



## Enhanced decision making mechanism of rule-based genetic network programming for creating stock trading signals

Shingo Mabu <sup>a,\*</sup>, Kotaro Hirasawa <sup>b</sup>, Masanao Obayashi <sup>a</sup>, Takashi Kuremoto <sup>a</sup>

<sup>a</sup> Graduate School of Science and Engineering, Yamaguchi University, Tokiwadai 2-16-1, Ube, Yamaguchi 755-8611, Japan

<sup>b</sup> Information, Production and Systems Research Center, Waseda University, Hibikino 2-2, Kitakyushu, Fukuoka 808-0135, Japan



### ARTICLE INFO

#### Keywords:

Evolutionary computation  
Genetic network programming  
Rule extraction  
Stock trading  
Technical analysis

### ABSTRACT

Evolutionary computation generally aims to create the optimal individual which represents optimal action rules when it is applied to agent systems. Genetic Network Programming (GNP) has been proposed as one of the graph-based evolutionary computations in order to create optimal individuals. GNP with rule accumulation is an extended algorithm of GNP, which extracts a large number of rules throughout the generations and stores them in rule pools, which is different from general evolutionary computations. Concretely, the individuals of GNP with rule accumulation are regarded as evolving rule generators in the training phase and the generated rules in the rule pools are actually used for decision making. In this paper, GNP with rule accumulation is enhanced in terms of its rule extraction and classification abilities for generating stock trading signals considering up and down trends and occurrence frequency of specific buying/selling timing. A large number of buying and selling rules are extracted by the individuals evolved in the training period. Then, a unique classification mechanism is used to appropriately determine whether to buy or sell stocks based on the extracted rules. In the testing simulations, the stock trading is carried out using the extracted rules and it is confirmed that the rule-based trading model shows higher profits than the conventional individual-based trading model.

© 2013 Elsevier Ltd. All rights reserved.

### 1. Introduction

A graph-based evolutionary algorithm called Genetic Network Programming (GNP) (Mabu et al., 2007) has been proposed for creating action-rules in agent systems. GNP has been successfully applied to decision making problems (Hirasawa et al., 2008) due to its distinguished representation abilities such as successive node transitions and reusability of nodes. Evolutionary Programming (EP) (Fogel, 1994) also evolves graph structure (Finite State Machine), but it must consider all the combinations of available inputs from environments to determine the output and next state. Since GNP uses only the necessary information by connecting nodes flexibly, it can deal with Non-Markovian processes with compact program structures.

Evolutionary computation generally evolves individuals in order to increase their fitness, and it obtains the optimal or near optimal individuals which represent action-rules when they are used in agent control systems. Then, the evolved individuals are actually used in the testing. GNP with rule accumulation (GNP-RA) (Wang et al., 2009) also belongs to evolutionary computation field; however, the concept of evolution is different from general

evolutionary computation. Although GNP-RA also evolves many individuals by crossover and mutation until the last generation, the important point is that the action rules (combinations of judgments and actions) obtained by the individuals during the training are stored in rule pools, that is, action rules are accumulated in the rule pools every generation. Therefore, GNP-RA stores many rules throughout the generations, while, in the original GNP, the best individual in the last generation represents action rules. In other words, GNP-RA can store many experiences obtained by the individuals in the training as rules.

Stock market analysis has been one of the most active research fields, where many Machine Learning techniques are adopted. Research on stock price prediction and trading models such as Neural Network (Chang et al., 2009), Support Vector Machines (Huang et al., 2005), rough set theory (Lee et al., 2010) and statistical analysis (Jensen et al., 2004) has been done. In recent years, evolutionary algorithms have been applied to many financial problems such as portfolio optimization (Torrubiano and Suárez, 2010), bankruptcy prediction (Cid et al., 2008), time-series prediction (Iba and Sasaki, 1999). Expert system based approaches for stock trading have been also proposed recently (Dymova et al., 2010; Dymova et al., 2012; Sebastianov and Dymova, 2009).

Generally speaking, methods for predicting stock prices and determining the timing of buying and selling stocks are divided into two groups; one is fundamental analysis which analyzes stock

\* Corresponding author. Tel./fax : +81 836 85 9519.

E-mail address: [mabu@yamaguchi-u.ac.jp](mailto:mabu@yamaguchi-u.ac.jp) (S. Mabu).

prices using the financial statement of each company, the economic trend and so on; the other is technical analysis which numerically analyzes the past movement of stock prices. Generally, the research on stock price prediction and trading models using soft computing belongs to technical analysis, so it determines the timing of buying and selling stocks based on the technical indexes such as rate of deviation, relative strength index, golden cross and so on. This paper also deals with technical analysis.

Stock trading models based on GNP have been proposed as applications (Chen et al., 2008; Mabu et al., 2007). GNP judges the current situation/trend in the stock market based on the technical indexes, then determines the timing of buying and selling stocks. In these models, the best individual in the last generation is used in the testing simulations, so all the stock trading rules are represented by one best individual. However, it is difficult for one individual to adapt to various kinds of stock market situations due to the following reasons. (1) GNP sequentially executes judgment and processing nodes one by one according to the node transition, so all the experiences/rules obtained in the training period cannot be directly used to decide the current action. (2) The number of action rules which one individual can contain is limited due to the limited size of the program structure, while the large size of the structure even causes much computational cost.

In this paper, GNP-RA for stock trading problems is developed, which extracts and stores useful buying and selling rules in the rule pools and the buying and selling actions are determined by a unique matching calculation and decision making techniques. Matching degrees introduced in this paper can tell us the timings of buying/selling. In addition, reinforcement learning algorithm is also used to extract rules. In Mabu et al. (2007), reinforcement learning is applied to GNP in order to enhance the intensified search and online learning abilities. This paper also uses reinforcement learning in order to obtain good action sequences, however, the more important point is to extract a large number of rules by an exploration ability of reinforcement learning with  $\varepsilon$ -greedy policy.

This paper is organized as follows. In the next section, after the basic structure of GNP is explained, how to represent action rules, how to store the rules in the rule pools and how to decide actions are explained. In Section 4, the stock trading problems treated in this paper is described and the simulation results are analyzed. Section 5 is devoted to conclusions.

## 2. Review of genetic network programming with reinforcement learning

GNP was originally developed to create programs that work in dynamic environments. In the original GNP, an individual is represented by a directed graph structure, evolved by crossover and mutation, and the node transition rules are learned by reinforcement learning. In this section, the phenotype and genotype representations and the learning and evolution processes are reviewed.

### 2.1. Phenotype representation

#### 2.1.1. Features of the directed graph structure

The phenotype representation of GNP is a directed graph structure shown in Fig. 1. It consists of a number of judgment nodes and processing nodes and one start node. The number of nodes is determined in advance depending on the complexity of the problems. Each judgment node has a if-then branch decision function and each processing node has an action function. For example, the judgment nodes in Fig. 1 examine technical indexes used in the stock market analysis. Node 1 examines Relative Strength Index (RSI) and selects a branch corresponding to the judgment result.

