1.099
	M-Ensemble	1.065	1.098	0.813	1.375	1.034	0.984	1.063	0.727	1.237	0.976	0.205	1.727	1.246
BA	*I-Ensemble*	0.806	1.097	1.133	1.678	1.300	0.946	0.724	1.214	1.714	0.584	0.899	1.098	1.211
	Winner	0.937	1.011	1.035	1.574	1.292	0.983	0.732	1.118	1.058	0.679	0.908	1.380	1.038
	A-Ensemble	0.903	1.047	0.995	1.570	1.287	0.970	0.737	1.109	1.064	0.656	0.894	1.379	1.008
	M-Ensemble	0.790	1.019	1.108	1.574	1.293	0.968	0.730	1.200	1.076	0.603	0.876	1.391	1.129
C	*I-Ensemble*	1.218	1.868	0.817	1.718	1.440	1.853	0.934	1.607	1.224	1.010	0.822	0.988	0.983
	Winner	1.062	1.759	0.818	1.333	1.192	1.813	1.004	1.712	1.133	1.006	0.834	1.052	1.005
	A-Ensemble	1.189	1.848	0.817	1.313	1.419	1.764	1.001	1.759	1.145	1.006	0.789	0.982	0.984
	M-Ensemble	1.138	1.803	0.817	1.511	1.444	1.747	0.949	1.698	1.254	1.008	0.769	1.011	0.983
CAT	*I-Ensemble*	1.362	1.424	1.255	1.197	1.383	1.175	0.967	1.147	1.013	1.237	0.964	1.631	1.157
	Winner	1.113	1.136	1.238	1.242	1.377	1.013	0.986	1.164	0.857	0.991	0.931	1.532	1.159
	A-Ensemble	1.119	1.130	1.219	1.269	1.396	0.985	0.967	1.268	0.824	0.997	0.973	1.332	1.123
	M-Ensemble	1.208	1.276	1.224	1.232	1.579	1.171	0.968	1.128	0.782	1.001	0.946	1.540	1.151
DD	*I-Ensemble*	1.047	1.019	1.063	1.258	1.242	1.359	1.051	1.249	0.795	0.987	1.124	1.163	1.092
	Winner	1.076	1.009	1.010	1.072	1.077	1.218	1.098	1.312	0.932	1.014	1.314	1.173	1.044
	A-Ensemble	1.138	1.017	1.011	1.073	1.072	1.191	1.133	1.304	0.914	1.025	1.276	1.167	1.160
	M-Ensemble	1.068	1.010	1.103	1.169	1.269	1.245	1.096	1.282	0.950	1.030	1.269	1.167	1.060
DELL	*I-Ensemble*	2.851	0.558	1.829	1.584	2.179	3.300	3.461	1.514	0.341	1.558	1.129	1.153	1.207
	Winner	3.111	0.734	1.826	1.505	2.092	2.606	3.217	1.260	0.338	1.891	0.901	1.137	1.224
	A-Ensemble	3.308	0.834	1.745	1.579	1.941	3.087	3.340	1.089	0.348	1.998	0.907	1.153	1.283
	M-Ensemble	3.304	0.733	1.798	1.803	2.081	3.240	3.432	1.229	0.346	1.957	0.983	1.129	1.281
DIS	*I-Ensemble*	1.349	1.026	1.135	1.110	1.119	1.510	0.869	1.003	1.014	0.827	0.887	1.502	1.157
	Winner	1.246	1.076	1.064	1.053	1.124	1.245	0.887	1.037	1.046	0.916	1.231	1.505	1.282
	A-Ensemble	1.236	1.090	1.055	1.006	1.199	1.284	0.835	1.039	0.933	0.974	1.317	1.495	1.237
	M-Ensemble	1.305	1.035	1.093	1.082	1.148	1.333	0.809	1.044	0.919	0.900	1.176	1.489	1.165
EK	*I-Ensemble*	0.897	1.188	1.048	1.556	1.213	0.832	1.192	1.006	0.659	0.741	1.354	0.801	1.327
	Winner	0.906	1.286	1.054	1.308	1.173	0.813	1.199	0.999	0.896	1.044	1.202	0.791	1.362
