Stock Trading Rules Using Genetic Network Programming with Actor-Critic

Shingo Mabu, Member, IEEE, Yan Chen, Kotaro Hirasawa, Member, IEEE and Jinglu Hu

Abstract—Genetic Network Programming (GNP) is an evolutionary computation which represents its solutions using graph structures. Since GNP can create quite compact programs and has an implicit memory function, it has been clarified that GNP works well especially in dynamic environments. In this paper, GNP is applied to creating a stock trading model. The first important point is to combine GNP with Actor-Critic which is one of the reinforcement learning algorithms. Evolution-based methods evolve their programs after task execution because they must calculate fitness values, while reinforcement learning can change programs during task execution, therefore the programs can be created efficiently. The second important point is that GNP with Actor-Critic (GNP-AC) can select appropriate technical indexes to judge the buying and selling timing of stocks using Importance Index especially designed for stock trading decision making. In the simulations, the trading model is trained using the stock prices of 20 brands in 2001, 2002 and 2003. Then the generalization ability is tested using the stock prices in 2004. From the simulation results, it is clarified that the trading rules of GNP-AC obtain higher profits than Buy&Hold method.

I. INTRODUCTION

Genetic Network Programming (GNP)[1], [2] has been proposed as an extended method of Genetic Algorithm (GA)[3] and Genetic Programming (GP)[4]. GNP represents its solutions using graph structures, which contributes to creating quite compact programs and implicitly memorizing past action sequences, therefore it has been clarified that GNP is an effective method mainly for dynamic problems. Moreover, an extended algorithm of GNP which combines evolution and reinforcement learning[5] (GNP-RL) has been proposed. Original GNP is based on evolution only, therefore the programs are evolved mainly after task execution or enough trial, i.e., offline learning. On the other hand, GNP-RL can change its programs incrementally based on rewards obtained during task execution, i.e., online learning. For example, when an agent takes a good action with a positive reward at a certain state, the action is reinforced and therefore the same action will be adopted with higher probability when visiting the state again. The online learning is one of the advantages of GNP-RL.

The other advantage of GNP-RL is a combination of a diversified search of GNP and an intensified search of RL. Because RL is executed based on immediate rewards obtained after taking actions, intensified search, i.e., local search, can be executed efficiently. Evolution could change programs largely than RL, therefore the programs (solutions) could escape from local minima, so we call evolution as a diversified search.

Recently, research on stock price prediction and trading model using softcomputing such as evolutionary computation and neural networks has been done[6], [7], [8]. Generally speaking, methods for predicting stock prices and determining the timing of buying or selling stocks are divided into two groups; one is fundamental analysis which analyzes stock prices using the financial statement of each company, the economic trend and movements of the exchange rate; the other is technical analysis which analyzes numerically the past movement of stock prices. Generally, the research on stock price prediction and trading model using softcomputing belongs to technical analysis, so it determines the timing of buying or selling stocks based on the technical indexes such as rate of deviation, Relative Strength Index, Golden cross and so on. The proposed method also belongs to technical analysis.

In this paper, there are two important points. First, we combine GNP and Actor-Critic[9], [10] which is one of the reinforcement learning methods, and Importance Index (IMX) is introduced for efficient stock trading decision making. Concretely speaking, IMX tells GNP-AC whether or not the buying or selling signals are likely to appear at the current day, and Actor-Critic learns a threshold value which predicts the rise or fall of the stock prices based on IMX. Actor-Critic has a distinguished ability to deal with continuous actions1, so we adopt it for the learning of GNP. Second, although there are so many technical indexes in the technical analysis, GNP with Actor-Critic can select appropriate indexes to judge the buying and selling timing of stocks. In other words, GNP with Actor-Critic could optimize the combinations of important technical indexes using Importance Index especially designed for stock trading decision making.

This paper is organized as follows. In Section II, the algorithm of the proposed method is described. Section III explains simulation environments, conditions and results. Section IV is devoted to conclusions.