The processing nodes determine buying and selling actions. The role of the start node is to determine the first node to be executed. Therefore, the node transition starts from the start node, and judgment and action sequence is realized by the connections between nodes and judgment results.

The nodes in GNP can be repeatedly used due to the graph structure, so complex programs can be created by compact structures, which contributes to the efficiency of the evolution (Mabu et al., 2007). In addition, only the necessary nodes for solving the target problem can be connected and used, so GNP does not need the complete information of the environment unlike Finite State Machines.

#### 2.1.2. Subnodes in judgment and processing nodes

Extended GNP with reinforcement learning (GNP-RL) (Mabu et al., 2007) has subnodes in each judgment and processing node (Fig. 2). In the original GNP, one function is assigned to each node and executed when the node is visited. On the other hand, in GNP-RL, several functions (two functions in Fig. 2) are assigned as subnodes and one of them is selected and executed by a reinforcement learning algorithm. Therefore, reinforcement learning can select better function/subnode for each node and the route of the node transition can be optimized. Since this paper deals with a stock trading problem, each subnode in a judgment node has a function to examine a technical index and that in a processing node has a function to buy or sell stocks.

### 2.2. Genotype representation

The graph structure is realized by the combination of gene structures shown in Fig. 3.  $NT_i$  shows the node type of node  $i$ .  $NT_i = 0$  means start node,  $NT_i = 1$  means judgment node, and  $NT_i = 2$  means processing node.  $d_i$  is a time delay which shows the time spent on the execution of node  $i$ . In this paper,  $d_i$  of judgment node is set at 1 and that of processing node is set at 5. The time delay is useful to fix the maximum number of nodes to be executed in each action step (in this paper, an action step is one trading day). If five time units (time delays) are assigned to one action step, the action step ends when the time units used by the node transition become five or exceed five. That is, GNP can execute "at most five judgment nodes" or "less than five judgments nodes and one processing node" in one step. The time delay is also effective for avoiding the deadlock of the node transition due to the judgment loop. Even if the judgment loop occurs and GNP cannot execute any processing, each action step will end with five time units and finally GNP ends its trial without taking any actions, which leads to the low fitness and such GNP will not be selected for the next generation.

$Q_1, Q_2, \dots$  are  $Q$  values (Sutton and Barto, 1998) assigned to subnodes in node  $i$ .  $ID_{i1}, ID_{i2}, \dots$  are the node functions of the subnodes. A  $Q$  value estimates the sum of the discounted rewards obtained in the future. The contents of the functions are described in the node function library. For example,  $NT_i = 1$  and  $ID_{i2} = 2$  show the function of subnode 2 in node  $i$  is  $J_2$ . Judgment nodes examine the technical indexes in the stock market, e.g., RSI, ROC and so on, (The technical indexes used in this paper are described later in Table 4). Each technical index is assigned to each judgment function listed in the library. Processing nodes decide buying and selling actions, so  $P_1$  and  $P_2$  in Fig. 3 show buying and selling actions, respectively.

$C_{i1}^A, C_{i1}^B, \dots$  and  $C_{i2}^A, C_{i2}^B, \dots$  show the next node numbers connected from node  $i$ . For example, subnode 1 in node  $i$  is connected to  $C_{i1}^A, C_{i1}^B, \dots$ . The superscripts  $A$  and  $B$  correspond to the judgment results, so if the judgment result is  $A$  at subnode 1, the next node becomes  $C_{i1}^A$ .

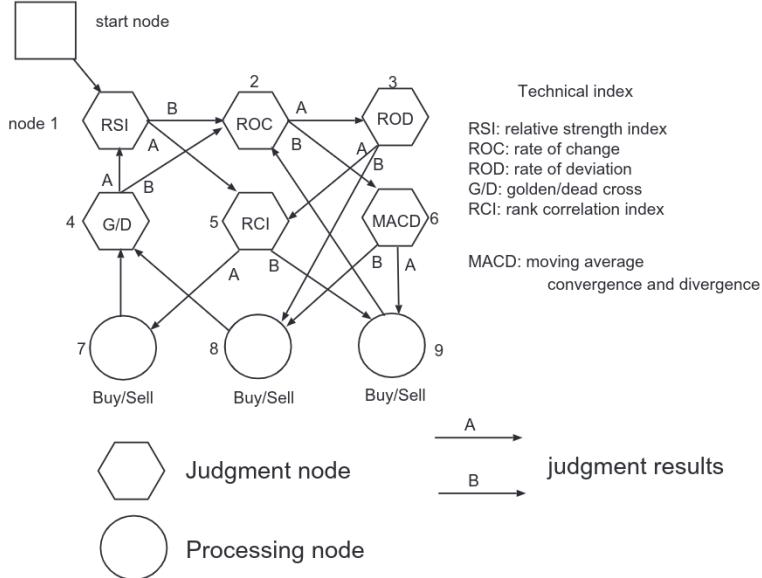


Fig. 1. Basic structure of GNP.

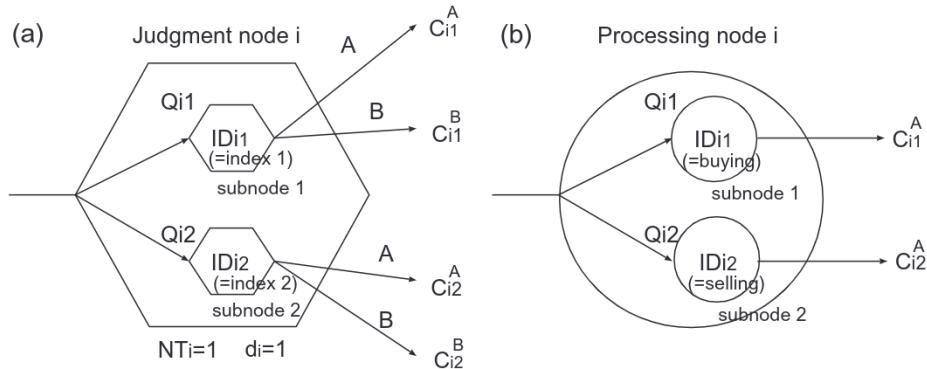


Fig. 2. Judgment node and Processing node structures.

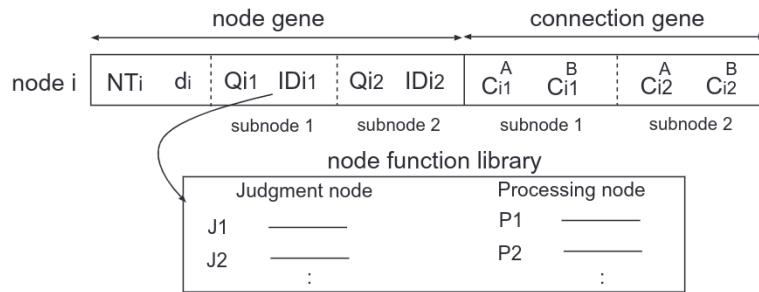


Fig. 3. Genotype representation of GNP.