	A-Ensemble	0.918	1.376	1.084	1.317	1.191	0.823	1.177	1.028	0.706	1.176	1.240	0.788	1.423
	M-Ensemble	0.929	1.221	1.124	1.439	1.182	0.846	1.172	1.160	0.640	0.895	1.147	0.788	1.367
GE	*I-Ensemble*	1.116	1.171	1.013	1.312	1.348	1.430	1.361	1.611	0.934	0.966	0.702	0.988	1.164
	Winner	1.062	1.150	1.009	1.282	1.218	1.408	1.379	1.512	0.947	0.928	1.035	0.922	1.135
	A-Ensemble	1.104	1.161	0.997	1.281	1.301	1.441	1.439	1.514	0.937	0.885	0.992	0.932	1.094
	M-Ensemble	1.129	1.162	0.993	1.239	1.237	1.437	1.383	1.618	0.898	0.883	1.172	0.913	1.151
GM	*I-Ensemble*	1.145	1.234	0.757	1.351	0.968	1.228	1.187	1.339	0.780	0.968	0.895	1.518	0.785
	Winner	1.029	1.084	0.915	1.330	0.997	1.211	1.139	1.348	0.753	1.160	1.052	1.299	0.827
	A-Ensemble	1.040	1.068	0.785	1.332	0.981	1.234	1.206	1.333	0.760	1.082	1.129	1.534	0.830
	M-Ensemble	1.022	1.061	0.799	1.303	0.968	1.290	1.113	1.296	0.785	1.027	1.095	1.552	0.754
HD	*I-Ensemble*	1.437	0.785	1.175	0.895	1.035	1.207	1.268	1.683	0.715	1.118	0.496	1.421	1.111
	Winner	1.444	0.835	1.166	1.017	1.082	1.254	1.301	1.656	0.728	1.065	0.569	1.177	1.082
	A-Ensemble	1.433	0.824	1.167	1.059	1.079	1.241	1.172	1.554	0.722	1.076	0.540	1.252	0.998
	M-Ensemble	1.439	0.808	1.174	0.982	1.060	1.273	1.381	1.666	0.730	1.063	0.483	1.375	1.023
HON	*I-Ensemble*	1.357	1.312	0.883	1.289	1.403	1.226	1.103	1.117	0.816	0.802	0.817	1.163	1.014
	Winner	1.099	1.339	0.994	1.143	1.249	1.233	1.159	1.119	0.897	0.922	0.668	1.297	1.032
	A-Ensemble	1.033	1.362	0.997	1.163	1.214	1.212	1.148	1.093	0.946	0.864	0.673	1.478	1.060
	M-Ensemble	1.202	1.244	0.924	1.187	1.376	1.216	1.110	1.067	0.843	0.837	0.720	1.318	0.965
HWP	*I-Ensemble*	1.284	1.267	1.400	1.200	1.302	1.306	1.197	1.790	0.727	0.716	0.930	1.458	0.924
	Winner	1.152	1.162	1.420	1.123	1.339	1.422	1.080	1.376	0.872	0.721	0.921	1.429	0.955
	A-Ensemble	1.102	1.098	1.452	1.156	1.336	1.381	1.083	1.255	0.996	0.716	0.819	1.436	0.893
	M-Ensemble	1.400	1.209	1.412	1.217	1.273	1.344	1.106	1.923	0.788	0.716	0.839	1.371	0.852
IBM	*I-Ensemble*	0.556	1.281	1.411	1.238	1.747	1.378	1.835	1.309	0.734	1.446	0.744	0.964	1.070
	Winner	0.666	1.007	0.988	1.128	1.575	1.211	1.603	1.254	0.747	1.518	0.950	0.942	1.081
	A-Ensemble	0.620	1.044	0.994	1.107	1.587	1.264	1.610	1.400	0.735	1.510	0.979	0.935	1.101
	M-Ensemble	0.602	1.010	0.979	1.157	1.653	1.405	1.524	1.364	0.734	1.483	0.949	0.952	1.064