II. GNP WITH ACTOR-CRITIC (GNP-AC)

A. Basic structure of GNP-AC

Hereafter, we call GNP-RL using Actor-Critic as GNP-AC. Fig. 1 shows a basic structure of GNP-AC. GNP-AC consists of judgment nodes and processing nodes, and they

1We define the threshold value as an “action” in reinforcement learning.
are connected to each other. The distinguished point of GNP-AC is the small number of nodes (37 nodes in this paper) and this characteristic contributes to creating the compact program and saving memory consumption. A judgment node judges the information obtained from environments and determines the next node to be executed. A processing node predicts that the stock price five days ahead will rise or fall in the future. In this paper, GNP-AC predicts whether the stock price will be higher or lower than the current price.

GNP-AC has two kinds of time delays: time delay GNP-AC spends on judgment or processing, and one it spends on node transitions. GNP-AC can evolve programs considering the time spent on judgment and processing. \( d_i \) is the time delay spent on the judgment or processing at node \( i \), and \( d_{i1}^A, d_{i1}^B, \ldots \) are time delays spent on the node transition from node \( i \) to the next node. In this paper, the role of time delays is to determine the maximum number of technical indexes considered when GNP-AC predicts the rise or fall of stocks. In detail, \( d_{i1}^A, d_{i1}^B, \ldots \) are set at zero time unit, \( d_i \) of each judgment node is set at one time unit, \( d_{i1} \) of each processing node is set at five time units. In addition, we suppose that the prediction is determined when GNP-AC uses five or more time units. In other words, node transition in one day ends when GNP-AC executes fewer than five judgments and one processing, or five judgments. The details is described in section II-C.2.

B. Gene structure of GNP-AC

The graph structure of GNP-AC is determined by the set of the following genes. A genetic code of node \( i \) (0 ≤ \( i \) ≤ \( n^3 - 1 \)) is also shown in Fig. 1.

\( K_i \) represents the node type, \( K_i = 0 \) means start node, \( K_i = 1 \) means judgment node and \( K_i = 2 \) means processing node. \( ID_i \) represents an identification number of the node function, e.g., \( K_i = 1 \) and \( ID_i = 2 \) mean the node is \( J_i \). \( d_i \) is the time delay spent on judgment or processing. \( V_i \) is a state value. In this paper, each node corresponds to a state. In addition, each node has its own probability density function to predict the rise or fall of the stock price, so its mean \( \mu_i \) and standard deviation \( \sigma_i \) are written in the gene.

C. Node transition of GNP-AC and its learning method

1) Judgment node: The node transition starts from a start node and continues based on the connections between nodes, the judgment results at judgment nodes and the predictions at processing nodes. If the current node \( i \) is a judgment node, the following procedure is executed.

The gene \( ID_i \) shows the technical index \(^4\) GNP-AC judges at node \( i \), and one of the branches of node \( i \) is selected by the judgment result. Each technical index has its own IMX function shown in Fig. 2; x axis shows the value of each technical index, and the sections A, B, C ... correspond to the judgment results. For example, \( J_i \) is a judgment which judges a rate of deviation from moving average, and if the rate is more than 0.1, the judgment result becomes E; and the next node number becomes \( C_{i1} \) . y axis shows the output of the IMX function and it is used at a processing node. However, the IMX output of golden cross, dead cross and MACD could be 1, 0 or -1 based on the cross of the lines, and each value corresponds to judgment result A, B, C, respectively. Concretely speaking, for three days after a golden cross appears, the IMX output becomes +1, and for three days after a dead cross appears, it becomes -1, otherwise. Furthermore, for three days after MACD passes through the signal from the lower side to the upper side, the IMX output becomes 1, and for three days after it does from the upper to the lower, the IMX output becomes -1, otherwise it becomes 0. Generally, golden cross indicates buying signal and dead cross indicates selling signal, therefore, buying signal becomes larger as the output IMX is close to 1, and selling signal becomes larger as it is close to -1.

The probability density function of a judgment node generates a random number which becomes a weight \( a_i \) of the IMX. Note that notation \( t \) shows that the node is a \( t \)th

\(^4\)Technical indexes tell the signals of buy and sell.

2007 IEEE Congress on Evolutionary Computation (CEC 2007) 509
Fig. 2. IMX functions in Judgment nodes

rate of deviation from the current price to the highest price

rate of deviation from the lowest price to the current price

2007 IEEE Congress on Evolutionary Computation (CEC 2007)

Psychological line

Volume ratio

ROC

RSI

%D (Stochastic)

Rate of deviation

2007 IEEE Congress on Evolutionary Computation (CEC 2007)
node executed after a start node. In detail, a start node is a first node, the node connected from the start node is a second node, the next node connected from the second one is a third node and so on. Therefore $t$ does not show the date of the trading, but the internal time of GNP-AC. The weighted IMX is $a_i \times IMX$ and used at the next processing node to predict the rise or fall of the stock price.