### 2.3. Node transition rule and its learning algorithm

In this paper, Sarsa (Sutton and Barto, 1998) which is one of reinforcement learning algorithms is used for the learning of GNP-RL (We call GNP-RL based on Sarsa “GNP-Sarsa”). The reason why we select Sarsa is that on-policy algorithm (Sarsa) is effective to optimize state transitions considering the effect of  $\epsilon$ -greedy policy used in this paper. The node transition of GNP-Sarsa is carried out as follows.

After the start node, when the current node is a judgment node, one of the subnodes is selected according to  $\epsilon$ -greedy policy, i.e.,

the subnode with the largest  $Q$  value is selected with the probability of  $1 - \epsilon$ , otherwise one subnode is randomly selected. After executing the function of the selected subnode, the current node is transferred to the next node according to the judgment result and connection. In Fig. 2 (a), suppose subnode 1 is selected, then one of the connections (A or B) is selected according to the judgment result. When the current node is a processing node, one of the subnodes is selected by the same way as judgment nodes. After executing the action of the selected subnode, the next node is determined by the connection from the subnode. The GNP-Sarsa structure corresponds to the basic reinforcement learning

framework. Each node is defined as “state” and the selection of a subnode is defined as “action” in the reinforcement learning framework and the optimal state transition is found by Sarsa. Every node transition,  $Q$  value is updated by the following equation.

$$Q_{ip} \leftarrow Q_{ip} + \alpha(r + \gamma Q_{jq} - Q_{ip}), \quad (1)$$

where  $Q_{ip}$  shows the  $Q$  value of the selected subnode  $p$  at node  $i$ ,  $Q_{jq}$  shows the  $Q$  value of the selected subnode  $q$  at the next node  $j$  and  $r$  is a reward obtained after executing node  $i$ .  $\alpha( \in [0,1.0] )$  is a learning rate,  $\gamma( \in [0,1.0] )$  is a discount rate.

#### 2.4. Genetic operation

The genetic operation is based on elite selection, tournament selection, crossover and mutation. The fitness in this paper is a profit earned in the training period. Here, the procedure of crossover and mutation (Mabu et al., 2007) is explained.

- Crossover: Two individuals are selected as parents by tournament selection. Two new offspring are generated from the parents by exchanging the genetic codes. In detail, each node number  $i$  is selected as a crossover node number with the probability of  $P_c$ , then the genes of the selected nodes are swapped between the parents.
- Mutation: One individual is selected by tournament selection and a new individual is generated by the following operations;
  - (1) Each connection gene ( $C_{i1}^A, C_{i1}^B, \dots, C_{i2}^A, C_{i2}^B, \dots$ ) is selected with the probability of  $P_m$  and changed to another node number.
  - (2) Each node function  $ID_i$  is selected with the probability of  $P_m$  and changed to another one.

### 3. Stock trading signal generation by GNP with Rule Accumulation (GNP-RA)

GNP-RA for the stock trading system consists of three parts: (1) evolution of rule generators (individuals) by GNP-Sarsa; (2) rule accumulation in the rule pools with occurrence frequency; (3) matching calculation to generate trading signals.

GNP-Sarsa evolves many individuals by efficiently combining evolution and Sarsa. Note that GNP-Sarsa is a part of GNP-RA and regarded as an evolving rule generator.

The main points of the proposed method are to enhance GNP-RA in terms of its rule extraction and its decision making abilities, and confirm the improvement of the fitness comparing to conventional GNP methods. In detail, a new rule extraction method using GNP-Sarsa is proposed in the training, where each extracted rule has its own occurrence frequency information which shows the importance of the rule. On the other hand, the extracted rules are separated into several rule pools considering up and down trends in the testing, and the unique matching calculation mechanism is introduced to create buying and selling signals.

#### 3.1. A structure of GNP with Sarsa for stock trading rule generation

**Fig. 4** shows the flowchart of the evolution and rule extraction. As the same way as general evolutionary computation, the individuals are evolved to increase fitness. However, during the trading in the evolution/training phase, buying and selling rules obtained by the elite individuals are stored in the rule pools every generation. Therefore, a large number of useful rules can be stored.

GNP-Sarsa uses graph structures to represent individuals as shown in **Fig. 1** and each individual creates a sequence of judgments and processing by the node transition. As described before, a judgment node examines a technical index assigned to the node

and selects one of the branches based on the judgment result. A processing node determines buying or selling action. Therefore, the combinations of judgment and processing nodes represent stock trading rules.

As shown in **Fig. 2**, each judgment node contains two subnodes which have their own technical indexes (index1 and index2) selected randomly from all the available indexes and have their  $Q$  values ( $Q_{i1}$  and  $Q_{i2}$ ). Each processing node also contains two subnodes which have their own actions selected randomly from buying/selling and have their  $Q$  values.

#### 3.2. Rule accumulation

##### 3.2.1. Rule extraction by the elite individuals

General evolutionary computation obtains the elite individual in the last generation and it becomes the solution for the problem. However, the aim of GNP-RA is not to obtain good individuals, but to accumulate a large number of good extracted rules in the rule pools every generation. Therefore, the rule accumulation is carried out throughout the generations.

In this paper, the rules are extracted by the elite individuals of GNP-Sarsa every generation. The node transitions of the elite individuals are saved and rules are extracted as shown in **Fig. 5**. A rule consists of successive judgment functions with their results (antecedent part) and the resulting processing, i.e., buying or selling (consequent part). The rules are stored in the buying rule pool and selling rule pool separately depending on the resulting actions of the rules. If the same rules which have been already stored in the rule pools are extracted by the elite individuals in other generations, such rules are not stored in the rule pools again.

In order to judge the current stock market situation accurately, each rule pool is divided into two pools, i.e., up and down trend pools. The up and down trends are defined by the condition of moving averages (MAs) of stock prices as follows,

$$\begin{aligned} &\text{if(MA of 25 days at current day } t \\ &- (\text{MA of 25 days at day } t - 25) \geq 0, \text{ Up trend} \\ &\text{otherwise, Down trend} \end{aligned} \quad (2)$$