INTC	*I-Ensemble*	1.514	1.408	1.041	1.516	2.216	1.114	1.679	1.622	0.730	1.078	0.564	2.127	0.789
	Winner	1.198	1.325	1.051	1.281	1.741	1.155	1.690	1.136	0.737	1.220	1.080	0.889	0.810
	A-Ensemble	1.128	1.344	1.061	1.356	1.683	1.114	1.660	1.107	0.656	1.198	1.028	0.861	0.810
	M-Ensemble	1.082	1.316	1.052	1.460	2.173	1.116	1.670	1.288	0.659	1.158	1.034	0.939	0.814
IP	*I-Ensemble*	0.946	1.026	1.108	1.135	1.141	1.067	0.932	1.395	0.776	1.117	0.947	1.184	1.087
	Winner	0.955	1.027	1.121	1.166	1.167	0.996	1.002	1.064	0.779	1.137	0.958	1.074	1.060
	A-Ensemble	0.946	1.014	1.095	1.192	1.219	0.952	0.964	1.064	0.866	1.181	0.957	1.063	1.088
	M-Ensemble	0.946	1.007	1.124	1.192	1.184	1.001	0.934	1.366	0.749	1.165	0.973	1.016	1.089

Symbols	Strategies	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004
JNJ	*I-Ensemble*	0.882	0.923	1.156	1.566	1.165	1.341	1.304	1.191	1.226	1.123	0.895	0.889	1.225
	Winner	0.890	0.937	1.109	1.524	1.152	1.317	1.320	1.197	1.337	1.057	0.866	0.789	1.064
	A-Ensemble	0.882	0.926	1.100	1.635	1.144	1.330	1.419	1.157	1.333	1.069	0.873	0.780	1.093
	M-Ensemble	0.882	0.916	1.122	1.605	1.225	1.334	1.399	1.128	1.286	1.143	0.897	0.788	1.199
JPM	*I-Ensemble*	0.942	0.966	0.814	1.425	1.280	1.215	0.945	1.132	1.232				
	Winner	0.928	0.963	0.823	1.270	1.076	1.079	0.988	1.081	1.236	N/A	N/A	N/A	N/A
	A-Ensemble	0.904	0.931	0.814	1.164	1.083	1.139	0.971	1.020	1.221				
	M-Ensemble	0.942	0.938	0.815	1.440	1.217	1.146	0.954	1.185	1.235				
KO	*I-Ensemble*	1.047	1.060	1.050	1.450	1.381	1.293	0.909	0.879	1.208	0.846	0.953	1.125	0.949
	Winner	1.133	1.060	0.982	1.365	1.339	1.255	0.913	0.876	1.162	0.881	0.927	0.974	0.933
	A-Ensemble	1.124	1.060	0.960	1.430	1.341	1.288	0.903	0.887	1.222	1.035	0.911	0.966	0.952
	M-Ensemble	1.043	1.060	1.028	1.476	1.376	1.313	0.899	0.863	1.062	0.943	0.959	0.976	0.953
MCD	*I-Ensemble*	1.248	1.185	1.050	1.645	1.035	1.076	1.667	1.030	0.845	0.875	0.616	1.403	1.002
	Winner	1.108	1.134	1.122	1.432	1.053	1.035	1.100	1.120	0.847	0.985	0.607	1.285	0.989
	A-Ensemble	1.232	1.094	1.159	1.424	1.057	1.029	1.156	1.064	0.845	1.011	0.639	1.310	0.997
	M-Ensemble	1.282	1.093	1.129	1.535	1.054	1.027	1.361	1.071	0.845	1.005	0.614	1.383	0.997
MMM	*I-Ensemble*	1.068	1.173	1.042	1.385	1.238	1.007	0.894	1.406	1.401	0.917	1.082	1.099	0.984
	Winner	1.088	1.116	1.062	1.205	1.082	0.997	0.895	1.025	1.022	0.914	1.038	1.020	1.029
	A-Ensemble	1.109	1.115	1.061	1.199	1.036	0.965	0.894	1.021	1.043	0.909	1.028	1.020	1.018