2) Processing node: If the current node $i$ is a processing node, GNP-AC predicts that the stock price will rise or fall based on the following procedure.

1) generate a random number $a_t$, based on the probability density function in node $i$.
2) calculate an average of the weighted IMXs obtained at the judgment nodes executed in the node transition from the previous processing node to the current processing node.

$$A_t = \frac{1}{|\mathcal{I}|} \sum_{i' \in \mathcal{I}} (a_{i'} \times IMX(i')) ,$$ (1)

where, $\mathcal{I}$ shows a set of the judgment node numbers executed in the node transition from the previous processing node to the current processing node. $IMX(i')$ and $a_{i'}$ show an IMX and the weight calculated at node $i' \in \mathcal{I}$. However, when a IMX output is zero at a judgment node of golden cross, deadcross and MACD, the node number is excluded from $\mathcal{I}$ for calculating $A_t$.

3) determine rise or fall
- when $A_t \geq a_t$, GNP-AC predicts that the stock price will rise five days ahead
- when $A_t < a_t$, GNP-AC predicts that the stock price will fall five days ahead

Trading rules of the proposed method are created in such a way that we buy as much stocks as possible using the funds\(^5\) when GNP-AC predicts the stock price will rise, and sell all the stocks when GNP-AC predicts the stock price will fall. The buying and selling orders are executed at the opening of the trading day, i.e., we can buy and sell stocks with the opening price.

4) The current node is transferred to the next node. When the prediction is rise, the next node number becomes $C_A^i$, and when the prediction is fall, it becomes $C_B^i$.

The above procedure puts the information of the technical indexes together into $A_t$, and GNP-AC predicts that the stock price will rise or fall by comparing $A_t$ with $a_t$. Therefore, the points of this paper are to learn the mean and standard deviation of the probability density functions of the judgment nodes and processing nodes by Actor-Critic, and do it during the trading period, i.e., online learning. In addition, $\mathcal{I}$ is determined by evolution, i.e., what kinds of judgments (technical indexes) should be considered is determined automatically by evolution; therefore GNP-AC can optimize the combinations of important technical indexes using IMX especially designed for stock trading as well as evolution.

In the general reinforcement learning framework, the current state $s$ is determined by the information obtained from environments, and an action $a$ corresponds to an actual action an agent takes, however, in GNP-AC, each node is defined as a state, and an action corresponds to generating $a_t$ based on a probability density function at each node. Here, the learning procedure is explained using Fig. 3.

Fig. 3 shows an example where GNP-AC predicts that the stock price will fall at node $h$, rise at the current node $i$, and fall at node $j$.

As described in section II-A, the node transition in one day ends when GNP-AC uses more than five time units. Thus, if the node transition in one day ends without executing processing node, let the prediction of rise or fall be the same as the previous one. The reward $r_t$ used in reinforcement learning is given by Eq. (2) five days after GNP-AC predicts the rise or fall.

$$r_t = \begin{cases} 1 & \text{(when the prediction is correct)} \\ 0 & \text{(when the prediction is wrong)} \end{cases}.$$ (2)

GNP-AC updates the state value $V$ and the parameters of the probability density function ($\mu$ and $\sigma$) using $r_t$. If the reward is larger than expected, $\mu$ and $\sigma$ are updated in order that a random number $a_t$ produced at the current node will be produced with higher probability the next time when the node is executed again and that $a_t$ will be generated with lower probability if the reward is smaller than expected. The learning is executed as follows considering the above issue.

$$\delta_t = r_t + \gamma V_{t+1} - V_t ,$$ (3)

$$V_t \leftarrow V_{t} + \alpha \delta_t,$$ (4)

$$\mu_t \leftarrow \mu_t + \alpha_\mu (a_t - \mu) \delta_t',$$ (5)

$$\sigma_t \leftarrow \sigma_t + \alpha_\sigma \left( \frac{(a_t - \mu)^2}{\sigma^2} - 1 \right) \delta_t' ,$$ (6)

$$\delta_t' = \begin{cases} 0 & (\delta_t = 0) \\ 1 & (\delta_t > 0) \\ -1 & (\delta_t < 0) \end{cases}.$$ (7)

Where,
- $\gamma$ : discount rate
- $\alpha_\mu, \alpha_\sigma$ : learning rate
- $V_t, \mu_t, \sigma_t$ : $V, \mu, \sigma$ executed at time $t$

In the above example (Fig. 3), $V_t$ is $V_i$, and $V_{t+1}$ is a $V$ value of the next state, i.e., node $k$.