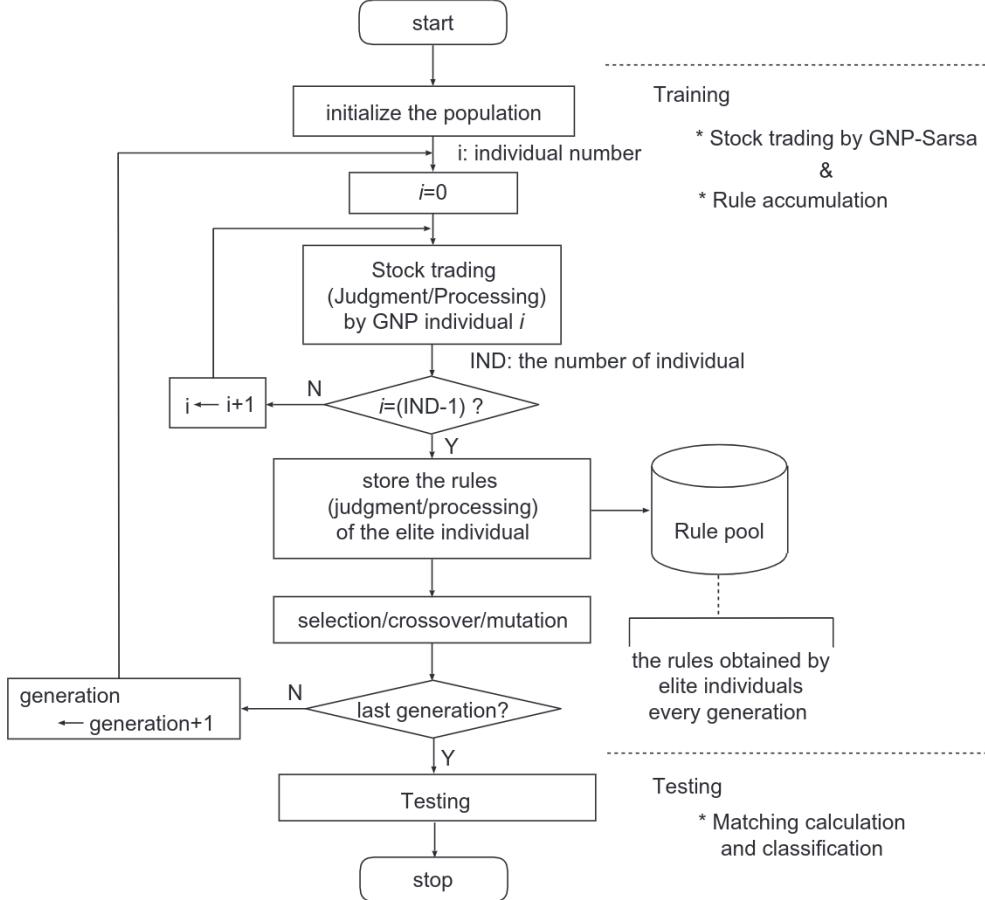
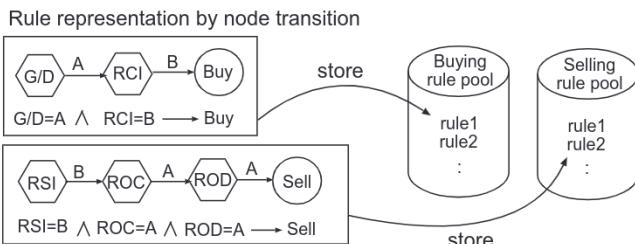
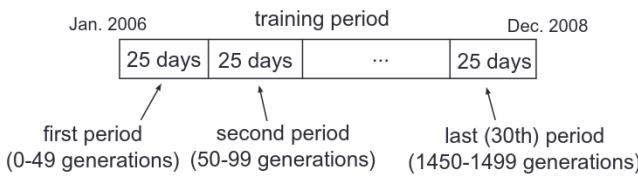
If rules are extracted when the market trend is up in the training period, they are stored in the up trend pools. If the trend is down, they are stored in the down trend pools. Therefore, four rule pools can be created, i.e., buying rule pool for up trend, that for down trend, selling rule pool for up trend and that for down trend.

##### 3.2.2. Training period division

Stock prices of three years are used as training data and those of the following one year as testing data. In order to extract a large number of good rules efficiently, the training period is divided into many periods. In detail, the training period is divided every 25-day, so, for example, the training data in 2006–2008 in Tokyo stock exchange in Japan can be divided into 30 periods (**Fig. 6**).

First, GNP-Sarsa evolves individuals for 50 generations using the data of the first period as training data, and the rules extracted by the elite individuals are stored in the rule pools. Second, the individuals are evolved for 50 generations using the data of the second period. The second period is a new/independent simulation which begins with initial funds and without having any stocks. This procedure is repeated until the training in the last period finishes.

The most important point in this method is to count the number of times each rule is extracted. For example, a certain rule  $r$  is extracted in five periods out of 30, the count  $c$  of rule  $r$  becomes “5”. In other words, rules which contribute to the fitness several times in different periods can be regarded as general rules which avoid the overfitting to the rare situation in the training. If a threshold is set at two, only the rules extracted at least two times are used

**Fig. 4.** Flow of evolution and rule extraction.**Fig. 5.** Rule representation.**Fig. 6.** An example of training period division.

for the decision making in the testing. In this paper, the rules extracted more than once ( $c > 1$ ) are used.

### 3.3. Trading signal generation by matching calculation in the testing

#### 3.3.1. Definition of matching degree

First, the matching degree of stock data  $d$  on a certain day with rule  $r$  is defined as follows.

$$Match_k(d, r) = \frac{N_k(d, r)}{N_k(r)}, \quad (3)$$

where  $N_k(d, r)$  is the number of matched judgment results of data  $d$  with the antecedent part of rule  $r$  in class  $k$ <sup>1</sup>,  $N_k(r)$  is the number of judgments (technical indexes) in the antecedent part of rule  $r$  in class  $k$ .

Then, the average matching degree of data  $d$  with all the rules selected from the rule pool  $k$  is calculated as follows.

$$m_k(d) = \frac{1}{|R_k|} \sum_{r \in R_k} Match_k(d, r), \quad (4)$$

where  $R_k$  shows a set of rules with  $c > 1$  in class  $k$ .

#### 3.3.2. Buying and selling signal generation

For each training data  $d$ , the average matching degree with the selected rules in class  $k$  is calculated by

$$m_k(d) = \frac{1}{|R_k|} \sum_{r \in R_k} Match_k(d, r), d \in D_{train}, \quad (5)$$

where,  $D_{train}$  shows a set of training data.

Then, the mean and standard deviation of the average matching degrees of all the training data are calculated as follows,

<sup>1</sup>  $k$  shows a rule pool,  $k \in \{buy_{up}, buy_{down}, sell_{up}, sell_{down}\}$ , where  $buy_{up}$  means the rule pool for buying in up trends,  $buy_{down}$  means that in down trends,  $sell_{up}$  means the rule pool for selling in up trends and  $sell_{down}$  means that in down trends.

$$\text{Mean}_k = \frac{1}{|D_{\text{train}}|} \sum_{d \in D_{\text{train}}} m_k(d), \quad (6)$$

$$\text{Std}_k = \sqrt{\frac{1}{|D_{\text{train}}|} \sum_{d \in D_{\text{train}}} (m_k(d) - \text{Mean}_k)^2}. \quad (7)$$

In the testing, a target class, i.e., the action which can be taken at the current situation, is firstly judged as follows.

#### Target class selection

```

if no stocks are held
  if up trend
    k = buyup
  else if down trend
    k = buydown
  end if
else if stocks are held
  if up trend
    k = sellup
  else if down trend
    k = selldown
  end if
end if

```

After the target class is selected, the following decision will be done.