	M-Ensemble	1.058	1.098	1.089	1.208	1.126	1.000	0.894	1.114	1.097	0.907	1.032	1.046	1.003
MO	*I-Ensemble*	0.959	0.755	1.050	1.495	1.214	1.344	1.152	0.512	1.191	1.148	0.811	1.196	1.045
	Winner	0.959	0.814	0.934	1.114	1.156	1.353	1.156	0.624	1.403	0.948	0.773	1.157	1.197
	A-Ensemble	0.959	0.755	0.994	1.087	1.214	1.358	1.195	0.653	1.290	0.966	0.684	0.982	1.146
	M-Ensemble	0.959	0.754	1.034	1.379	1.215	1.367	1.211	0.629	1.364	0.941	0.729	1.073	1.199
MRK	*I-Ensemble*	0.796	0.799	1.204	1.677	1.243	1.321	1.278	0.962	1.349	0.705	0.960	0.928	0.952
	Winner	0.845	0.811	1.064	1.188	1.338	1.174	1.328	0.970	1.315	0.949	1.005	0.928	0.903
	A-Ensemble	0.863	0.806	1.068	1.117	1.372	1.158	1.367	0.957	1.274	1.037	0.950	0.928	0.951
	M-Ensemble	0.793	0.803	1.136	1.403	1.261	1.245	1.341	0.973	1.273	0.864	0.951	0.928	0.924
MSFT	*I-Ensemble*	1.156	0.957	1.229	1.438	1.819	1.575	2.020	1.665	0.372	1.531	0.888	1.140	1.035
	Winner	1.134	0.996	1.220	1.203	1.853	1.298	1.899	1.857	0.380	1.491	1.026	1.051	1.041
	A-Ensemble	1.075	1.001	1.221	1.242	1.828	1.334	1.973	1.994	0.384	1.559	1.072	1.020	1.075
	M-Ensemble	1.137	1.002	1.251	1.155	1.841	1.587	1.950	1.704	0.378	1.539	1.166	1.095	1.055
NMSB	*I-Ensemble*	4.804	4.291	1.350	3.045	1.558	1.937	1.231	1.136	1.319	1.319	1.123	1.469	1.072
	Winner	6.926	4.617	4.246	4.325	2.777	2.997	1.886	1.493	2.151	1.399	1.156	1.293	1.192
	A-Ensemble	7.356	4.505	4.392	4.271	2.852	3.095	2.071	1.529	2.284	1.438	1.280	1.310	1.214
	M-Ensemble	7.458	4.495	4.210	4.333	2.999	2.996	2.004	1.483	2.319	1.414	1.233	1.339	1.198
ORCL	*I-Ensemble*	2.093	2.133	1.504	1.513	1.581	0.895	1.847	3.018	0.893	0.530	0.901	1.238	1.056
	Winner	1.193	1.795	1.578	1.266	1.457	0.924	1.352	1.389	0.957	0.608	0.865	1.156	0.988
	A-Ensemble	1.059	1.900	1.551	1.453	1.632	0.946	1.615	1.445	0.909	0.544	0.829	1.171	1.024
	M-Ensemble	1.188	1.865	1.523	1.468	1.710	0.943	1.792	1.413	0.909	0.559	0.847	1.271	1.034
PG	*I-Ensemble*	1.200	1.079	1.121	1.385	1.259	1.468	1.112	1.248	0.746	1.089	1.015	1.132	1.121
	Winner	1.088	1.051	1.121	1.297	1.165	1.337	1.105	1.301	0.746	1.000	1.020	1.067	1.077
	A-Ensemble	1.108	1.089	1.152	1.298	1.135	1.423	1.110	1.299	0.739	1.045	0.950	1.123	1.060
	M-Ensemble	1.069	1.055	1.081	1.358	1.169	1.381	1.111	1.308	0.660	1.017	0.856	1.134	1.106
RYFL	*I-Ensemble*	5.058	9.069	13.122	5.734	6.118	2.971	1.356	2.186	2.165	3.209	1.527		