D. Evolution of GNP-AC

In the evolution phase, the best individual is preserved, and the rest of the individuals are replaced with the new ones produced by crossover and mutation. The fitness is a total correct predictions during the trading term.

\(^5\)Initial funds is 5,000,000 Japanese Yen.
1) Crossover:
1) select two individuals using tournament selection
2) select each node as a crossover node with the probability of $P_c$
3) exchange the genes of the corresponding crossover nodes, i.e., the nodes with the same node number, between the two parents
4) produced new individuals become the new ones in the next generation

2) Mutation:
1) select one individual using tournament selection
2) mutation operation
   a) change connection: Each node branch ($C_i^A, C_i^B, \ldots$) is selected with the probability of $P_m$, and the selected branch is reconnected to another node.
   b) change parameters ($\mu, \sigma$): each $\mu$ and $\sigma^7$ are changed to other values with the probability of $P_m$.
   c) change node function: Each node function ($ID_j$) is selected with the probability of $P_m$, and the selected function is changed to another one.
3) produced new individual becomes the new one in the next generation

III. SIMULATION
To confirm the effectiveness of GNP-AC, we carried out the trading simulations using 20 brands selected from the companies listed in the first section of Tokyo stock market in Japan, especially large market capitalization companies. The simulations are carried out using three terms; training, validation and testing terms.

- Training: January 4, 2001–October 1, 2003 (677 days)
- Validation: October 2, 2003–December 30, 2003 (60 days)
- Testing: January 5, 2004–December 30, 2004 (246 days)

First, GNP-AC programs are evolved using training data, and every generation, all the programs are evaluated using validation data to check the generalization ability. If the mean fitness of all the individuals in the validation term becomes less than $\eta$ (0.45 in this paper), all the individuals are initialized to avoid overfitting to the training data. After the programs are evolved with 300 generations, the best program in the training term is selected from the best population with the highest mean fitness in the validation term. Whole flowchart of learning and evolution is shown in Fig. 4.

A. Conditions of GNP-AC
GNP-AC uses judgment nodes which judge the technical indexes shown in Table I. In addition, we use several kinds of judgment nodes with different calculation periods for each technical index except golden/dead cross and MACD. Therefore, the number of kinds of judgment nodes is 26 as shown in Table I. The total number of nodes in each individual is 37 including 10 processing nodes, 26 judgment nodes and one start node. The function $ID_j$ of each judgment node is determined randomly at the beginning of the first generation, and changed appropriately by evolution. The initial connections between nodes are also determined randomly.

The parameters on the evolution and the learning are de-
TABLE I
CALCULATION PERIODS OF THE TECHNICAL INDEXES [DAY]

<table>
<thead>
<tr>
<th>technical index</th>
<th>period 1</th>
<th>period 2</th>
<th>period 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate of deviation from moving average (MA)</td>
<td>5</td>
<td>13</td>
<td>26</td>
</tr>
<tr>
<td>RSI</td>
<td>5</td>
<td>13</td>
<td>26</td>
</tr>
<tr>
<td>ROC</td>
<td>5</td>
<td>13</td>
<td>26</td>
</tr>
<tr>
<td>Volume ratio</td>
<td>5</td>
<td>13</td>
<td>26</td>
</tr>
<tr>
<td>RCI</td>
<td>9</td>
<td>18</td>
<td>27</td>
</tr>
<tr>
<td>Stochastics</td>
<td>12</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Psychological line</td>
<td>5</td>
<td>14</td>
<td>—</td>
</tr>
<tr>
<td>rate of deviation from the current price to the highest price (HIGH)</td>
<td>20</td>
<td>60</td>
<td>—</td>
</tr>
<tr>
<td>rate of deviation from the lowest price to the current price (LOW)</td>
<td>20</td>
<td>60</td>
<td>—</td>
</tr>
<tr>
<td>Golden/Dead cross</td>
<td>5 (short term), 26 (long term)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MACD</td>
<td>12 (short term), 26 (long term), 9 (signal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*total number of judgment nodes is 26