#### Buying and Selling decision First, calculate the average

matching degree  $m_k(d)$ , where  $d$  is a stock data on the decision making day in the testing.

Then,

```

If mk(d) ≥ Meank + ηkStdk
  If k = buyup or buydown
    buy stocks using all the funds
  else if k = sellup or selldown
    sell all the stocks
  end if
else
  no action
end if

```

$\eta_k$  (=0.1) is a coefficient. There is a trade off between large and small  $\eta_k$ . If it is large, situations which match many rules are considered as buying/selling timings. However, it might miss some trading chances to buy/sell stocks, and trading times become low due to the severe threshold. Small  $\eta_k$  makes trades often, but it might buy/sell stocks at the timings which are not really good timings. After examining several settings, we used  $\eta_k = 0.1$  which showed relatively good performance.

## 4. Simulations

As described in the previous sections, the trading rules are extracted in the training period and the buying and selling signals are generated by the extracted rules in the testing period. Therefore, in this section, the effectiveness of the rule extraction and the trading results in the testing simulations are analyzed.

### 4.1. A stock trading problem

A stock trading problem dealt with in this paper is described as follows.

(1) Select 16 stocks from the first section of Tokyo Exchange in Japan. These are Japanese leading companies with large market capitalization and relatively high liquidity, and also selected from various fields such as car, metal, real estate etc. considering domestic and foreign demand.

(2) In order to extract good rules for a specific stock, one population of GNP-Sarsa is evolved for maximizing the profit for each stock. The initial funds for buying the target stock is five million Japanese Yen.

(3) The available stock information is the past opening price, closing price, high price, low price and volume of trades on each day. Thus, technical indexes can be calculated by these stock prices and volumes. The proposed stock trading model concentrates on technical analysis based on the basic and simple indicators derived by the above information. The aim of the proposed method is to obtain a large number of good trading rules by combining relatively simple indicators by evolution, which leads to the effective indicators automatically.

The training and testing periods are as follows, where the rules are extracted using the training data of three years and the testing simulations based on the extracted rules are carried out using the data of the following year.

- (a) Training: Jan. 4, 2006 – Dec. 30, 2008 (738 days)
- (b) Testing: Jan. 5, 2009 – Dec. 30, 2009 (243 days)

The training and testing data include both up and down trends, so they are appropriate periods for the performance evaluation. Because the companies are selected from various fields, their stock prices show various trends.

**Table 1**  
Transaction cost.

Buying/selling price [yen]	Transaction cost [yen]
1–100,000	145
100,001–200,000	194
200,001–500,000	358
500,001–1,000,000	639
1,000,001–1,500,000	764
1,500,001–30,000,000	1,209
30,000,000–	1,277

**Table 2**  
Simulation conditions.

The number of individuals = 301
(mutation: 179, crossover: 120, elite: 1)
The number of nodes = 61 (judgment node: 40, processing node: 20, start node: 1)
the number of generations = 1500
Crossover rate $P_c = 0.5$ , mutation rate $P_m = 0.02$
$\alpha = 0.1, \gamma = 0.4, \varepsilon = 0.1$

**Table 3**  
Calculation periods of the technical indexes [day].

Technical index	Period 1	Period 2	Period 3
Rate of deviation	5	10	25
RSI	5	10	25
ROC	5	10	25
Volume ratio	5	10	25
Stochastics	5	10	25
RCI	5	10	25
Golden/Dead cross	5 (short term), 25 (long term)		
MACD	5 (short term), 25 (long term), 9 (signal)		
candle chart		---	

**Table 4**

Judgment results for each technical index and candle patterns.

Technical index	Judgment result				
	A	B	C	D	E
1 Rate of deviation	[−1, −0.1]	(−0.1, −0.05]	(−0.05, 0.05)	[0.05, 0.1)	[0.1, ∞)
2 RSI	[0, 0.2]	(0.2, 0.8)	[0.8, 1]	−	−
3 ROC	[0, 0.9]	[0.9, 1.1]	(1.1, ∞)	−	−
4 Volume Ratio	[0, 0.3]	[0.3, 0.7]	(0.7, 1]	−	−
5 Stochastics	[0, 0.3]	(0.3, 0.7)	[0.7, 1]	−	−
6 RCI	[−1, −0.7]	(−0.7, 0.7)	[0.7, 1]	−	−
7 Golden/Dead Cross	Golden cross	Dead cross	the other	−	−
8 MACD	Golden cross	Dead cross	the other	−	−
9 Candle chart			each pattern in Fig. 7 corresponds to a judgment result		

- (4) The decisions of buying and selling are made every trading day before the opening of the market.  
(5) This problem deals with the spot trading that takes a long position only (no margin trading), therefore the reward for Sarsa learning is given when GNP-Sarsa sells stocks.

$$\text{reward} = \text{selling price} - \text{buying price}, \quad (8)$$

which shows the profit of one set of trades, i.e., one time buying and one time selling.

Fitness function is,

$$\text{Fitness} = \text{Total profit in the training period}. \quad (9)$$

The transaction costs are considered as shown in Table 1 which refers to (C) SBI Securities Co., Ltd (<https://www.sbisec.co.jp/>), one of the Japanese Internet securities companies. The transaction cost is calculated based on the buying or selling price every time.

#### 4.2. Simulation conditions

The parameters used in the simulations are shown in Table 2. These parameters are selected experimentally which show good performances in the training in terms of the fitness, computational time and the balance of exploitation and exploration. The total number of individuals is 301, which means that, every generation, 179 individuals are created by mutation, 120 individuals by crossover and one elite individual is preserved. The total number of nodes is 61 including 40 judgment nodes, 20 processing nodes and one start node. When the population is initialized, randomly selected judgment and processing functions are assigned to the judgment and processing nodes, respectively, and the connections

between nodes are also determined randomly. Q values are initialized at zero.

Table 3 shows the technical indexes used in the simulations and those indexes, except golden/dead cross, MACD and candle chart, are calculated using three calculation periods. For example, Rate of deviation (ROD) of five days indicates the deviation of the current stock price from the moving average of five days. Similarly, ROD of 10 days and 25 days are calculated. Therefore, three judgment functions are used for one kind of technical index. Considering the other technical indexes, the total number of judgment functions become (21 = 6 × 3 + 3).

Table 4 shows the judgment results of each technical index, for example, if the value of rate of deviation is 0.08, the judgment result becomes D which corresponds to the branch to be selected. In the case of candle chart judgment, the judgment result is determined by the eight candle patterns shown in Fig. 7.