	Winner	7.577	12.320	15.427	6.203	7.458	4.884	1.542	2.820	3.486	5.280	2.020	N/A	N/A
	A-Ensemble	8.970	12.472	15.735	6.627	7.481	4.935	1.528	3.279	4.014	5.273	2.016		
	M-Ensemble	9.239	12.688	15.554	6.487	7.335	4.811	1.498	3.190	4.033	5.187	2.023		
SBC	*I-Ensemble*	1.098	1.227	0.979	1.251	0.901	1.433	1.481	0.995	1.070	0.876	0.776	0.811	1.096
	Winner	1.120	1.265	1.007	1.214	0.925	1.452	1.483	1.015	0.980	0.995	0.742	0.951	1.095
	A-Ensemble	1.127	1.248	0.995	1.200	0.911	1.518	1.393	1.065	0.922	1.108	0.727	0.999	1.098
	M-Ensemble	1.015	1.211	0.973	1.144	0.930	1.522	1.507	1.026	0.945	1.045	0.724	1.000	1.104
SUNW	*I-Ensemble*	1.163	0.938	1.279	2.458	1.227	1.614	1.726	3.141	0.665	0.528	0.294	1.498	1.304
	Winner	1.174	0.930	1.194	1.361	1.428	1.735	1.128	2.741	0.735	0.567	0.914	1.407	1.261
	A-Ensemble	1.185	0.943	1.155	1.218	1.383	1.783	1.187	3.078	0.683	0.593	1.024	1.284	1.291
	M-Ensemble	1.205	0.891	1.239	1.346	1.352	1.624	1.530	2.990	0.667	0.541	0.972	1.409	1.297
T	*I-Ensemble*	1.107	0.991	0.950	1.360	0.996	1.292	1.313	1.079	0.359	1.395	0.641	0.827	0.825
	Winner	0.970	0.985	0.950	1.010	1.029	1.103	1.194	1.084	0.422	1.317	1.079	0.756	0.874
	A-Ensemble	1.001	1.020	0.950	1.043	1.078	1.132	1.289	1.061	0.378	1.319	1.139	0.665	0.808
	M-Ensemble	0.934	0.948	0.950	1.104	1.035	1.152	1.322	1.054	0.351	1.285	0.976	0.749	0.784
UTX	*I-Ensemble*	0.987	1.126	1.070	1.469	1.383	1.127	1.625	1.139	1.204	0.779	1.000	1.005	1.095
	Winner	0.969	1.018	1.120	1.274	1.322	1.159	1.427	1.123	1.210	0.711	0.887	1.081	1.061
	A-Ensemble	1.007	1.026	1.156	1.233	1.345	1.149	1.413	1.154	1.201	0.685	0.928	1.455	1.105
	M-Ensemble	0.949	1.075	1.141	1.282	1.432	1.134	1.476	1.112	1.200	0.677	0.936	1.238	1.099
WMT	*I-Ensemble*	1.063	0.811	0.807	1.173	1.105	1.290	1.256	1.832	0.821	1.122	0.966	1.115	1.081
	Winner	1.071	0.807	0.761	1.087	0.996	1.256	1.124	1.720	0.931	1.108	0.995	1.088	1.061
	A-Ensemble	1.063	0.812	0.784	1.120	0.996	1.268	1.109	1.772	0.920	1.117	0.984	1.095	1.059
	M-Ensemble	1.063	0.811	0.765	1.074	1.067	1.190	1.191	1.811	0.895	1.101	0.972	1.070	0.960
XOM	*I-Ensemble*	1.115	1.132	0.951	1.115	1.249	1.290	1.190	1.139	1.262	0.927	0.991	1.197	1.256
	Winner	1.163	1.090	1.030	1.042	1.173	1.288	1.223	1.099	1.376	0.978	1.045	1.044	1.045
	A-Ensemble	1.147	1.076	1.023	1.034	1.183	1.301	1.185	1.124	1.420	0.960	1.007	1.016	1.011
	M-Ensemble	1.179	1.127	1.009	1.166	1.360	1.295	1.190	1.068	1.420	0.951	0.986	1.043	1.253