TABLE II
SIMULATION CONDITIONS

number of individuals= 600
(mutation: 359, crossover: 240, elite: 1)
number of nodes=37 (judgment node: 26, processing node: 10, start node: 1)
$P_c = 0.5, P_m = 0.02, \text{tournament size}= 5$
$\alpha = 0.01, \alpha_\mu = 0.001, \alpha_\sigma = 0.001, \gamma = 0.8$

Fig. 5. Fitness curve in the training and validation period (Toyota Motor)

Fig. 6. Stock price of Toyota motor and typical buying and selling points in 2004 (test period)

Fig. 7. Change of funds in the test simulation (Toyota motor)

B. Simulation results

First, 600 individuals are evolved for 300 generations using the training data and also evaluated using the validation data every generation. Fig. 5 shows the fitness curves of the simulations of Toyota motor, and each line is the average over 20 independent simulations. The highest line shows the fitness of the best individual for the training data at each generation. The second one shows the mean fitness of all the individuals for the training data. The third one shows the mean fitness of all the individuals for the validation data.

From the figure, we can see that GNP-AC can predict the rise or fall of the stock prices correctly with higher probability for the training data and validation data as the generation goes on. The fitness curves of the other companies have almost the same tendency as those of these two companies.

Next, the test simulation is carried out using one selected individual. The method of selecting one individual for the test simulation is described at the beginning of section III. Table III shows the correct prediction rate and profit and loss in the testing term. The values in Table III are the mean of the 20 independent simulations with different random seeds. For the comparison, the table also shows the results of Buy&Hold.
TABLE III
CORRECT PREDICTION RATE AND PROFITS IN THE TEST SIMULATIONS

<table>
<thead>
<tr>
<th>brand</th>
<th>GNP-AC</th>
<th>Buy&amp;Hold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>correct prediction rate [%]</td>
<td>profit (profit rate [%])</td>
</tr>
<tr>
<td>Toyota Motor</td>
<td>63.1</td>
<td>900,500 (18.0)</td>
</tr>
<tr>
<td>Mitsubishi Estate</td>
<td>54.7</td>
<td>845,350 (16.9)</td>
</tr>
<tr>
<td>Showa Sell Sekiyu K. K.</td>
<td>53.5</td>
<td>395,235 (7.9)</td>
</tr>
<tr>
<td>NEC</td>
<td>56.3</td>
<td>-151,150 (-3.0)</td>
</tr>
<tr>
<td>Fuji Heavy Ind.</td>
<td>52.7</td>
<td>198,050 (4.0)</td>
</tr>
<tr>
<td>Mitsui &amp; Co.</td>
<td>52.8</td>
<td>370,550 (7.4)</td>
</tr>
<tr>
<td>Sony</td>
<td>50.9</td>
<td>170,000 (3.4)</td>
</tr>
<tr>
<td>Tokyo Gas</td>
<td>63.6</td>
<td>250,850 (5.0)</td>
</tr>
<tr>
<td>KDDI</td>
<td>50.6</td>
<td>-104,000 (-2.1)</td>
</tr>
<tr>
<td>Nomura Holdings</td>
<td>58.8</td>
<td>123,135 (2.5)</td>
</tr>
<tr>
<td>Shin-Etsu Chemical Co., Ltd.</td>
<td>49.3</td>
<td>-171,550 (-3.4)</td>
</tr>
<tr>
<td>Nippon Steel Cooperation</td>
<td>52.5</td>
<td>398,150 (8.0)</td>
</tr>
<tr>
<td>Shiseido Company, Limited</td>
<td>53.3</td>
<td>454,250 (9.1)</td>
</tr>
<tr>
<td>The Sumitomo Trust &amp; Banking Co., Ltd.</td>
<td>53.9</td>
<td>1,243,150 (24.9)</td>
</tr>
<tr>
<td>Kyocera Cooperation</td>
<td>53.0</td>
<td>402,650 (8.1)</td>
</tr>
<tr>
<td>Obayashi Cooperation</td>
<td>53.1</td>
<td>883,050 (17.7)</td>
</tr>
<tr>
<td>Meiji Seika Kaisha, Ltd.</td>
<td>48.9</td>
<td>242,750 (4.9)</td>
</tr>
<tr>
<td>Asahi Kasei Cooperation</td>
<td>55.6</td>
<td>155,050 (3.1)</td>
</tr>
<tr>
<td>Fuji Television Network, Incorporated</td>
<td>59.7</td>
<td>-438,300 (-8.8)</td>
</tr>
<tr>
<td>Teijin Limited</td>
<td>48.1</td>
<td>941,000 (18.8)</td>
</tr>
<tr>
<td>mean</td>
<td>54.2</td>
<td>355,436 (7.1)</td>
</tr>
</tbody>
</table>