#### 4.3. Simulation results

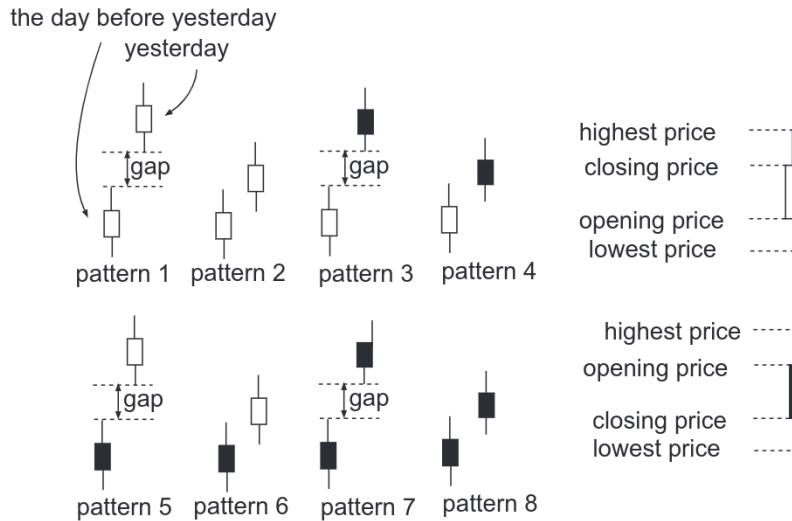
##### 4.3.1. Training results

In the training, the evolution of the individuals and the extraction of rules are carried out simultaneously. As the generation goes on, a large number of buying and selling rules are extracted from the elite individuals as shown in Fig. 8 (a) which shows the average number of extracted rules for Toyota motor over 30 independent simulations. In this case, 651.4 buying rules for up trends, 637.4 buying rules for down trends, 623.2 selling rules for up trends and 649.0 selling rules for down trends are extracted. The training period contains both up and down trends, so we can see that the rules for up trends are extracted more than those for down trends until 600th generation, but after that, those for down trend are extracted more.

**Table 5**

Profits and their rates in the test simulations.

Stock	GNP-RA profit [yen] (profit rate [%])	GNP-RA w/o Sarsa profit [yen] (profit rate [%])	GNP-Sarsa profit [yen] (profit rate [%])	Buy&Hold profit [yen] (profit rate [%])
Toyota Motor	1,158,526 (23.2)	951,840 (19.0)	565,747 (11.3)	1,421,582 (28.4)
Mitsubishi Estate	1,175,337 (23.5)	662,089 (13.2)	25,249 (0.5)	−149,418 (−3.0)
Showa Shell	−487,650 (−9.8)	−436,057 (−8.7)	−503,628 (−10.1)	−691,218 (−13.8)
East Japan Railway	347,103 (6.9)	81,208 (1.6)	−368,214 (−7.4)	−782,418 (−15.6)
NEC	216,322 (4.3)	−365,577 (−7.3)	81,045 (1.6)	−1,106,418 (−22.1)
Fuji Heavy Ind.	1,399,760 (28.0)	987,817 (19.8)	2,114,878 (42.3)	3,877,582 (77.6)
Sekisui House	352,551 (7.1)	154,306 (3.1)	266,611 (5.3)	301,382 (6.0)
Mitsui & Co.	1,637,648 (32.8)	1,637,531 (32.8)	546,316 (10.9)	1,842,582 (36.9)
Sony	2,603,137 (52.1)	1,572,360 (31.4)	908,489 (18.2)	1,760,082 (35.2)
Tokyo Gas	−997,783 (−20.0)	−761,423 (−15.2)	−654,319 (−13.1)	−892,418 (−17.8)
KDDI	−626,097 (−12.5)	−672,499 (−13.4)	−671,271 (−13.4)	−968,418 (−19.4)
Tokyo Electric Power	−730,026 (−14.6)	−649,505 (−13.0)	−925,371 (−18.5)	−1,098,418 (−22.0)
Daiwa House	1,229,480 (24.6)	948,587 (19.0)	−195,010 (−3.9)	597,582 (12.0)
Nomura Holdings	896,856 (17.9)	319,610 (6.4)	−270,347 (−5.4)	−489,918 (−9.8)
Shin-Etsu Chemical Co., Ltd.	1,046,660 (20.9)	882,250 (17.6)	361,285 (7.2)	1,130,582 (22.6)
Nippon Steel Cooperation	1,602,297 (32.0)	1,014,374 (20.3)	928,264 (18.6)	1,341,582 (26.8)
mean	676,507 (13.5)	395,432 (7.9)	138,108 (2.8)	380,895 (7.6)



**Fig. 7.** Candle patterns.

One of the advantages of using Sarsa in the training is to enhance the exploration ability and extract a large number of rules. Fig. 8 (b) shows the number of rules for Toyota motor extracted by GNP-RA without Sarsa averaged over 30 independent simulations, where 191.4 buying rules for up trends, 203.5 buying rules for down trends, 184.5 selling rules for up trends and 206.0 selling rules for down trends are extracted. Comparing Fig. 8 (a) with Fig. 8 (b), we can see that the number of extracted rules by GNP-RA is larger than that without Sarsa. This paper focuses on extracting a large number of rules in order to cover many situations, so the selection of important rules from all the extracted rules is a next research topic in the future. In the next subsection, the comparison between these two methods in terms of the testing results is also given.

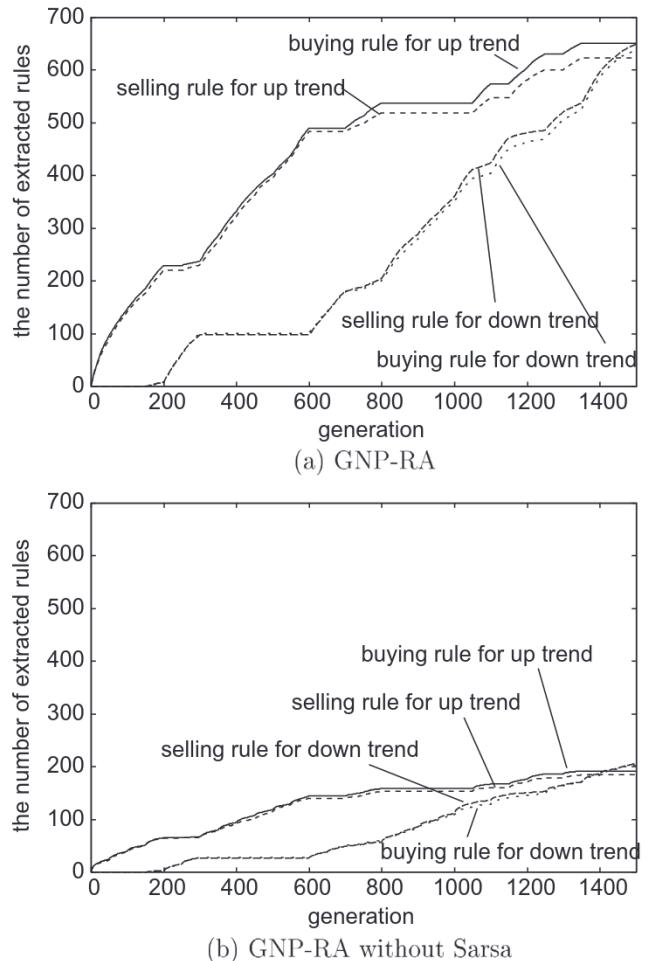
#### 4.3.2. Testing results

In the testing, the buying and selling actions are determined by the matching calculation introduced in Section 3.3.2.

Table 5 shows the testing results averaged over 30 independent simulations, where the profits and profit rates for each stock obtained by GNP-RA, GNP-RA without Sarsa, GNP-Sarsa and Buy&Hold are shown. GNP-Sarsa does not have the rule accumulation mechanism, but uses the elite individual in the last generation for the testing. Buy&Hold buys stocks using all the money in hand on the first day in the testing period and sells all the stocks on the last day, which is often regarded as a benchmark of stock trading simulations. The effectiveness of the combination of Sarsa and RA is focused, therefore the comparisons between the above methods are carried out.

From Table 5, GNP-RA shows the best average profit among these four methods. GNP-RA makes larger profits than GNP-RA without Sarsa in 13 cases out of 16, which shows that the rule extraction by the combination of evolution and Sarsa works effectively. GNP-RA also makes larger profits than GNP-Sarsa in 14 cases. GNP-Sarsa needs to represent trading rules by the limited size of the structure and it is difficult for the node transition of GNP-Sarsa to consider all the experienced situations for the decision making. Some stocks have very large up trends, but GNP-Sarsa cannot make large profits during these periods although it avoids large losses in the trades of other stocks with severe down trends. Finally, GNP-RA makes larger profits than Buy&Hold in 11 cases. In some cases with strong up trends, Buy&Hold has advantages over GNP-RA and other methods. Especially, in the case of Fuji Heavy Ind., Buy&Hold gains the profit rate of 77.6%, which is much larger profit than that gained by the other methods and gives large

weight on the average profit. However, in the cases of down trends, it produces large losses although GNP-RA can avoid large losses or, rather, make profits. The trading signals by GNP-RA are based on many rules obtained in the training period, therefore GNP-RA takes various cases experienced in the training into consideration. As a result, it has an advantage in terms of the stability without taking large losses.

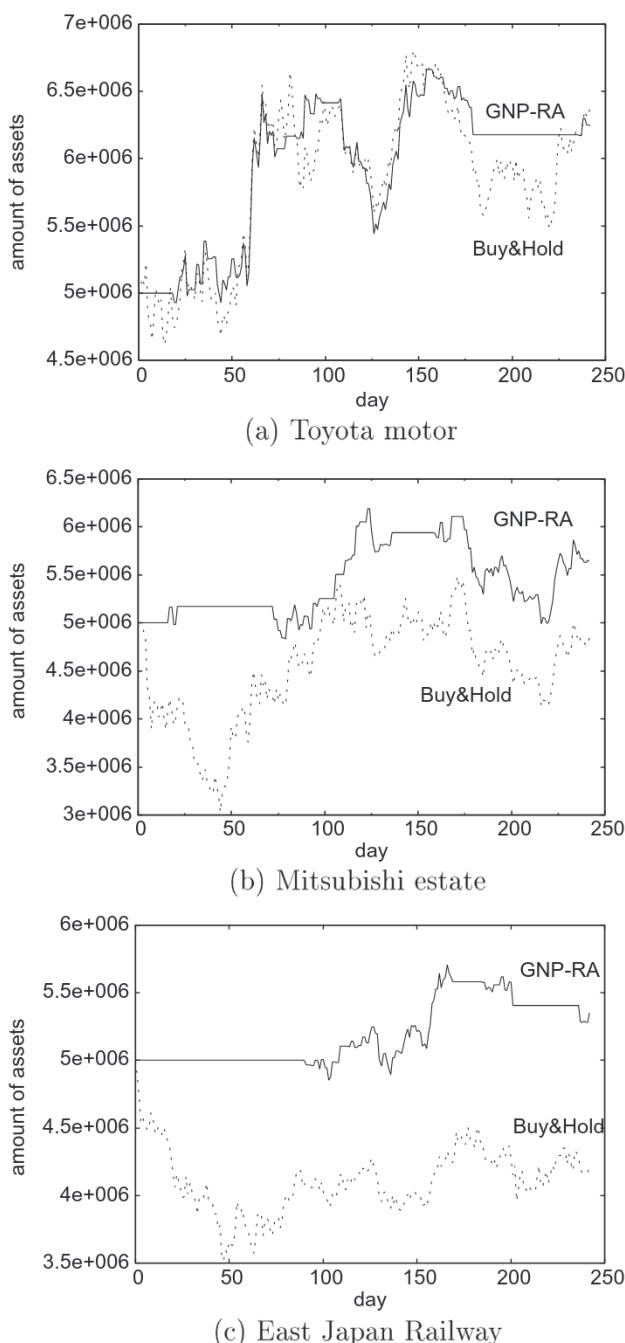


**Fig. 8.** The number of rules for Toyota motor extracted by GNP-RA in the training.

Next, some trading processes are analyzed to find the characteristics of GNP-RA. Fig. 9 (a) shows a typical example of the changes of the amount of assets in the trade of Toyota motor by GNP-RA, where the line of Buy&Hold shows the stock price changes in the testing period. Both GNP-RA and Buy&Hold can get profits, but Buy&Hold gets a little higher profit than GNP-RA because of the overall up trend. GNP-RA sometimes sells stocks before the local down trends and avoid large losses, but it cannot increase the profit so much in the local up trends. Fig. 9 (b) shows the case of Mitsubishi Estate. Buy&Hold makes a large loss at the beginning of the testing period due to the down trend, and the funds on the final trading day is lower than the initial funds. On the other hand, GNP-RA does not make trades in the severe down trend and makes profit in the up trend, so the final funds is larger than the initial funds. In some up trend periods, GNP-RA cannot get much profit,

so there is still room for improvement in the generation of trading signals in the case of up trends. For example, the system could be enhanced by introducing learning algorithms of trend judgment although this paper judges the trend based on the moving average of 25 days. Fig. 9 (c) shows the case of East Japan Railway. GNP-RA also avoids making trades in the severe down trend at the beginning of the testing period and makes profit using some local up trends. As a whole, it can be said that GNP-RA automatically extracts appropriate trading rules for up and down trends, respectively, because reasonable trading actions are taken by GNP-RA in both trends. In the trading of the other stocks, almost the same tendency can be found.