which is often considered to be a benchmark in trading stocks simulations. Buy&Hold buys as much stocks as possible at the opening of the market on the first day of the simulations, and sells all the stocks at the opening on the last day. From the table, the proposed method can obtain equal or larger profits than Buy&Hold in the trade of 14 brands out of 20, and almost the same profits in the trades of Nippon Steel Cooperation and Kyocera Cooperation. Especially, the stock prices of NEC, Fuji Heavy Ind, KDDI, Nomura Holdings, Shin-Etsu Chemical Co., Ltd. Asahi kasei cooperation and Fuji Television Network, Incorporated are down trend, so Buy&Hold always makes a loss, however the proposed method can obtain profits in three brands and always decrease the loss in other brands.

Fig. 6 shows the change of the price of Toyota motor in the test period and also shows typical buying and selling points by the proposed method. Fig. 7 shows the change of the funds as a result of the trading. From these figures, GNP-AC obtains large profits in the up trend and rarely make a large loss even in the down trend.

Fig. 8 shows the ratio of the nodes used in the test period in order to see which nodes are used and which are most efficient for stock price prediction. The x-axis represents the unique node number. The total number of nodes is 37, and each processing node has a node number (1–10), and each judgment node has a node number (11–36). The x-axis also shows the functions of the nodes, thus if the width of Stochastic2 is wider than the other indexes, then the proportion of Stochastic2 is larger than the other indexes, and GNP-AC selects Stochastic2 by evolution as an important function. The y-axis shows the ratio of the used nodes. From the figure, we can see that the processing nodes are used to predict whether the stock price will rise or fall, and the judgment nodes of ROC, Volume ratio and stochastics are frequently used. Thus it can be said that GNP-AC judges that these nodes are important to predict stock prices. GNP-AC can automatically determine which nodes should be used in the current situation by evolving node functions and connections between nodes, in other words, GNP-AC can optimize the combination of technical indexes used for prediction.

Fig. 9 shows the mean and standard deviation of the probability density function in each node. Some mean values of the processing nodes are positive, and the others are negative because the threshold values should be different depending on the node transition before executing the processing nodes. For example, let us compare the case where judgment nodes of MA1, Volume ratio2 and HIGH are executed in the node transition and the different case where judgment nodes of Stochastics2 and ROC2 are executed. In these cases, the threshold values should be different because GNP-AC judges the different judgment information. While, about half of the standard deviations are converged to small values, and almost all the values are smaller than the initial value (0.1). Therefore, it can be said that Actor-Critic finds appropriate threshold values and make the standard deviations of the threshold values small enough in order to produce them with higher probability.
In this paper, a stock trading model using GNP-AC is proposed. First, a newly defined IMX function is assigned to each technical index to tell GNP-AC whether buying or selling stocks is recommended or not. Second, Actor-Critic learns the shapes and positions of probability density functions to produce appropriate threshold values used to predict correctly whether the stock prices will rise or fall. We carried out simulations using stock price data of 20 brands for three years. From the simulation results, it is clarified that the fitness becomes larger as the generation goes on and the profits obtained in the testing term are better than Buy&Hold in the simulations of 12 brands out of 20 and even when the down trend, the proposed method can obtain profits in some brands and also decrease the loss in the other brands.

There remain some problems to be solved. In this paper, the calculation period of each technical index is fixed in advance. However, to improve the performance of the proposed method, we should develop a new method that can learn appropriate calculation periods. Also, we will evaluate the proposed method comparing with other methods using many data of other brands.

REFERENCES