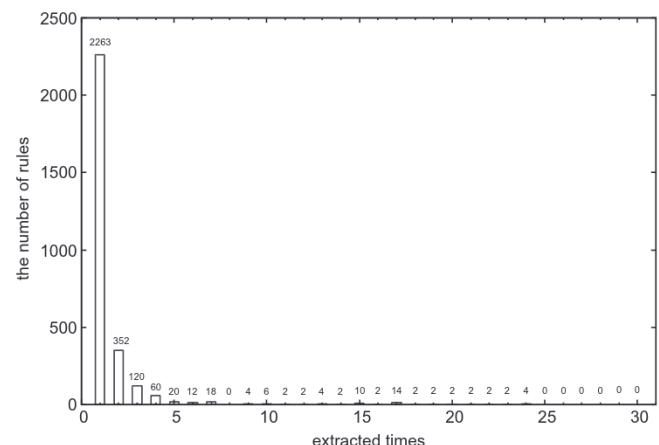
In this paper, the performance is evaluated by profit rates. Real stock trading systems have to be evaluated by many different parameters, such as the number of winning and losing trades, max draw-down, gross profit, gross loss and so on, considering many kinds of situations. However, the main points of this paper are to enhance the rule extraction and decision making algorithms of GNP, and show the superiority of the proposed method comparing to the conventional GNP methods. In order to make the target problem clear, the profit rate is defined as the fitness, and the fitness improvement is evaluated to confirm the efficiency of the enhancement. However, it is one of the remaining works to be studied in the future to consider many kinds of parameters for creating more realistic stock trading models. In addition, this paper deals with 16 stocks for analyzing the results of each stock in detail, but in the future, larger dataset with much more stocks and more statistical tests should be considered for more analysis.



**Fig. 9.** Typical example of the changes of the amount of assets in the testing period.

#### 4.3.3. Discussion on the importance of extracted rules

In the testing simulations, the rules extracted more than once ( $c > 1$ ) are used to make trading signals. Then, the distribution of the number of rules with respect to the extracted times is shown in Fig. 10, and the discussion is done on which rules are used. From Fig. 10, we can see that 2,263 rules are extracted in only one period among 30 periods, 352 rules are in two periods, the rules extracted more times decrease as the extracted times become larger, and no rules are extracted more than 24 times. In this paper, the condition  $c > 1$  is selected considering the balance between the number of general rules and the enough number of rules which is sufficiently cover the various kinds of stock market situations. The rules extracted only once have relatively small reliability, so they were removed from the rule pools, however, the rules extracted more than once were used to keep variations of the rules. If the condition  $c > 2$  is adopted, the ratio of the rule pruning becomes a little too large.



**Fig. 10.** Distribution of the number of rules with respect to the extracted times (Nippon Steel Cooperation).

However, there is a possibility that some of the rules extracted only once are actually important, and the rules extracted many times are not so important. Therefore, in the future, the data mining approach will be considered in order to select the important rules using statistical analysis, e.g., support, confidence and chi-square values.

## 5. Conclusions

GNP-RA extracts a large number of rules throughout the generations, which is different from general evolutionary computation. The advantage of GNP-RA is to remember various situations occurred in the training period and store them as rules in the rule pools. This paper enhances GNP-RA in order to apply it to stock trading systems. We can prepare various rule pools based on up and down trends, and buying and selling timings considering occurrence frequency, therefore, more flexible and practical decision making system can be created. The combination of evolution and Sarsa has been already proposed in the previous research; however, GNP-RA proposed in this paper can take advantage of the exploration ability of  $\epsilon$ -greedy policy used in Sarsa and generate more rules efficiently. In the stock trading simulations, GNP-RA can make larger profits than GNP-RA without Sarsa, GNP-Sarsa and Buy&Hold due to the above characteristics. However, GNP-RA sometimes misses the chances to increase the profits in the up trends, therefore the appropriate trend judgment mechanism is necessary. The rule pools of up and down trends are prepared based on the moving average of 25 days, but the automatic trend division could enhance the system. In addition, a data mining approach could evaluate the importance of the extracted rules and improve the quality of the rule pools.

## References

- Chang, P. C., Fan, C. Y., & Liu, C. H. (2009). Integrating a piecewise linear representation method and a neural network model for stock trading points prediction. *IEEE Transactions on System, Man, Cybernetics C*, 39(5), 80–92.
- Chen, Y., Mabu, S., Shimada, K., & Hirasawa, K. (2008). Trading rules on stock markets using genetic network programming with sarsa learning. *Journal of Advanced Computational Intelligence and Intelligent Informatic*, 12(4), 383–392.
- Cid, E. A., Castillo, P. A., Esparcia, A., Sharman, K., Merelo, J. J., Prieto, A., Mora, A. M., & Laredo J. L. J. (2008). Comparing multiobjective evolutionary ensembles for minimizing type i and ii errors for bankruptcy prediction. In *Proc. of the IEEE World Congress on Computational Intelligence* (pp. 2902–2908).
- Dymova, L., Sevastianov, P., & Bartosiewicz, P. (2010). A new approach to the rule-base evidential reasoning: Stock trading expert system application. *Expert Systems with Applications*, 37(8), 5564–5576.
- Dymova, L., Sevastianov, P., & Kaczmarek, K. (2012). A stock trading expert system based on the rule-base evidential reasoning using Level 2 Quotes. *Expert Systems with Applications*, 39(8), 7150–7157.
- Fogel, David B. (1994). An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1), 3–14.
- Hirasawa, K., Eguchi, T., Zhou, J., Yu, L., & Markon, S. (2008). A double-deck elevator group supervisory control system using genetic network programming. *IEEE Transaction System, Man, Cybernetics C*, 38(4), 535–550.
- Huang, W., Nakamori, Y., & Wang, S. Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers and Operations Research*, 32(10), 2513–2522.
- Iba, H., & Sasaki, T. (1999). Using genetic programming to predict financial data. In *Proc. of the Congress on Evolutionary Computation* (Vol. 99, pp. 244–251).
- Jensen, M. H., Johansen, A., Petroni, F., & Simonsen, I. (2004). Inverse statistics in the foreign exchange market. *Physica A*, 340, 678–684.
- Lee, S. J., Ahn, J. J., Oh, K. J., & Kim, T. Y. (2010). Using rough set to support investment strategies of real-time trading in futures market. *Applied Intelligence*, 32(3), 364–377.
- Mabu, S., Hirasawa, K., & Hu, J. (2007). A graph-based evolutionary algorithm: Genetic network programming (GNP) and its extension using reinforcement learning. *Evolutionary Computation*, 15(3), 369–398.
- Mabu, S., Izumi, Y., Hirasawa, K., & Furuzuki, T. (2007). Trading rules on stock markets using genetic network programming with candle chart. *SICE Transaction*, 43(4), 317–322 (in Japanese).
- Sevastianov, P., & Dymova, L. (2009). Synthesis of fuzzy logic and Dempster-Shafer Theory for the simulation of the decision-making process in stock trading systems. *Mathematics and Computers in Simulation*, 80(3), 506–521.
- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning – an introduction*. Cambridge, Massachusetts, London, England: MIT Press.
- Torrubiano, R. R., & Suárez, A. (2010). Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints. *Computational Intelligence Magazine*, 5, 92–107.
- Wang, L., Mabu, S., Ye, F., Eto, S., Fan, X., & Hirasawa, K. (2009). Genetic network programming with rule accumulation and its application to tile-world problem. *Journal of Advanced Computational Intelligence and Intelligent Informatic*, 13(5), 551–560.